Sequence Analysis
Essex Summer School

Brendan Halpin
University of Limerick[1]

July 13-15 2007

[1]Contact: brendan.halpin@ul.ie

Contents

1 Friday afternoon

1.1 Start here!

Each time you start Stata, you will need to issue two commands:

```
cd s:/sequence/practicals/lab1
adopath +s:/sequence/practicals/utilities
```

The location `s:/sequence/practicals/` is just for example; the precise location will not be known until after these notes are printed, but will be given to you. (Note that Stata treats backslashes (\) as special, and accepts forward slashes (/) in their place.)

1.2 A simple example

Use the code example below to calculate OM distances. It first reads in a small number of short sequences (variables `s1` to `s5` are the elements, `len` is the length, maximum 5, there are four states – examine the data file). It then sets up the substitution matrix as a Stata matrix.

```
#delimit ;
infile  id len s1 s2 s3 s4 s5 using lab1small.dat;

matrix subsmat = ( 0,1,2,3 \
                   1,0,1,2 \
                   2,1,0,1 \
                   3,2,1,0 );
```

The "`#delimit ;`" statement allows us to have multi-line commands in Stata do-files, terminated with a semi-colon. Note the syntax for the matrix: comma joins elements within rows, backslash joins rows.

The `oma` command calculates all pairwise distances. It takes the list of sequence elements as its parameters (`s1-s5`) and has four mandatory options (*i.e.*, after the comma):

- `subs(`*matrixname*`)` for the substitution cost matrix
- `indel(`*number*`)` for the indel cost
- `pwdist(`*matrixname*`)` for the matrix in which the results are to be returned (will be created)
- `length(`*len*`)` for sequence length (a variable or a number)

Since the highest substitution cost is 3, the indel cost must be ≥ 1.5.

```
oma s1-s5, subs(subsmat) indel(2)
         pwdist(distmat)
         length(len);
matrix list distmat;
```

We follow the `oma` command by `matrix list distmat` which lists the result matrix. Since this is symmetric, only the lower triangle is shown.

For small data sets, the `svmat` command is useful. `svmat distmat, names(dist)` appends the results to the data set. (A fuller discussion of saving matrices is at section 2.6.)

Saving the distances as variables can be useful for comparison with a small number of reference sequences.

1.3 Looking at the internals

The `oma` command has fifth option, `workspace(`*yes*`)`, which is optional and will cause the working matrices to be displayed (do not use for big data sets!). This prints out the matrix of actual substitution costs between the pair of sequences, and the $m + 1$ by $n + 1$ matrix of calculations (where m and n are the lengths of the two sequences). Note that column 1 and row 1 of the substitution matrix are filled with zeroes to correspond.

```
oma s1-s5, subs(subsmat)
           indel(2)
           pwdist(distmat)
           length(len)
           workspace(yes);
```

See if you can make sense of the operation of the OM algorithm as it works through the workspace matrix.

1.4 Costs

Experiment with the code: add your own sequences (by editing the data file or changing values in the data editor), change the substitution costs, change the indel cost, and see what happens.

1.5 Using SQ

Kohler et al's `SQ` add-on is also easy to use. It needs the sequences in "long" rather than "wide" format (one record per person–month, for example, rather than one record per person with one variable per month). For this we need to "`reshape`" the data: list the data before and after issuing the command, to be sure you understand what it does.

```
#delimit ;
infile id len s1 s2 s3 s4 s5 using lab1small.dat;

reshape long s, i(id) j(m);
```

In the data, where sequences are shorter than 5 elements, the remaining variables are set to -1. After `reshape`, these are now cases, and we can drop them, prior to telling Stata that the data are sequences (`sqset`) with state in `s`, sequence identifier in `id` and the sequence order in `m`:

```
drop if s==-1;
sqset s id m;
```

`SQ` has a number of useful utilities: `sqdes` describes the sequenes, and `sqtab` tabulates them.

```
sqdes;
sqtab;
```

Finally, to run OM:

```
matrix subsmat = (0,1,2,3 \
                  1,0,1,2 \
                  2,1,0,1 \
                  3,2,1,0  );

sqom, subcost(subsmat) indelcost(2) full standard(longer);
matrix list SQdist;
```

Note a major difference: `SQ` spots that sequences 1 and 4 are identical, and drops sequence 4. For more information on `SQ`, do `help sq` and `help sqom`.

`SQ` has a lot of advantages, and I recommend you explore its capabilities. Start with `help sq`. However, it has a single important limitation: it uses Stata's "Mata" matrix language to calculate the OM distances, and this is substantially slower than using C plugins, as the `oma` command does. With Stata 9 the time difference is of the order of a factor of 40, though that is likely to reduce as Mata matures.

1.6 Using TDA

TDA has a fast implementation of OM, and has clustering capabilities on a par with Stata. To calculate OM distances with TDA, use code like the following:

```
nvar(dfile = exampleprog.dat,
     ID = c1, S1 = c2, S2 = c3, S3 = c4, S4 = c5, S5 = c6, );
seqdef()=S1,,S5;
```

This reads in `ID` and `S1` to `S5` from a file, and then declares `S1` to `S5` to be sequence data.

```
mdef(SCOST,4,4)=
0.0, 1.0, 2.0, 3.0,
1.0, 0.0, 1.0, 2.0,
2.0, 1.0, 0.0, 1.0,
3.0, 2.0, 1.0, 0.0;

seqm(scost = SCOST,
     icost = 2)=exampleprog.dist;
```

This defines the substitution cost matrix (`SCOST`) and fits all pairwise OM distances. `icost = 2` sets the indel cost, and the distances are written to `exampleprog.dist`.

