Sequence Analysis for Social Scientists

Brendan Halpin, Dept of Sociology
University of Limerick

Oslo, June 16-18 2015

Contents

1 Session 1: Introduction

1.1 Installing the software

We will work with Stata. This requires initially downloading add-ons. Try:

```
net from http://teaching.sociology.ul.ie/sadi/
net install sadi
```

to begin with. This installs my SADI package. See http://www.ul.ie/sociology/pubs/wp2014-03.pdf for full information on SADI.

As well as SADI we need two other packages:

```
ssc install moremata
ssc install sq
```

This installs some matrix manipulation tools that my software, SADI, uses, and the SQOM package of Kohler et al., which is useful for graphics.

1.2 A simple example

We begin with a simple example. This code creates a small sequence data set:

```
input id s1 s2 s3 s4 s5
1 1 2 3 2 3
2 2 3 4 1 4
3 4 4 4 4 4
4 1 1 2 3 3
5 4 3 2 1 4
end
```

Put the code in the Stata do-file editor and run it. These sequences have 5 observations and 4 states.

1.3 Hamming distance

Hamming distance is the sum of the difference at each time point. It requires sequences to be the same length, and recognises similarity at the same time point. To calculate Hamming distance we need to define the difference between the states. To do this we need a 4x4 matrix:

```
matrix subs = (0,1,2,3\ ///
               1,0,1,2\ ///
               2,1,0,1\ ///
               3,2,1,0)
matrix list subs
```

This particular example sets the "substitution cost" to the simple difference between the values.

To make the calculation, issue the following command:

```
hamming s1-s5, subsmat(subs) pwd(ham)
matlist ham
```

The distances are written to the `ham` matrix, which has the same order as the data set. For one or two pairs of sequences, verify that the Hamming distance is what you would expect.

1.4 Optimal Matching distance

The OMA distance is similar to Hamming distance, except that it also allows insertion and deletion, and can thus recognise similarities that are out of alignment. To do this it needs an `indel()` option to put a cost on insertion and deletion, and a `length()` option because it can work with sequences of different length. If the indel cost is less than half any substitution cost, those substitutions will not be used, so in general use, the minimum indel cost is half the largest substitution cost.

```
oma s1-s5, subsmat(subs) pwd(oma) len(5) indel(1.5)
matlist oma
```

Compare the `ham` and `oma` distance matrices: what do you notice about them?

1.5 OMA's workspace

To understand how the OM algorithm works, the `workspace` option will print the internal work matrices for each pair of sequences (use only with very small datasets!).

```
oma s1-s5 if inlist(id,1,5), subsmat(subs) pwd(oma) len(5) indel(1.5) workspace
```

Sequence 1 is represented by the rows, sequence 2 by the columns. The margins of the table count the cost of repeated insertions or deletions. Beginning in the top left of the table, each cell is filled as the minimum of the one above it plus the indel cost (delete from seq 1), the one to its left plus the indel cost (delete from seq 2), and the one above-left plus the substitution cost. Note that a deletion from one sequence is equivalent to an insertion in the other.

The bottom right element of the workspace is the OM distance (before it is normalised by dividing by the length of the longer sequence). Working backwards allows us to determine what each operation was (insertion/deletion/substitution), to identify the optimal alignment or alignments.

```
Seq 1:     1   2   3   2   3
Seq 2:     4   3   2   1   4

Substitution costs:
           4     3     2     1     4
```

```
      1   3.00   2.00   1.00   0.00   3.00
      2   2.00   1.00   0.00   1.00   2.00
      3   1.00   0.00   1.00   2.00   1.00
      2   2.00   1.00   0.00   1.00   2.00
      3   1.00   0.00   1.00   2.00   1.00

Working space:
  0.00 |   1.50   3.00   4.50   6.00   7.50
  -----+-----------------------------------
  1.50 |   3.00   3.50   4.00   4.50   6.00
  3.00 |   3.50   4.00   3.50   5.00   6.50
  4.50 |   4.00   3.50   5.00   5.50   6.00
  6.00 |   5.50   5.00   3.50   5.00   6.50
  7.50 |   7.00   5.50   5.00   5.50   6.00
```

In this example the result is 6 (or 1.2 when normalised, less than the Hamming distance of 1.4). Working back, the last operation was substitution (6 = 5 + 1, comparing 3 and 4), so we are now one cell up and one to the left, comparing 2 and 1. The cheapest way to arrive here is from the cell to the left (3.5 + 1.5 = 5, delete 1), because this allows us to backtrack through two perfect matches (2 vs 2, and 3 vs 3). At this point we are comparing 2 and 4 (row 2, col 1 in the body of the table) where a substitution cost of 2 puts us in the margin of the table, and we delete 1 to arrive at the start.

Repeat this exercise for sequence pairs (1,3) and (2,4). Note the the (1,3) distance is the same as Hamming, but the (2,4) is not.

Repeat the exercise for pair (1,5) but increase the indelcost to 2. What happens?

