A MATLAB TUTORIAL FOR MULTIVARIATE ANALYSIS

Royston Goodacre
Department of Chemistry, UMIST, PO Box 88, Sackville St, Manchester M60 1QD, UK.
R.Goodacre@umist.ac.uk   T: +44 (0) 161 200 4480   F: +44 (0) 161 200 4519

The files you'll need are available at http://personalpages.umist.ac.uk/staff/R.Goodacre/mat_tut/

Introduction

These notes hopefully serve as an introduction to the wonderful world of Matlab. They cover the basics that are needed in order to carry out multivariate analysis (MVA). Specifically they will give details of how to do and view the results of principal components analysis (PCA), discriminant function analysis (DFA) and hierarchical cluster analysis (HCA). If more in depth 'programming' is needed then please RTM.

Whilst Matlab can do things like PLS, ANNs and some evolutionary programming you will need to purchase these toolboxes yourselves and learn how to use them. I am sorry but I will not support these, they already exist within the group (outside Matlab) and you should approach the relevant people.

The MVA processes described below are covered in detailed in:

- Timmins, É.M., Howell, S.A., Alsberg, B.K., Noble, W.C. and Goodacre, R. (1998) Rapid differentiation of closely related *Candida* species and strains by pyrolysis mass spectrometry and Fourier transform infrared spectroscopy. *Journal of Clinical Microbiology* **36**, 367-374. [CANDIDA.PDF]
- Goodacre, R., Timmins, É.M., Burton, R., Kaderbhai, N., Woodward, A.M., Kell, D.B. and Rooney, P.J. (1998) Rapid identification of urinary tract infection bacteria using hyperspectral, whole organism fingerprinting and artificial neural networks. *Microbiology* **144**, 1157-1170. [UTI.PDF]

It would be nice if you would please cite these two when publishing your work.

Any commands for typing into the Matlab command window in this document appear in the `Courier font`. Any responses from Matlab in the Arial font.

It took me, without any tutorials, approximately 6 months to get where I am now with Matlab and I hope that I am still improving. I would add that the expertise I have in chemometrics has taken significantly longer! So have fun, persevere and happy ('Pr'/M)atlabing…

Help

All the functions that are used have some help associated with them whether they are from Matlab, Dr Bjørn Alsberg or myself. Please read them. Access PCA for example by typing:

```
» help pca
```

Help on its own simply returns the topic areas (directories on hard disk).

To use this help engine you need to know the function. This is not always the case so please use the 'Help desk (html)'. This allows searches of Matlab produced functions and not any written in-house.

Demonstrations of some of Matlab's functions can be accessed by typing:

```
» help demo
```

and in particular you are encouraged before starting to type:

```
» demo matlab
```

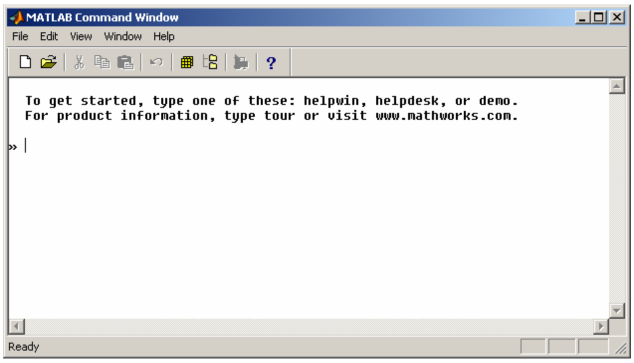
includes lots of information, of particular use is the section on matrices, and graphics.

Of course for some bed time reading there is always the manual…

Finally, please use me after you have given it a good sweat. But bribery does work…

Starting Matlab

Like any other program double click on the Matlab icon or go via the 'start' menu. This will bring up the Matlab command window:



```
 To get started, type one of these: helpwin, helpdesk, or demo.
 For product information, type tour or visit www.mathworks.com.

» |
```

We are now read to have fun…

Basics

In order to do things you type them into the command window, and Matlab will perform the task you have requested. The following are some need-to-know things:

Matlab is case sensitive to matrices/arrays help within it, and to functions, but insensitive to directories. It is easier to always work in lower case!

To change directory type:

```
» cd 'e:\dir1\sub dir2\'
```

The use of '' means that directories with spaces may be used. All the usual DOS changing directory commands also work.

To see what is in a directory type:

```
» dir
```

To see which directory you are currently in:

```
» pwd
```

If in the above directory you will see that this produces

```
ans =
e:\dir1\sub dir2
```

note that a matrix within Matlab called ans has been created.

To create a matrix, type:

```
» a = [1 2; 3 4; 5 6]
```

or

```
» a = [1 2;
3 4;
5 6]
```

both will return

```
a =

   1   2
   3   4
   5   6
```

to the command window.

But if you do not want the data displayed type

```
» b = [100, 101; 211 1];
```

the ‘;’ will not return the output to the screen.