1.7 Using real data

If you would like to fit OM to a real data set, `../birthseq.dta` contains 73 months of labour market experience of women who have a birth in month 25, with a 4-state state space (full-time employed, part-time employed, unemployed and non-employed). Using the syntax above, carry out an OM analysis on the two-year period between months 13 and 36.

2 Saturday morning

2.1 Reminder

Each time you start Stata, you will need to issue two commands:

```
cd s:/sequence/practicals/lab2
adopath +s:/sequence/practicals/utilities
```

Note that each lab has its own subdirectory.

2.2 Hamming distance

Hamming distance compares sequences of equal length, element by element. It can use a substitution matrix like OM but because it does not do indels, it can only recognise similarity at the same location. The example file, `hamming.do`, calculates Hamming distances for all pairwise combinations of sequences in the `hsmall.dat` file. It uses the file `hamming.ado`, which contains Mata code (Stata's powerful new matrix language) to do this.

Verify one or two of the pairwise distances by hand: they are the sum of the element-wise substitution costs, divided by the length.

2.2.1 Compare with OM

Using syntax from lab 1, generate OM pairwise distances for the same data set, using the same substitution matrix and an indel cost of 2. Compare your results, and see if you can explain the similarities and differences.

Re-fit the OM distances using an indel cost of 20. Explain the results.

2.3 Clustering with OM distances

We can easily run cluster analyses on our distance matrices in Stata, using its `clustermat` command. Generate an OM distance matrix using the `birthseq` data set, which contains labour market histories before and after a birth for a sample of women (drawn from the BHPS). The four states are full-time employed, part-time, unemployed and out of the labour market. Use months 13 to 36 (the birth happens at month 25):

```
set matsize 700;
use pid state13-state36 using ../birthseq;
oma state13-state36, subsmat(subsmat) indel(2)
                     pwdist(bsdist) length(24);
```

Matrix size needs to be increased to deal with the $\frac{675 \times 674}{2}$ distances that will be calculated, and the results end up in matrix `bsdist`.

The `clustermat` command takes two parameters, the type of clustering to be done (for now we will use Wards' method) and the matrix. We also need the option `add`, to add the cluster results to the data set:

```
clustermat wards bsdist, add;
```

This creates no output but we can get dendrograms and cluster groupings easily.

```
cluster dendrogram, cutnumber(40);
```

The `cluster dendrogram` will draw a dendrogram, but it has to be limited in the number of groups (`cutnumber`) since it can't manage 675 cases.

We can create a cluster grouping readily as follows;

```
cluster g8=groups(8);
```

The number of clusters is to be chosen by the analyst – 8 may be too big or small.

Create a 16-group solution also, and crosstabulate them:

```
tab g16 g8;
```

Relate the structure in the table to the dendrogram.

2.3.1 Representing sequences as strings

Sequences can be difficult to view when they are spread over so many variables. We can collapse them into a single string variable with code like the following:

```
gen str24 stripe = "";
foreach x of varlist state13-state36 {;
replace stripe = stripe + "F" if ‘x’==1;
replace stripe = stripe + "p" if ‘x’==2;
replace stripe = stripe + "U" if ‘x’==3;
replace stripe = stripe + "n" if ‘x’==4;
};
```

With this variable it becomes easier to look at the cluster contents. Experiment:

```
sort g8 g16 stripe
list g8 g16 stripe if g8==4,clean
```

2.3.2 Regular expressions

Regular expressions (or "regexes") are representations of patterns in strings.

- `a*b`: zero or more `a`s followed by a `b`
- `a+b`: at least one `a` followed by a `b`
- `a?b`: zero or one `a`s followed by a `b`
- `a.+b`: an `a` followed by at least one character, followed by a `b`
- `(ab)+c`: one or more `ab`s followed by a `c`
- `^abc`: a string starting with `abc`
- `abc$`: a string ending with `abc`
- `a[bcd]+e`: `a` followed by at least one of `b`, `c` or `d`, followed by `e`

Stata has regex capability. Combined with string-representations of sequences, regexes provide a very powerful way of selecting sequences for viewing or further processing.

Try the following:

```
list stripe if regexm(stripe,"^F.+n$")
list stripe if regexm(stripe,"^F+p+n")
list stripe if regexm(stripe,"^FFFFFFFFFFF.+pn")
```

This can be a very useful way of characterising cluster groups.

2.4 Cluster Hamming distances and compare

Calculate Hamming distances for the same data, and compare the resulting clusters with those from OM. Which sorts of sequence sort similarly, and which differently? Use crosstabs of the cluster grouping and the `stripe` variable, and perhaps the `regexm` function, to examine the results.

2.5 Clustering with TDA

TDA will conduct a cluster analysis directly on the `exampleprog.dist` file of pairwise distances. This file has a particular structure. There is a line for each distinct pair of sequences, with five columns: sequence 1 id, sequence 2 id, sequence 1 length, sequence 2 length, OM distance. In TDA the OM distance is not automatically standardised to sequence length.

```
nvar(
    dfile = examplebstda.dist,
noc = 4950,
A = c1, B = c2, C = c3, E = c4, D = c5,
);
```

This code reads in distances from a file. Column 5 has the distances and this is put into variable `D`. There are 4950 pairwise distances in the example, representing 100 sequences.

```
gdd(opt=4) = D;

hcls(
    opt = 2,
    pcf = cl1plot.cf,
    df = cluster1.df,
) = d2;
```

The `gdd` command creates a graph structure, and the `hcls` command conducts the cluster analysis, writing a set of graphics commands to `cl1plot.cf`. Running TDA on this file will create a Postscript graphic of the dendrogram, in `cl1plot.cf.ps`. The underlying clustering information is written to `cluster1.df`.

If you want to create cluster groups, you need to use the `hclsp` command. Full information is available in the TDA manuals (section 7.5.1).

2.6 Saving distance matrices

OM can take a long time to run, for large numbers of long sequences, and it is sometimes useful to save its output for future use. There are a number of ways to do this.

```
svmat distmat, names(d)
```

`svmat` takes a Stata matrix (in this case, `distmat`) and appends it to the data set, with columns of the matrix becoming variables, named `d1` to `d`*n* where *n* is the number of sequences.

We can save this data set and re-read it later, recreating the matrix using the Stata command `mkmat`:

```
mkmat d1-d675, matrix(distmat)
```

This technique has limitations where there are high numbers of sequences.

A second technique uses Mata, which is much less limited, which can save individual matrices to files. First we need to use the Mata function `st_matrix` to read the Stata matrix into a Mata matrix, and then the `mata matsave` command to save it:

```
mata:
DM = st_matrix("distmat")
mata matsave distances DM, replace
end
```

This saves the matrix as the file `distances.mmat`. This file can be read at a later time as follows:

```
mata:
mata matuse distances
st_matrix("distmat", DM)
end
```

The `st_matrix("distmat", DM)` command puts the Mata matrix `DM` into the Stata matrix `distmat`, which is then accessible to commands like `clustermat`.

3 Saturday afternoon

3.1 Multidimensional scaling

The "other" thing to do with similarity or dissimilarity matrices, after clustering, is multidimensional scaling. A simple and robust form of MDS is principal components analysis. This attempts to construct an N-dimensional space so that the distances between the cases can be represented by their locations in this space. N is ideally small.

Stata takes a dissimilarity matrix, say `rom`, and does PCA on it as follows:

```
mdsmat rom, dim(3)
```

The `dim(3)` saves the three largest principal components (more can be saved but it is difficult to interpret more than three).

3.1.1 Run a PCA

Run PCA on the distance matrix from an optimal matching run, and examine the results. How well do the first three components summarise the distances?

The components can be copied into variables with the commands:

```
matrix D=e(Y);
svmat D;
```

This will create variables `D1`, `D2` and `D3`.

3.1.2 Examine the dimensions graphically

Plot them against each other:

```
graph matrix D1 D2 D3
```

Plot pairs with labels:

```
twoway (scatter D1 D2,mlabel(stripe));
twoway (scatter D1 D3,mlabel(stripe));
twoway (scatter D2 D3,mlabel(stripe));
```

3.1.3 Examine the dimensions in Stata

Examine the data within Stata: how do the sequences change as you go from one end to the other end of a dimension?

```
sort D1 D2 D3
list stripe D1 D2 D3,clean
sort D2
list stripe D1 D2 D3,clean
sort D3
list stripe D1 D2 D3,clean
```

You should notice that the extreme values of the dimensions often correspond to the simplest sequences, *i.e.*, those with one or very few spells.

3.1.4 Attempt to interpret the dimensions

Sometimes the dimensions are interpretable. In the case of the `birthseq` data, the original substitution matrix defines the state space as uni-dimensional. Indeed it puts the four states on an equally spaced linear scale, with full-time employment at one end and non-employment on the other. If we treat these as scores and simply sum the values for a sequence (so that two years of non-employment is worth 24, and 2 years of full-time is worth 96), the resulting variable is strongly collinear with the first dimension of the PCA.

3.2 Exploring substitution matrices

Using the `birthseq` data, fit OM distances for a number of different substitution matrices. The default we have used to date implies a linear, interval state space. At least compare the default with the other two matrices proposed here, but feel free to try your own too:

```
matrix subsmat1 = (0,1,2,3 \
                   1,0,1,2 \
                   2,1,0,1 \
                   3,2,1,0 );
matrix subsmat2 = (0,1,1,1 \
                   1,0,1,1 \
                   1,1,0,1 \
                   1,1,1,0 );
matrix subsmat3 = (0,1,2,4 \
                   1,0,2,4 \
                   2,2,0,1 \
                   4,4,1,0 );
```

`subsmat2` has a different implication: each state is different from each other state, but they are all equally different. This implies the four states are located at the apices of a regular tetrahedron in 3-dimensionial space. `subsmat3` has a different structure. FT-employment is different from PT-employment, more different from unemployment but far more different from non-employment. PT-employment, however, is no more closer to unemployment and non-employment than full-time is, while unemployment remains one unit from non-employment.

If you cluster the solutions, and generate groups, the solutions will be broadly similar. However, the order of the clusters may be different so a simple tabulation will be difficult to interpret. Instead use the `permtab` command, which will permute the columns of the table in order to find the best match (highest excess of observed over expected on the diagonal). For example, if `a8` and `b8` are your cluster group variables, `permtab a8 b8` will tabulate them, and report κ_{max}, an indicator of agreement (Reilly et al., 2005)).