To find out what b contains, type

```
» b
```

command window now returns

```
b =

  100  101
  211    1
```

To see what matrices are in Matlab type:

```
» whos
```

This will return any matrix in Matlab’s memory. For example:

```
Name      Size      Bytes    Class

a         3x2       48       double array
ans       1x26      32       char array
b         2x2       32       double array

Grand total is 35 elements using 130 bytes
```

note ‘a’ and ‘b’ is an array of numbers and ‘ans’ an array of characters.

To save the matrices/arrays type:

```
» save myfile1
```

This saves a Matlab machine code file called ‘MYFILE1.MAT’ *in the current directory*. Please note that Matlab does not save files automatically if you quit without saving there is no 'are you sure?' button and you will loose everything. Also Matlab does not track what you type into it you need to do this separately in a text file. Annotate this profusely because coming back to some maths 6 months later can be hard on the old grey material. More of this later.

To load .MAT files and text files type either:

```
» load myfile1
» load mydata.txt
```

Easy isn't it!

But beware if loading in text files there are some little rules; (1) if the text file contain more than one number, i.e. it is a matrix, then it has to be filled fully otherwise it will not load, (2) avoid text files with names with numbers at the beginning, these are turned into matrices with numbers at the front and Matlab will think they are numbers not arrays and will not act on them, (3) the extension on text files will disappear, (4) files are always loaded in lower case font, (5) if loading in character arrays then these need to be of equal length and each entry encapsulated in ''. More of this later.

Matlab has a memory of what has been typed before. You can navigate this by using the ↑ and ↓ arrow keys. When you have typed lots of things this can become tedious, however if you know that a line started with the letter 'p' type:

```
» p
```

follow this with the ↑ and ↓ arrow keys to find everything started with a 'p' Bob. You can refine this and use 'plot↑' to jump to lines starting with plot…

Usually you do this to modify a line and rerun the command. You can navigate the individual code lines using the ← and → arrow keys, the Delete key deletes and Home and End keys let you jump to the appropriate ends. Where the cursor is sat is where anything typed appears; overwrite does not exist.

What do I do if I have a line which I know is rubbish and want to delete part or all of it? Move to the LHS of anything you want deleted, using the ← and → arrow keys, and type '`Ctrl K`'. This deletes everything to the right of the cursor.

To delete matrices from Matlab use the command clear:

```
» clear ans
```

This removes only the matrix 'ans'.

```
» clear a b c
```

This removes the matrices 'a', 'b' and 'c'.

```
» clear
```

This removes everything, be careful with this one!

To close Matlab happily type:

```
» exit
```

I reiterate that Matlab does not save files automatically so make sure you have.

Matrix algebra

Matlab is a mathematical laboratory and works on matrix algebra. It requires no special handling of vector or matrix maths, so you do not have to learn C^{++}.

In order to get the most out of Matlab and have some recognition of what to do when things appear to go pear shaped please spend some time with a matrix algebra book and run and digest the Matlab demonstration on this. To access type:

```
» demo matlab
```

in LHS window select 'Matrices'. In RHS lower window select 'Basic matrix operations', then hit the 'Run Basic matrix …' button.

Please note that matrix names can not start with a number or have a decimal point in them.

Now you need to spend a little time learning how to reference subsets within a matrix. So we will create a 3x4 matrix, type:

```
» a = [1 2 3 4; 5 6; 7 8; 9 10 11 12]
```

Matlab returns

```
a =

    1    2    3    4
    5    6    7    8
    9   10   11   12
```

to the command window.

to reference the first number type:

```
» a(1,1)
```

Matlab returns

```
ans =

    1
```

to the command window.

You can do this for the whole of the matrix by specifying the row and column.

to reference the first row type (imagine this is an FT-IR spectrum):

```
» a(1,:)
```

Matlab returns

```
ans =

    1    2    3    4
```

to the command window

So the ':' means everything

to reference the first column type (imagine this is the first FT-IR wavenumber):

```
» a(:,1)
```

Matlab returns

```
ans =

     1
     5
     9
```

to the command window.

To reference just the 1st and 3rd row type (imagine you are selecting a subset of the FT-IR spectra):

```
» a([1 3],:)
```

Matlab returns

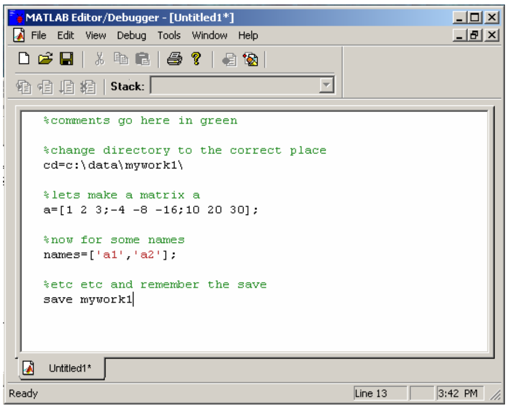
```
ans =

     1     2     3     4
     9    10    11    12
```

to the command window.
So the '[]' can be used to specify an index.

What have I done?