You can use the information from `permtab` to examine the off-diagonal cases and attempt to understand why they cluster separately under different substitution cost regimes, for example as follows:

```
sort a8 b8 stripe
list a8 b8 stripe if b8==2,clean
```

3.3 Methods for comparison

The `permtab` command and κ_{max} represent one way of comparing different analyses of the same sequences. Comparison of multidimensional scaling analyses is also very informative, particularly if one can find ways of interpreting the dimensions. Another direct method for comparison is to look at the correlation between the distances for two analyses. To get the distances into a form where we can calculate correlations and create scatterplots, we need to use Mata. In the following example, OM distances are calculated for two differrent substitution matrices, one implying a uni-linear distance structure, and the other an equal distance between each state. The `drop _all` is needed because we want to create a new data set but retain the distance matrices.

```
#delimit ;
use birthseq;

set matsize 700;
matrix subsmat1 = (0,1,2,3 \
                   1,0,1,2 \
                   2,1,0,1 \
                   3,2,1,0 );
matrix subsmat2 = (0,1,1,1 \
                   1,0,1,1 \
                   1,1,0,1 \
                   1,1,1,0 );


oma state7-state42, subsmat(subsmat1) pwdist(rom1) indel(1.5) length(36);
oma state7-state42, subsmat(subsmat2) pwdist(rom2) indel(0.5) length(36);

drop _all;
```

The following Mata code reads the two Stata matrices into the Mata matrices `D1` and `D2`, and then translates them into column vectors using the `vech()` function. This takes one triangle of a symmetric matrix and maps it onto a single column.

```
mata:;
D1 = st_matrix("rom1");
D2 = st_matrix("rom2");
V1 = vech(D1);
V2 = vech(D2);
st_addobs(length(V1));
st_addvar("double","V1");
st_addvar("double","V2");
st_view(V=.,.,.);
V[.,.]=V1,V2;
end
```

The remaining Mata commands create a new data set and map the column vectors into it. We can then calculate Pearson and Spearman correlations, and examine the scatterplot:

```
corr V1 V2;
spearman V1 V2;
scatter V1 V2,msize(0.1);
```

4 Sunday morning

4.1 Degenne's method

Degenne et al. (1996) suggested – but did not implement – a method focusing on comparing vectors of cumulated duration. Sequence similarity is defined as a function of the vectors of cumulated duration in each state, measured at each time point. In geometric terms, Degenne's suggestion is to measure the angle between the vectors at each time point (*i.e.*, in N-dimensional space where the dimensions are the cumulated duration in each of N states, the angle defined by the location of one individual at time t, the origin, and the location of the other). While two sequences with the same cumulated duration at the end will necessarily end up at the same point, their progress through this space will be different if the order in which they do things is not the same.

$$D_{ij} = \sum_t \cos^{-1}(\mathbf{X}_{it}, \mathbf{X}_{jt})$$

where $\mathbf{X}_{ti}$ is the vector of cumulated duration, at time t, for person i.

In practice this seems to give more weight to earlier observations, so I also present a version which weights later observations higher:

$$D^w_{ij} = \sum_t 1.215^i \cos^{-1}(\mathbf{X}_{it}, \mathbf{X}_{jt})$$

This weight is derived by trial and error, and has not been tested widely, so I present it only as an illustration.

It is also possible to calculate the Euclidean distance between the vector elements, rather than the angle. This will perhaps privilege later elements, in that different careers will diverge more as time goes on.

These methods are available in `degenne.ado`, programmed in Mata. To fit them use syntax like this:

```
set matsize 700;

degenne state1-state73, pwdist(resdeg1) length(73)
                        nstates(4) degtype("plain");
degenne state1-state73, pwdist(resdeg2) length(73)
                        nstates(4) degtype("w");
degenne state1-state73, pwdist(resdeg3) length(73)
                        nstates(4) degtype("e");
```

Compare the results with OM results (*e.g.*, generate clusters or do MDS and compare).

4.2 Duration-sensitive OM

Standard OM treats sequences as strings of tokens. Duration in spells is present in the form of repeated tokens, but no allowance is made for the fact that, sociologically, deleting a month from a long spell is a much smaller change than

deleting all of a one-month spell. I present a variation on OM which takes account of the presence of runs of the same token in a sequence when calculating indel and substitution costs. The cost of deleting an element from a run (an indel is necessarily a deletion in one sequence, and a substitution can be considered a deletion–insertion pair) is reduced by a factor of $\frac{1}{\sqrt{l}}$ where l is the length of the run. Thus to delete one of a 2-element run, the cost is $\frac{1}{\sqrt{2}} = 0.71$ times the normal cost, for a 3-element run, $\frac{1}{\sqrt{3}} = 0.58$, for a 10-element run, 0.31 and so on.

```
omav a-e, subs(subsmat) indel(2)
          pwdist(distmat2)
          length(len)
          workspace(yes)
          facexp(0.5);
```

The one new option is `facexp(0.5)`, which allows varying the adjustment factor from the default square root. A value of zero reduces the method to standard OM ($\frac{1}{x^0} = 1.0$) and a value of 1 means the total cost of deleting a whole run is the same as deleting one element in OM. `facexp(0.5)` gives the square root.

Fit OM and variant-OM to the `omav.dat` data, examining the work-space matrices to see how they differ when runs are present.

4.3 Elzinga's combinatorial spell-wise method

Elzinga has proposed a range of "combinatorial" distance measures (Elzinga, 2003, 2005). They focus on ordered sub-sequences (*e.g.*, where "a, c, e" is a subsequence of "a, b, c, d, e") and have the sociologically intuitive interpretation that sequences are more similar the more they have the same elements in the same order. The key notion in his measures is to count the number of subsequences (of lengths from 1 up to the whole of the shorter sequence) that both sequences contain. These methods can be computationally intensive for longer sequences, but he has written an efficient implementation in his CHESA software. Also, for spell data, he proposes a method (sometimes referred to as the X,T method) which counts commonalities in sequences of spells, weighting by their length. This can be quite fast, since even relatively long histories usually contain relatively small numbers of spells.

A version of his spell-oriented method is available as a Stata plugin. This requires data in the format, $s_1, l_1, s_2, l_3, \ldots s_n l_n$ where s_i and l_i are respectively the state in spell i and the length of spell i, where n is the maximum number of spells observed. The actual number of spells for each must also be known, and given as an option.