A question that one may be faced with when opening a MYDATA.MAT data file 6 months after having processed it. Unless you have a perfect memory then you must track what you do yourself. *Matlab does not do this for you.*

So a good hint is in the same directory that the data will be kept, place the .MAT file a text file to track what you have done called a DOIT.M file (perhaps you can be more imaginative with the file name than this!). The so called 'm files' are recognized by Matlab and are what all the function files end in. You can create one of these using the Matlab editor/debugger.
Go to the command window, file → new → M-file. The editor/debugger looks like this:


MATLAB Editor/Debugger - [Untitled1*]
File Edit View Debug Tools Window Help
Stack:
Untitled1*
Ready
Line 13
3:42 PM

```
%comments go here in green

%change directory to the correct place
cd=c:\data\mywork1\

%lets make a matrix a
a=[1 2 3;-4 -8 -16;10 20 30];

%now for some names
names=['a1','a2'];

%etc etc and remember the save
save mywork1
```

Alternatively, use NOTEPAD or PFE (program file editor), I use the latter.

PCA

The following is a worked example with PyMS data (pyrolysis mass spectrometry) where the aim was to investigate quantitatively the levels of ampicillin mixed with either *Escherichia coli* or *Staphylococcus aureus*.

> From: Goodacre, R., Trew, S., Wrigley-Jones, C., Neal, M. J., Maddock, J., Ottley, T. W., Porter, N. and Kell, D. B. (1994) Rapid screening for metabolite overproduction in fermentor broths using pyrolysis mass spectrometry with multivariate calibration and artificial neural networks *Biotechnology and Bioengineering* **44**, 1205-1216.

The data is found in file AMPDATA.TXT and the XL file with all the descriptors in AMPEXP.XLS.

The data contains:

42 objects x 150 variables.

in other words 42 quantitative measurements,
with 150 values each (in this case the intensities of m/z 51-200)

The XL file contains information on:

A column of the sample number (the order of data in AMPDATA.TXT),
A column giving details of the bacterial backgrounds,
A column giving details of the levels of ampicillin (in μg.ml^{-1}),
A column of names – used to identify which sample is which.

Load data into matlab:

1. change directory in which you have placed these data
   ```
   » cd ‘c:\this is where\the data\is kept\’
   ```
2. load the data matrix
   ```
   » load ampdata.txt
   ```
3. check it has loaded correctly
   ```
   » whos
   ```

This should return to the command window:

```
  Name          Size        Bytes       Class

  ampdata       42x150      50400       double array

Grand total is 6300 elements using 50400 bytes
```

note that the extension (.txt) has gone

We need to be able to identify the 42 rows (objects) so use that names column in the XL file. The characters need to be entered into Matlab in ‘quotes’ else it thinks they are numbers and is not happy.

A key thing to remember is this is a character array and so each entry needs to be the same size, because the names E1 and E10 are different lengths ∴ E1 becomes ‘E1 ’ and E10 ‘E10’.

Character arrays are entered in the same way as numbers so for these data:

```
» names=['E0 ';'E1 ';'E2 '; … ;'S19';'S20'];
```

or

```
» names=['E0 '
'E1 '
'E2 '
```

```
...
'S19'
'S20'];
```

I am afraid this is hard work. You may wish to use a text editor (e.g., PFE) for this, which can replace the carriage returns with “ ` ” “new line” “ ` ”. I will let you decide which is best.

We now have the two matrices we need, check by typing:

```
» whos
```

This will return

```
Name        Size      Bytes    Class

ampdata     42x150    50400    double array
names       42x3      252      char array

Grand total is 6426 elements using 50652 bytes
```

We are now ready to do some PCA. For help on this function type:

```
» help pca
```

This will return

```
[tt,pp,pr]=PCA(X,comp);
comp is the number of principal components
Here the NIPALS algorithm is used
pr = percentage explained variance
```

This is a function and it has 2 things going into it:

X = the data (ampdata),

comp = the number of principal components to extract (let us use 3).

The function will output 3 matrices, these must be specified else they are not returned. If there is more than one matrix returned then these must be in square brackets and be separated by commas. The 3 things coming out of this are:

tt = the PC scores,

pp = the PC loadings,

pr = the %age explained variance

If this makes no sense then time to read a good MVA book. Chatfield, C. and A. J. Collins (1980). *Introduction to Multivariate Analysis*. London, Chapman & Hall, is a good one or try Manly, B. F. J. (1994). *Multivariate Statistical Methods: A Primer*. London, Chapman & Hall.

So lets do our PCA, I like to track what I do by numbering the matrices at the end, so I would type:

```
» [tt1,pp1,pr1]=pca(ampdata,3);
```

using the ‘;’ is very wise at this point! All that is returned is the incremental value of ‘i = *n*’ which tells you how many (*n*)PCs have been extracted.

To see what is produced type:

```
» whos
```

This will return

```
Name        Size      Bytes    Class

ampdata     42x150    50400    double array
```

```
  names        42x3       252        char array
  pp1          3x150      3600       double array
  pr1          3x1        24         double array
  tt1          42x3       1008       double array

Grand total is 7005 elements using 55284 bytes
```

We have extracted just 3 PCs hence:

tt1 (the scores matrix) is 42 rows (objects)
by 3 columns (PC1, PC2 and PC3)
pp1 (the loading matrix) is 3 rows (PC1, PC2 and PC3)
by 150 columns (loading values for each m/z)
pr1 (explained variance) is 3 by 1 for %age explained for (PC1, PC2 and PC3)
note, this is cumulative %ages

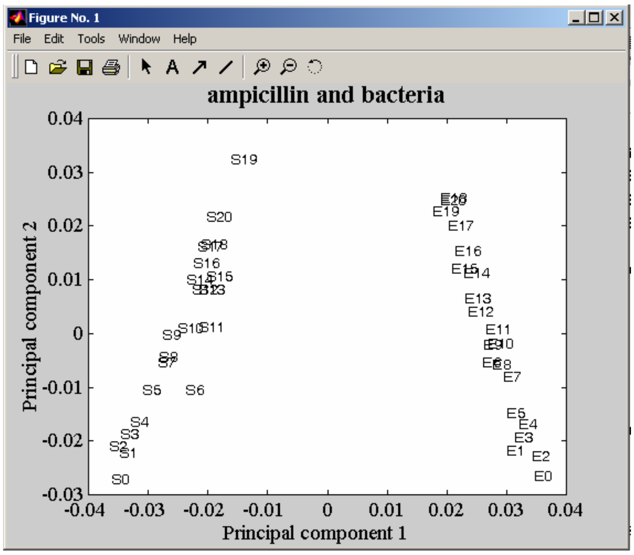
To visualize the outcome we need to do some plotting. You can do this the hard way using the plot function or the nice one called PLOT_PCA.M; help is available:

```
    » plot_pca(tt1,1,2,names);
```

You will be prompted for a title to the plot
For no title just do quotes ‘’.

```
    Title please (in quotes)  'ampicillin and bacteria'
```

This produces:



This can be copied and pasted into word using the edit → copy figure command.

We can see (1) left to right differentiates between *E. coli* (E) and *S. aureus* (S)
(2) ampicillin concentration increases from bottom to top.

Finally remember to save your work!

```
    » save amp_work;
```

Getting FT-IR data into Matlab

The data that is produced by the FT-IR spectrometer is in its own machine language and needs to be converted into ASCII (text) prior to importing it into Matlab.

Follow these steps in Windows^{NT}:

1. In Windows Explorer or File Manager change directory to the one with the relevant file(s).
2. Place the conversion program OPUS2NT.EXE into this directory.
3. We next make a batch file to do the conversion(s):
    a. open notepad
    b. on the first line type: opus2nt inputfile outputfile
        i. inputfile will be something like mydata1.2 from the IR OPUS software
        ii. outputfile will be something like data1_2.txt
    c. do the same for multiple conversion of files on separate lines
    d. save as (for example) CONVERT.BAT (you need the BAT extension)
4. In Windows Explorer or File Manager double click on CONVERT.BAT

Now follow these steps in Matlab:

1. When starting a new piece of work remember to save what you had previously and clear the worksheet by typing:
```
» save oldwork
» clear
```
2. Change directory to the one with the convert file data1_2.txt:
```
» cd 'e:\dir1\sub dir2\etc\'
```
3. load the file:
```
» load data1_2.txt
```
4. check it is there and it is the correct size:
```
» whos
```

The rest of this document gives details of a worked example with FT-IR data where the aim is to investigate the ability of FT-IR to classify bacteria isolated and cultured from urinary tract infection.

From: Goodacre, R., Timmins, É.M., Burton, R., Kaderbhai, N., Woodward, A.M., Kell, D.B. and Rooney, P.J. (1998) Rapid identification of urinary tract infection bacteria using hyperspectral, whole organism fingerprinting and artificial neural networks. *Microbiology* **144**, 1157-1170. [UTI.PDF]

The data is found in file UTIDATA.TXT and the XL file with all the descriptors in is UTIEXP.XLS.

The data contains:

236 objects x 882 variables.

in other words 236 spectra from 59 bacteria, in quadruplicate,
with 882 values each (the absorbance from 4000-600 cm^{-1})

The XL file contains information on:

A column of the sample number (the order of data in UTIDATA.TXT),
A column giving the identities of the UTI bacteria,
A column giving details of the replicates analysed,
A column of names – used to identify which sample is which.