The following example illustrates its use.

```
input id length m1 l1 m2 l2 m3 l3 m4 l4
1 3   1  4   2  4   3  2  -1 -1
2 2   1  4   2  4  -1 -1  -1 -1
3 3   1 14   2  4   3  4  -1 -1
4 3   1 13   2  4   3  4  -1 -1
5 3   1 12   2  4   3  4  -1 -1
6 3   1 12   2  4   3  4  -1 -1
```

```
7 4   5 12   5  4   5  5   5  5
end

combin m1-l4, pwdist(restuple) length(len) nstates(5)

matrix list restuple
```

Note first that this method generates similarities on a 0–1 scale. Identical sequences have a similarity of 1, and sequences with no elements in common have a similarity of 0. Cluster analysis usually requires dissimilarities so we have to convert these values before proceding:

```
matrix diftuple = J(_N,_N,1) - restuple
clustermat wards diftuple, add
```

`J(_N,_N,1)` creates a square matrix (dimension being the number of cases) full of ones, and the operation reverses the values (1 becomes 0, 0 becomes 1, *etc.*).

Note: see `spellbirthseq.do` for Stata code to translate `birthseq.dta` into a format suitable for Elzinga's X,T method.

4.4 Summarising clusters graphically

Once a cluster solution has been settled on, how to present it graphically? Two commonly used graphical devices are to represent the sequences as horizontal lines on graphs where time is on the x-axis and the states are represented by colours. For relatively small numbers of sequences, individual cases can be distinguished easily, and for larger numbers of cases the structure within clusters is clearly visible, if individuals are not. Let us call this a "sequence vector graph". The other commonly used device is a graph of the distribution of states across time, by cluster group. We could call these "state distribution graphs".

Both types of graph require a certain amount of complex data manipulation.

4.4.1 Sequence vector graphs

The idea of a sequence vector graph is: first, to represent each sequence as a horizontal line (vertical is possible too) where the x-axis represents sequence time and the line's colour the state at each timepoint; and second, to present these lines vertically above each other in an appropriate way. The usual meaning of appropriate here is to sort the lines by their cluster order, and to place them ideally exactly a line's thickness above each other (unless there are really quite few lines, having white lines between sequences makes the graph hard to read). It is additionally possible to insert gaps at chosen locations, for instance to break the graph into a specific set of clusters.

The first task is to rearrange the data into a structure appropriate for vector graphs. Vector graphs draw lines from (x_1, y_1) to (x_2, y_2). If $y_1 = y_2$ the line will be horizontal. We can thus represent a sequence as a set of vectors, first by representing it as a set of spells, $(t_0, t_1, x_1), (t_1, t_2, x_2), \ldots (t_{n-1}, t_n, x_n)$ where t_{i-1} and t_i are start and end time, and x_i is the state in spell i. The corresponding vectors would look like $[(t_0, y), (t_1, y)], [(t_1, y), (t_2, y)]$ and so on.

If we use the `birthseq` data as an example, where `g8` and `g99` are groupings from a cluster analysis, we would proceed as follows, first `reshape`ing

the wide `state1-state73` format into a long format, with 73 observations per individual.

```
reshape long state, i(id) j(t);
```

We then proceed to examine this "long calendar" format to determine its "spell" structure, that is, where consecutive series of observations are in the same state. We number such spells consecutively from 1 within individuals, determine their start-time, end-time and length and then drop all but one record per spell.

```
gen spellno=1;
by id: replace spellno=spellno[_n-1]+(state~=state[_n-1]) if _n>1;
sort id spellno t;
by id spellno: gen length = _N;
by id spellno: gen x = t[1];
by id spellno: gen x2 = t[_N]+1;
by id spellno: gen mark = _n==1;
keep if mark;
drop mark;
```

At this point the data set contains a variable number of records per individual, with start and end-time and state information.

We now sort by the nested cluster structure, and by id and time within. We can now calculate a y value for each individual, where each individual is a fixed quantity (6 in this example) above the previous, and each cluster is additionally higher (by 20 in the example).

```
sort g8 g99 id t;
gen y=1;
replace y = y[_n-1]+6*(id!=id[_n-1]) if _n>1;
replace y = y + (g8-1)*20;
gen y2=y;
```

We can now draw the graph, using Stata's `pcspike` vector-format graph. To get separate colours for the states, we superimpose four graphs, one for each state. Since the linewidth option needs to be the same in each of the four, and needs to be chosen by trial and error, it is easiest to put it in a local macro.

```
local lw = 0.2;
twoway pcspike y x y2 x2 if state==1,
         lwidth(*'lw') legend(label(1 "FTE"))
         ylabel("") ytitle("") xtitle("Months")||
       pcspike y x y2 x2 if state==2, lwidth(*'lw')
         legend(label(2 "PTE")) ||
       pcspike y x y2 x2 if state==3, lwidth(*'lw')
         legend(label(3 "UE")) ||
       pcspike y x y2 x2 if state==4, scheme(s2color)
         lwidth(*'lw') legend(label(4 "NonE"));
```

That will plot the entire set of clusters. We can also plot individual clusters as follows:

```
local clnum 1;
```

```
local lw;
twoway pcspike y x y2 x2 if state==1 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(1 "FTE")) ylabel("")
         ytitle("") xtitle("Months")||
       pcspike y x y2 x2 if state==2 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(2 "PTE")) ||
       pcspike y x y2 x2 if state==3 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(3 "UE")) ||
       pcspike y x y2 x2 if state==4 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(4 "NonE")) scheme(s2color);
```

A tip Given that this can be repetitive, it helps to wrap the graph code in a program definition:

```
capture program drop clusgr;
program define clusgr;
args clnum lw;
twoway pcspike y x y2 x2 if state==1 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(1 "FTE")) ylabel("")
         ytitle("") xtitle("Months")||
       pcspike y x y2 x2 if state==2 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(2 "PTE")) ||
       pcspike y x y2 x2 if state==3 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(3 "UE")) ||
       pcspike y x y2 x2 if state==4 & g8==‘clnum’,
         lwidth(*‘lw’) legend(label(4 "NonE")) scheme(s2color) ;
end;

clusgr 1 1.0;
clusgr 1 0.5;
clusgr 1 0.65;
clusgr 2 2.5;
```

4.4.2 State-distribution graphs

State distribution graphs are perhaps simpler, presenting the distribution across the state space of each cluster at each time-point. It nonetheless requires relatively substantial data manipulation. What we need is a set of time series, showing the numbers in each state at each time point, for each cluster.

We load a data file containing the state information and a series of cluster membership variables, and `reshape` it long, so that observations are person months:

```
use graphbs;
gen id=_n;
keep id g5 state*;
reshape long state, i(id) j(m);
```

We then use the `tab` command to generate four dummy variables, `a1` to `a4`, to represent state. After sorting by cluster and month, we `collapse` by cluster and month, calculating the totals for the four state dummy variables.

```
tab state, gen(a);
sort g5 m;
collapse (sum) a1 a2 a3 a4, by(g5 m);
```

We now have one observation per cluster per month, with four variables indicating totals in the states. To graph these in the same figure we need to cumulate them. In the following example we also label them and write the graph code as a little program for convenience.

```
gen b2=a1+a2;
gen b3=b2+a3;
gen b4=b3+a4;

label variable a1 "FTE";
label variable b2 "PTE";
label variable b3 "UE";
label variable b4 "NonE";


capture program drop tgr;
program define tgr;
args clnum;
/* NB Graph in reverse order of cumulation, to overwrite
   appropriately */
graph  twoway area b4 m if g5==‘clnum’, lw(0.1) yscale(range(0))
            ||area b3 m if g5==‘clnum’, lw(0.1)
            ||area b2 m if g5==‘clnum’, lw(0.1)
            ||area a1 m if g5==‘clnum’, lw(0.1) ;
end;

tgr 1;
tgr 2;
```

4.5 Defining modal sequences

Another approach to summarising clusters is to define typical sequences for each one. Aasave et al. (2007 (forthcoming) propose a good one: the observed sequence closest to the centroid of each cluster. Unfortunately, I haven't implemented a way of doing this in Stata, but propose a simpler summary: the "modal sequence". This is the sequence composed, element by element, of the most common state at that time point. This is conceptually and programmatically simple but is artificial: it is not a real sequence and can indeed represent an impossible sequence. Moreover, transitions within the sequence do not necessarily reflect transitions in the data – for instance the modal sequence may have a transition from never-married to married while the data is showing never-married to cohabiting to married, without cohabiting ever being the modal state. However, it is simple and intuitive.

To generate a modal sequence, we can use Stata's `egen` command, with the `by` prefix to operate on the separate cluster groups. The code `by g8: egen t1 = mode(state1)` will set `t1` to the modal state at time 1. By wrapping it in a

loop we can do it for all 73 months:

```
#delimit ;
use graphbs;

sort g8;
forvalues x=1/73 {;
by g8: egen t‘x’ = mode(state‘x’);
};
```

We can generate a string representation of the sequence as follows. Note the special treatment of `tx==.`: if the mode can't be determined (*e.g.*, two equal largest categories) `egen` gives a missing value, and we insert this in the string as a blank.

```
gen str73 typ="";
foreach x of varlist t1-t73 {;
qui replace typ = typ + "F" if ‘x’==1;
qui replace typ = typ + "p" if ‘x’==2;
qui replace typ = typ + "U" if ‘x’==3;
qui replace typ = typ + "n" if ‘x’==4;
qui replace typ = typ + " " if ‘x’==.;
};
```

Finally, to display one sequence per cluster:

```
by g8: gen clfirst = _n==1;
list g8 typ if clfirst, clean;
```

5 Sunday afternoon

5.1 Comparing clustering algorithms

Cluster analysis is a problematic facet of sequence analysis. CA will give a solution even if there is no cluster structure in the data. That is not necessarily a show-stopper, as it can still be a useful means of partitioning complex data. We are not engaged in molecular biology, however, where the patterns of similarity between sequences map directly onto processes that are naturally represented in a tree structure. We are instead attempting to develop a data driven typology. In this, it is important to understand what we are doing.

There are numerous methods of cluster analysis. Even if we restrict ourselves to SAHN methods (Rohwer and Pötter, 2005, section 7.5.1) – sequential, agglomerative, hierarchical, non-overlapping – there are many methods. These have in common the notion of starting with a partitioning of the data where each case forms a cluster, looking for the pair of clusters that are "closest" to each other (by a method to be specified) and joining them to form a bigger cluster. This process iterates, until the number of partitions is reduced to a target number, or to one which contains the whole data set. It is thus sequential, and aggolmerative, and generates a tree structure composed of mutually exclusive clusters. What varies between SAHN methods is the method used to define inter-cluster distance, which is to say the definition of distance, and the how this maps to distance between clusters as distince from distance between pairs of cases.