Prepare data, names and class matrices like before:

1. change directory in which you have placed these data
   ```
   » cd 'c:\this is where\the data\is kept\'
   ```
2. load the data matrix
   ```
   » load utidata.txt
   ```
3. create a names matrix using the column in XL
   ```
   » names=[xl];
   ```
   `xl` here is shorthand to say copy the columns from XL and paste into Matlab, remember the array/matrix needs to be in square brackets.
4. create a class matrix using the column in XL
   ```
   » class=[xl];
   ```
   This is used to specify to Matlab which samples are replicates. Note, it is more normal to have these in the order;
   1, 2, 3 …, *n*, 1, 2, 3 …, *n*, 1, 2, 3 …, *n*.
5. Another way of creating the class matrix is to use the groups function
   ```
   » class=groups(59,4,2);
   ```
   Syntax is shown by typing » `help groups`
   ```
   [group] = groups(samps,reps,A)
    returns a group files for dfa in order
      A=1     ABC repeat (i.e., A,B,C, ...., A,B,C etc)
      A=2     AAA repeat (i.e., A,A,A, ...., Z,Z,Z)
    samps = number of groups
    reps = number of replicates
   ```
6. check that all matrices are in memory
   ```
   » whos
   ```
   This should return to the command window:
   ```
   Name        Size       Bytes      Class

   Class       236x1      1888       double array
   Names       236x1      472        char array
   utidata     236x882    1665216    double array

   Grand total is 208624 elements using 1667576 bytes
   ```

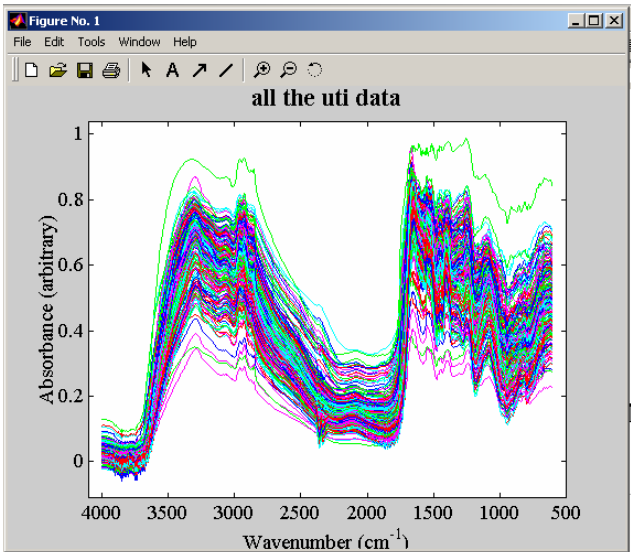
If the data have been collected as absorbance spectra as these have then we can start to scale the FT-IR data. You can check this by typing:

```
» plotftir(utidata)
```

You will be prompted for a title

```
Title please (in quotes)  'all the uti data'
```

This produces:



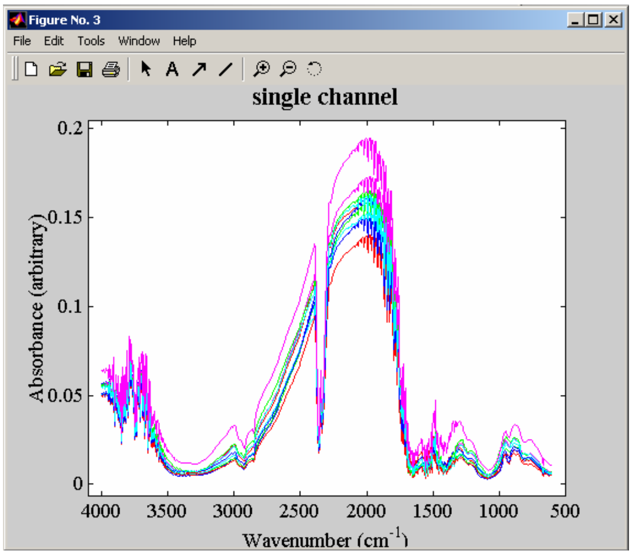
This can be copied and pasted into word using the edit → copy figure command.

Formatting single channel FT-IR spectra

Most data is now collected as single channel spectra so you will need to load:

1. The spectra from the 100 blank wells. » `load blank.txt`
2. The spectrum from the 1 reference well. » `load refwell.txt`
3. The spectra from the 100 filled wells. » `load filledwells.txt`

If you plotftir these as above then the spectra will look 'strange':



In order to produce the absorbance spectra for first 50 wells according to Beer-Lambert Law, type:

```
» abs1=-log10([filledwells(1:50,:)]./[blank(1:50,:)]);
```

the square brackets are not compulsory but they are here to highlight that the operator is './', check help and demo Basic matrix operations for the meaning of this.

Then check that the spectra look good. I'll let you judge this but rule of thumb is to take the highest peak per spectrum (usually Amide I, 1660 cm^{-1}) and see if it lies between ~0.25 t0 ~1.4 absorbance. If it is below this then there is too little IR light being absorbed, if too high then Beer-Lambert does not hold as the relationship between light in and light out becomes non-linear.

To create the absorbance spectra using just the single reference well, for first 50 wells type:

```
» for i=1:10
abs2(i,:)=-log10(plate1samp(i,:)./plate1ref(1,:));
end
```

OK so this is a bit complex having to do a for loop, but it does begin to show you the power of Matlab.
For the use of matrix indexing refer to the basic section above.

CO_2?