Stata provides the following hierarchical agglomerative methods:

- Single linkage: `clustermat singlelinkage`
- Complete linkage: `clustermat completelinkage`
- Average linkage: `clustermat averagelinkage`
- Weighted average linkage: `clustermat waveragelinkage`
- Median linkage: `clustermat medianlinkage`
- Centroid linkage: `clustermat centroidlinkage`
- Ward's linkage: `clustermat wardslinkage`

They have different characteristics and will form different sorts of clusters. Single linkage, for instance, defines inter-cluster distance as the smallest distance between a case in one cluster and a case in the other. This is very useful where clusters may have irregular shapes but are relatively separate from each other. However, it can "chain" clusters together, such that very different cases can end up in the same group. Complete linkage, on the other hand, defines inter-cluster distance as the *largest* distance between pairs of cases. The other methods take more of the pairs into account: average linkage calculates inter-cluster distance as the average distance between all possible pairs formed between cases in one cluster and the other. Weighted average weights this calculation to avoid large clusters dominating smaller ones. (See section 4.2 of Everitt et al. (2001) for more information.)

Wards' method also uses information from all elements of the clusters. The criterion for choosing which pair of clusters to amalgamate, is to minimise the

increase in the intra-cluster sum of squares, taking account of cluster size.

5.2 Comparing methods

A number of different methods are available to you (Hamming, Degenne (angle, weighted, Euclidean), OM, duration-sensitive OM, and Elzinga's XT method). Moreover, some of these have variable cost structures. You also have a number of ways of comparing the results of different algorithms: cluster groupings, dendrograms, correlation, multidimensional scaling. As an exercise, pick a small number of algorithms and/or cost structures, and use a small number of methods for comparing their results. The question to answer is how much, and in what ways, does the algorithm (and its weighting) affect the substantive results achieved.

5.3 Multiple domain analysis

Individual sequences, even in simple state spaces, are complex enough that SA can provide an overview not otherwise available. If we wish to consider multiple state spaces, for example labour market combined with fertility, the complexity increases. SA offers particular advantages for following multiple domain life-course processes.

The simplest strategy to use is to generate a new state space variable as the crosstabulation of the original variables. Thus we might combine an employed–not-employed variable with partnered–single to create the following combined state space:

- employed–single
- employed–partnered
- not employed–single
- not employed–partnered

We can then proceed with SA exactly as before. In reality, we are likely to end up with a combined state space with many more states in it than this example, but this not a difficulty in itself. Where the difficulty arises is, for those algorithms that have substitution cost matrices, how to define the joint cost structure.

5.4 Analysing, and analysing with, SA-derived typologies and dimensions

What follows are some notes on analysis with, and of, the results of sequence analysis. Time does not permit working through examples in the PC-lab.

The goal of sequence analysis is usually to inform more conventional analysis, bringing in "holistic" information from longitudinal data. The typical analysis will consist in using the empirical typology as an explanatory variable in models predicting some outcome which takes place after the trajectories have completed. For instance, one might use include a classification of the first five years of labour market experience in a model predicting wage at age 30. In this case, the classification will enter the model as a set of dummy variables, alongside conventional variables. It is also possible to consider the trajectories as outcomes, however, and predict the classification, using models such as multinomial logistic regression, and variables measured before the start of the trajectory. To continue with the early labour market example, we may wish to

understand why some people have difficult, chaotic periods of insertion in the world of work, and others have relatively easy paths.

A third form of analysis is possible, and this is to use the sequence information itself to predict the empirical typology. This may serve as a way of understanding what drives the typology. The "true" nature of the clusters is not really available to us, as they are created by a complex computational process, but we may be able to derive systematic statements about the classification by a combination of inspection and modelling. "Explanatory" variables in this context can include the individual state variables, and summaries thereof, such as the pattern of cumulated duration in the various states, the frequence of transition between states, and so on.

Parallel with analyses exploiting the empirical typology as a dependent or independent variable we can also consider using the dimensions extracted by MDS, particularly if these have ready interpretations. Use of the dimensions of the space implied by the pairwise distances can avoid some of the problems arising from cluster analysis, where for some sequences the cluster of which they are a member is relatively unstable – small changes in the parameterisation could put them in different clusters. In this respect, if the cluster analysis is considered in some sense as recovering a true "latent" classification, the result actually achieved is subject to measurement error. Use of the dimensions will avoid cluster-derived measurement error, though it will still be subject to errors arising earlier in the process. Furthermore, to the extent that there is a natural cluster structure, the cluster grouping might be more informative.

Bibliography

Aasave, A., Billari, F. and Piccarreta, R. (2007 (forthcoming)) Strings of adulthood: Analyzing work-family trajectories using sequence analysis, *European Journal of Population*.

Abbott, A. (1983) Sequences of social events: Concepts and methods for the analysis of order in social processes, *Historical Methods*, **16**(4), 129–147.

Abbott, A. (1988) Transcending general linear reality, *Sociological Theory*, **6**, 169–186.

Abbott, A. (1990) Conceptions of time and events in social science methods, *Historical Methods*, **23**(4), 140–150.

Abbott, A. (1991) History and sociology: the lost synthesis, *Social Science History*, **15**(2), 201–238.

Abbott, A. (1992) From causes to events: Notes on narrative positivism, *Sociological Methods and Research*, **20**(4), 428–455.

Abbott, A. (1995a) A comment on "Measuring the agreement between sequences", *Sociological Methods and Research*, **24**(2), 232–243.

Abbott, A. (1995b) Sequence analysis: New methods for old ideas, *Annual Review of Sociology*, **21**, 93–113.

Abbott, A. (2000) Reply to Levine and Wu, *Sociological Methods and Research*, **29**(1), 65–76.