It may be necessary to remove CO_2 vibrations from the spectra. This can be done using the CO2CORR.M function. This removes the CO_2 peaks (the bands 2403-2272 cm^{-1} and 683 to 656 cm^{-1}), best to fill the offending area with a trend then one can still plot nice spectra (that is to say the wavenumbers are still evenly spaced). Type:

```
» co2basl=co2corr(utidata,1);
```

Scaling the FT-IR data

Before doing any analysis it will be necessary to scale the FT-IR data. I have written lots of variants, the most popular of which are:

SCALEM.M	scales so that the min and max bins are between 0 and 1.
NORMHIGH.M	normalizes to the highest peak in each spectrum, highest =1.
NORMTOT.M	normalizes so the total area = 1.
VECNORM.M	performs vector normalization.
DETRENDM.M	detrends from first and last bin.
DERIVATS.M	calculate Savitzky & Golay derivatives.

Obviously it would be nice to know how to scale our data. There is however no accepted protocol here or in the literature and this is where I say "I'm afraid what you have to do is data dependent and task dependent…". All of the above have their pros and cons.

I will say that for FT-IR data I have generally found that if the data are all very similar in shape and magnitude that SCALEM is a good start, so type:

```
» scaled=scalem(co2basl);
```

Inspect the results visually, it may be that DETRENDM is also worth doing on the scaled data, if so type:

```
» detrended=detrendm(scaled);
```

Again view the results using the PLOTFTIR function.

If the results of this do not look too good then take Savitzky & Golay derivatives, to take the 1st derivative type:

```
» diff1=derivats(detrended,5);
```

The window size of 5 is the default.
Due to the windowing the new matrix diff1 decreases from 882 to 878.

To take the 2nd derivative, type:

```
» diff2=derivats(diff1,5);
```

Due to the windowing the new matrix diff2 decreases from 878 to 874.

It is acceptable to take derivatives straight from the raw data, scaled or detrended. This is where you will have to try optimizing the pre-processing regimes.

For the UTI data the following route is best:

Raw data → CO_2 corrected → scaled using SCALEM → detrended → 1st derivative spectra.
It is these we will now use for MVA.

PCA on the UTI FT-IR data

Remembering which pre-processing you have used, perform PCA as before:

```
» [tt2,pp2,pr2]=pca(diff1,20);
```

i will 'count' from 1 to 20

Check that it has worked:

```
» whos
```

This should return to the command window:

```
Name        Size      Bytes     Class

class       236x1     1888      double array
co2basl     236x882   1665216   double array
detrended   236x882   1665216   double array
diff1       236x878   1657664   double array
diff2       236x874   1650112   double array
names       236x1     472       char array
pp2         20x878    140480    double array
pr2         20x1      160       double array
scaled      236x882   1665216   double array
tt2         236x20    37760     double array
utidata     236x882   1665216   double array

Grand total is 1268852 elements using 10149400 bytes
```

We have extracted 20 PCs hence:

tt2 (the scores matrix)	is 236 rows (objects) by 20 columns (PC1, PC2, …, and PC20)
pp2 (the loading matrix)	is 20 rows (PC1, PC2, …, and PC20) by 878 columns (loading values for each derivatised wavenumber)
pr2 (explained variance)	is 20 by 1 for %age explained for (PC1, PC2, …, and PC20) note, this is cumulative %ages

DFA on the UTI FT-IR data

DFA usually starts with the principal components (PCs) scores as the starting point with some *a priori* knowledge of which sample is which. This *a priori* knowledge is most often the replicates (from the class matrix) and when doing the analysis in this way helps to remove any sample presentation differences. Once can use different *a priori* knowledge but if you do this, this needs to be done with care and cross validated.

To look at the syntax for dfa type:

```
» help dfa
```

This should return to the command window:

```
[U,V,eigenvals] = DFA(X,group,maxfac)
  Performs DISCRIMINANT FUNCTION ANALYSIS

  INPUT VARIABLES

  X           = data matrix that contains m groups
                Dim(X) = [N x M]. All columns must be independent.
  group       = a vector containing a number corresponding
                to a group for every row in X. If you have
                m groups there will be numbers in the range
                1:m in this vector.
  Maxfac      = the maximum number of DFA factors extracted

  OUTPUT VARIABLES

  U           = DFA scores matrix (Dim(U) = [N x maxfac])
                the eigenvalues are multiplied with each column
                in this matrix.
  V           = DFA loadings matrix, Dim(V) = [M x maxfac]
  eigenvals   = a vector of DFA eigenvalues
```

You will note that the syntax is very similar to PCA except one extra matrix called group is required.

To perform DFA on the first 20 PCs with the *a priori* knowledge of replicates, type:

```
» [U2,V2,eigenvals2] = DFA(tt2,class,10);
```

We are extracting 10 DFs (discriminant functions).
A useful tip is to increment the output matrices so they match the inputs,
hence why we use U2, V2 and eigenvals2 with tt2.

We have extracted 10 DFs hence:

U2 (the scores matrix)	is 236 rows (objects) by 10 columns (DF1, DF2, …, and DF10)
V2 (the loading matrix)	is 10 columns (DF1, DF2, …, and DF10) by 20 rows (loading values for PCs 1-20)
eigenvals (eigenvalues)	is 10 by 1 for eigenvalues (DF1, DF2, …, and DF10) note, this is individual eigenvalues per DF.

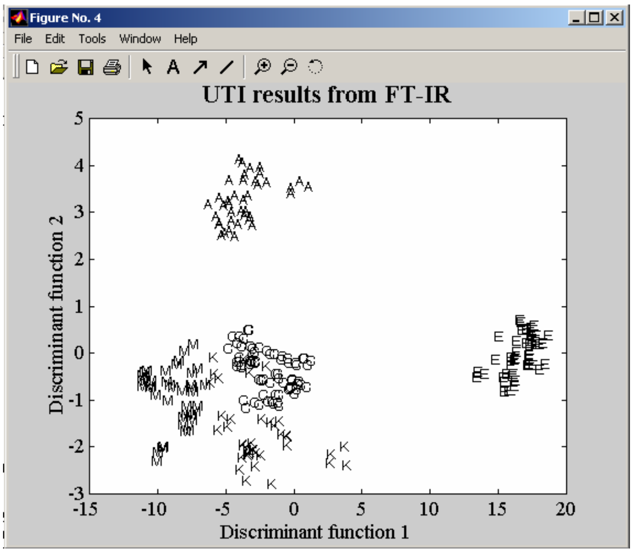
To visualize the DFA scores use PLOT_DFA.M; help is available:

```
» plot_dfa(U2,1,2,names);
```

You will be prompted for a title to the plot
For no title just do quotes ‘’.

```
Title please (in quotes)  'UTI results from FT-IR'
```

This produces:



This can be copied and pasted into word using the edit → copy figure command.

We can see (1) *Enterococcus* spp. (E) clearly separated in DF1, just as well (!) these are Gram +ve all the rest are Gram –ve.

(2) *Pseudomonas aeruginosa* (A) clearly separated in DF2.

(3) *Proteus mirabilis* (M) also separated in DF1 from *Escherichia coli* (C) and *Klebsiella* spp. (K).

(4) *E. coli* (C) and *Klebsiella* spp. (K) appear to be separated, this is highlighted if you view the 3rd DF by plotting:

```
» plot_dfa(U2,1,3,names);
Title please (in quotes)  'DF1 vs DF3 for UTI'
```

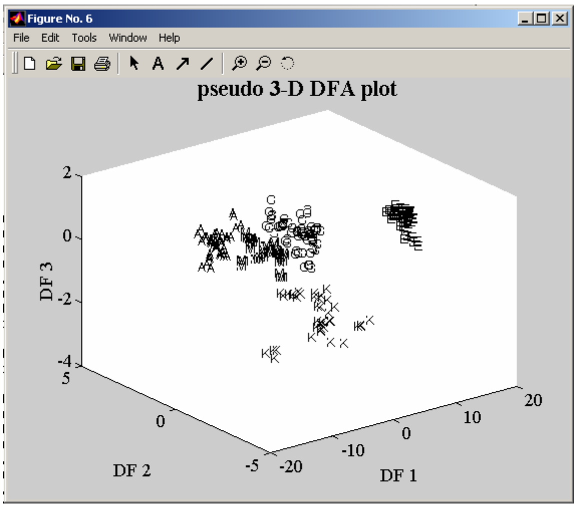
This illustrates that it is not just the first 2 DFs that are important and that lower ones can be valuable.

We can plot 2-D slices through the 3-D cube by typing:

```
» plotdfa3(U2,1,2,3,names)
Title please (in quotes)  'pseudo 3-D DFA plot'
```

You then often need to ‘optimize’ the view.

This produces:



HCA on the UTI FT-IR data

When the DFA plot starts to look complex, or if trees are your thing, then you can do HCA. This is performed by a program outside Matlab. Matlab is used to pass the relevant parameters to the agglomerative clustering program OCNT.EXE. Full help is available on the syntax by typing:

```
» help oc_clustering
```

The help is extensive and I will not go into it here.

We could do HCA on the entire DF scores matrix but this would produce a dendrogram which would have all the quadruplicate measurements (236 leaves on the trees). Much more sensible to use the DF score means of these groups (i.e., only 59 leaves). In order to do this type;

```
» U2avg=meanidx(U2,class);
```

U2avg is the mean matrix from matrix U2 based on the replicates structure found in the matrix class.

You can check this has worked by typing:

```
» whos U2*
```

This should return to the command window:

```
Name        Size        Bytes       Class

U2          236x10      18880       double array
U2avg       59x10       4720        double array

Grand total is 2950 elements using 23600 bytes
```

Note that U2* just returns a subset of the matrices to the command window.

We also need to make a matrix that has the names associate with the average DF scores (U2avg). You can do this by making a new matrix in XL and then pasting it into Matlab by typing:

```
» namesavg = [xl]
```

`xl` refers to the column in XL you have just made.