Abbott, A. and Forrest, J. (1986) Optimal matching methods for historical sequences, *Journal of Interdisciplinary History*, **XVI**(3), 471–494.

Abbott, A. and Hrycak, A. (1990) Measuring resemblance in sequence data: An optimal matching analysis of musicians' careers, *American Journal of Sociology*, **96**(1), 144–85.

Abbott, A. and Tsay, A. (2000) Sequence analysis and optional matching methods in sociology, *Sociological Methods and Research*, **29**(1), 3–33.

Anyadike-Danes, M. and McVicar, D. (2005) You'll never walk alone: Childhood influences and male career path clusters, *Labour Economics*, **12**(4), 511–530.

Blair-Loy, M. (1999) Career patterns of executive women in finance: An optimal matching analysis, *American Journal of Sociology*, **104**(5), 1346–1397.

Bradley, D. W. and Bradley, R. A. (1983) Application of sequence comparison to the study of bird songs, in Sankoff and Kruskal (1983), chapter 6.

Brüderl, J. and Scherer, S. (2004) Methoden zur analyse von sequenzdata, *Kölner Zeitschrift für Soziologie und Sozialpsychologie*, **44**, 330–347.

Buchmann, M. and Sacchi, S. (1995) Mehrdimensionale Klassifikation beruflicher Verlaufsdaten: Eine Anwendung auf Berufslaufbahnen zweier Schweizer Geburtskohorten, *Kölner Zeitschrift für Soziologie und Sozialpsychologie*, **47**(3), 413–442.

Chan, T. W. (1995) Optimal Matching Analysis: A methodological note on studying career mobility, *Work and Occupations*, **22**, 467–490.

Clark, W. A. V., Deurloo, M. C. and Dieleman, F. (2003) Housing careers in the United States, 1968-93: Modelling the sequencing of housing states, *Urban*

Studies, **40**(1), 143–160.

Degenne, A., Lebeaux, M.-O. and Mounier, L. (1996) Typologies d'itinéraires comme instrument d'analyse du marché du travail. Troisièmes journées d'études Céreq-Cérétim-Lasmas IdL, Rennes, 23–24 May 1996.

Dijkstra, W. and Taris, T. (1995) Measuring the agreement between sequences, *Sociological Methods and Research*, **24**(2), 214–231.

Elzinga, C. H. (2003) Sequence similarity: A non-aligning technique, *Sociological Methods and Research*, **32**(1), 3–29.

Elzinga, C. H. (2005) Combinatorial representations of token sequences, *Journal of Classification*, **22**(1), 87–118.

Everitt, B. S., Landau, S. and Leese, M. (2001) *Cluster Analysis*, 4th edn. London, Arnold.

Halpin, B. (2003) Tracks through time and continuous processes: Transitions, sequences, and social structure, *Working paper 2003-01*, Dept of Sociology, University of Limerick.

Halpin, B. and Chan, T. W. (1998) Class careers as sequences: An optimal matching analysis of work-life histories, *European Sociological Review*, **14**(2).

Kruskal, J. B. (1983) An overview of sequence comparison, in Sankoff and Kruskal (1983).

Lesnard, L. (2006) Optimal matching and social sciences, *Document du travail du Centre de Recherche en Économie et Statistique 2006-01*, Institut Nationale de la Statistique et des Études Économiques, Paris.

Levine, J. H. (2000) But what have you done for us lately? Commentary on Abbott and Tsay, *Sociological Methods and Research*, **29**(1), 34–40.

Levy, R., Gauthier, J.-A. and Widmer, E. (2006) Entre contraintes institutionnelle et domestique : les parcours de vie masculins et féminins en Suisse, *Canadian Journal of Sociology*, **31**(4), 461–489.

Malo, M. A. and Muñoz-Bullón, F. (2003) Employment status mobility from a life-cycle perspective: A sequence analysis of work-histories in the BHPS, *Demographic Research*, **9**, 119–162.

McVicar, D. and Anyadike-Danes, M. (2002) Predicting successful and unsuccessful transitions from school to work using sequence methods, *Journal of the Royal Statistical Society (Series A)*, **165**, 317–334.

Pollock, G. (2007) Holistic trajectories: A study of combined employment, housing and family careers by using multiple-sequence analysis, *Journal of the Royal Statistical Society: Series A*, **170**(1), 167–183.

Reilly, C., Wang, C. and Rutherford, M. (2005) A rapid method for the comparison of cluster analyses, *Statistica Sinica*, **15**(1), 19–33.

Rohwer, G. and Pötter, U. (2005) *TDA User's Manual*. Universität Bochum.

Sankoff, D. and Kruskal, J. B. (eds) (1983) *Time Warps, String Edits and Macromolecules*. Reading, MA, Addison-Wesley.

Scherer, S. (2001) Early career patterns: A comparison of Great Britain and West Germany, *European Sociological Review*, **17**(2), 119–144.

Stovel, K. and Bolan, M. (2004) Residential trajectories: Using optimal alignment to reveal the structure of residential mobility, *Sociological Methods and Research*, **32**(4), 559–598.

Wang, W. and Zaïane, O. R. (2002) Clustering web sessions by sequence alignment, *Proceedings, 13th International Workshop on Database and Expert Systems Applications*, IEEE, IEEE, 394–398.

Wilson, C. (2006) Reliability of sequence-alignment analysis of social pro-

cesses: Monte carlo tests of ClustalG software, *Environment and Planning A*, **38**(1), 187.

Wu, L. L. (2000) Some comments on "Sequence analysis and optimal matching methods in sociology: Review and prospect", *Sociological Methods and Research*, **29**(1), 41–64.