Alternatively since we know that the names are in blocks of 4 we can type:

```
» namesavg = names([((1:59)*4)-3],:);
```

we are using complex indexing here.

Or we can use the NAMESIDX.M function and type:

```
» namesavg = namesidx(names,class);
```

this uses the same class file as used for MEANIDX.

Best to check that this has worked by plotting it:

```
» plot_dfa(U2avg,1,2,namesavg);
Title please (in quotes)  'DFA average data'
```

You can then compare this with the non-average version we plotted earlier.

We are now ready for the HCA analysis, type:

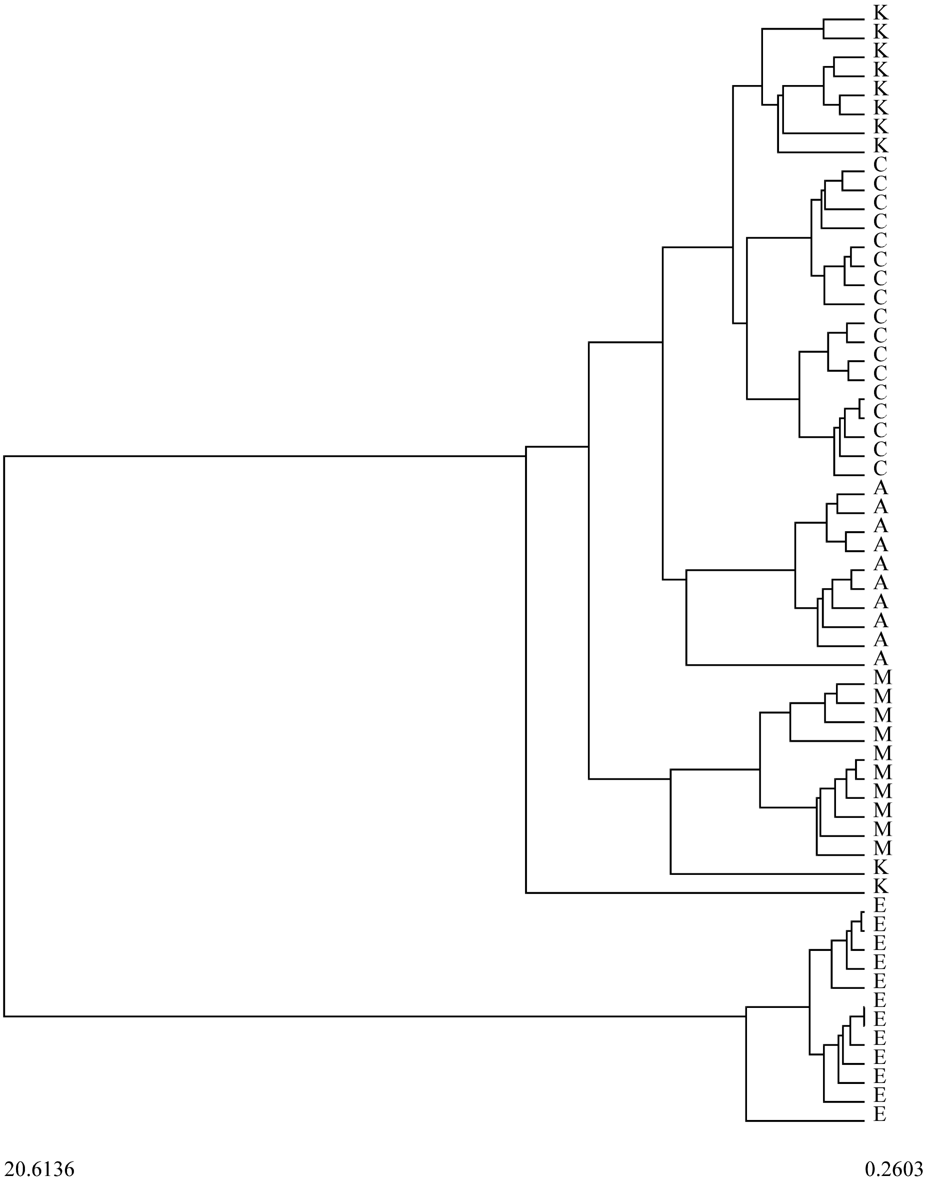
```
» oc_clustering(U2avg,namesavg,'Mink',2,'dis means ps util amps tt');
```

This should return to the command window:

```
Reading Upper Diagonal
```

```
Read: 59 Entries
CPU time: 0.000000 seconds
Setting up unclust
Setting up notparent
Setting up clust
CPU time: 0.000000 seconds
Means linkage on distance
Doing Cluster Analysis...
Opening AMPS order and tree files for writing
Total CPU time: 0.800000 seconds
```

The results appear in the directory you are currently in. Five files are produced; TT.TORD, TT.TREE, UTI1.PS, OCRESULTS.DAT, OCTEMP.DAT. It is the postscript file UTI1.PS that has the dendrogram in it and you have to read this using your favourite postscript program. The results should like:



Finally remember to save your work!

```
    » save uti_work;
```