1.6 Differential length

The optimal matching algorithm is designed to deal with sequences of differing lengths.

```
clear
input id s1 s2 s3 s4 s5 seqlen
1 1 2 3 2 3 5
2 2 3 4 1 4 5
3 4 4 4 . . 3
4 1 1 2 3 . 4
5 4 3 2 1 4 5
end

oma s1-s5, subsmat(subs) pwd(om2) len(seqlen) indel(1.5)

matlist om2
```

1.7 Further experimentation

If you have time, experiment with other sequences, by editing the `input` section of the code. Experiment with other substitution matrices, for instance the following:

```
matrix extreme = (0,1,2,5\ ///
                  1,0,1,2\ ///
                  2,1,0,1\ ///
                  5,2,1,0)
```

which treats category 4 as extra different (note you will need to change the `indel()` option too).

2 Session 2: Working with real data

2.1 Working with real data

We will work with two main example data sets, a six-year span of labour market activity of women who have a birth at the end of year two (drawn from the BHPS) and the McVicar/Anayadike-Danes school leaver data set. We will work with the mothers' labour market sequences first:

```
use http://teaching.sociology.ul.ie/oslo/bsseq
```

The labour market status variable has the values full-time, part-time, unemployed and non-employed, but we have 72 consecutive observations. How do we get an overview?

```
tab state1 state72
tab state1 state24
tab state24 state72
```

Better to make a chronogram or time-dependent state-distribution graph:

```
chronogram state*
```

This loses all individual information but summarises the temporal average well. To retain the individual level view, we need to use indexplots. SQOM is useful here, but requires the data in a different format (long, with one observation per person–time-unit):

```
reshape long state, i(pid) j(t)
sqset state pid t
sqindexplot, order(pid)
```

This plot orders the data by PID, which is pretty much at random with respect to the sequences. As a result, it is hard to see much structure. The default `sqindexplot` is to sort "lexically", by month 1, then by month 2, etc.

```
sqindexplot
```

This is a little better, and in particular we see that there are significant numbers of sequences 100% in the same state. However, for those that change state, after the first transition there is the same chaos as before. You could also sort by another transition, e.g., the date of the first month in work after the birth. To calculate this variable we move back to the wide format temporarily.

```
reshape wide
gen retdate = 99
forvalues x = 25/72 {
  replace retdate = `x' if inlist(state`x',1,2) & retdate==99
}
reshape long
sqindexplot, order(retdate)
```

We can use one or two other SQOM utilities to get more information on the sequences:

```
sqtab
sqdes
```

`sqtab` tabulates the sequences (showing a truncated representation), which makes it apparent (with numbers) that we have lots of simple sequences, with 27.45% being non-employed the whole time. `sqdes` shows that there are 417 (of 940) unique patterns. For 392 patterns there is only a single example, but the 100% non-employed pattern has 258 examples.

2.2 Survival curves

Indexplots and chronograms take advantage of the whole time-span but it can be useful to focus on single transitions in a more traditional framework such as Kaplan-Meier survival curves. Let's look at the return to work again, defined as the first month full or part-time after the birth (for some it is month 1, for some never).

```
reshape wide
gen returned = retdate<99
stset retdate, failure(returned)
sts graph
```

2.3 School-leavers data

Download the MVAD data. Note that this has six states (school, training, further education, higher education, unemployment, employment). Use the tools described above to get an overview of it. Note that this dataset has a few useful covariates, so try adding the `by` option to the chronogram, indexplot and survival graphs (e.g., `by(male)` or `by(gcse5eq)`).

2.4 OM and Hamming with real data

Calculating OM and Hamming distances works the same way as with the small data set above. The only difference is setting the substitution and indel costs. For the mothers data, the same substitution matrix can be used:

```
matrix subs = (0,1,2,3\ ///
               1,0,1,2\ ///
               2,1,0,1\ ///
               3,2,1,0)
matrix list subs
```

This puts the four states on a single dimension from full-time work to non-employment. For the MVAD data, you could use the cost matrix they used in their paper:

```
matrix mvdanes = (0,1,1,2,1,3 \ ///
                  1,0,1,2,1,3 \ ///
                  1,1,0,2,1,2 \ ///
                  2,2,2,0,1,1 \ ///
                  1,1,1,1,0,2 \ ///
                  3,3,2,1,2,0 )
```

For either or both data sets, calculate OM (set indel to 1.5) and Hamming distances. You may need to increase the maximum matrix size first (the default is 800 and there are 940 mothers' sequences):

```
clear
use http://teaching.sociology.ul.ie/oslo/mvad
matrix mvdanes = (0,1,1,2,1,3 \ ///
                  1,0,1,2,1,3 \ ///
                  1,1,0,2,1,2 \ ///
                  2,2,2,0,1,1 \ ///
                  1,1,1,1,0,2 \ ///
                  3,3,2,1,2,0 )
hamming state1-state72, subsmat(mvdanes) pwd(ham)
oma state1-state72, subsmat(mvdanes) indel(2) pwd(oma) len(72)
```

Note that the `hamming` command is relatively slow as it is coded in Mata, rather than C. The `oma` command has to make rather more calculations so it is necessary to code this in C for speed. However both commands are making 440,391 sequence comparisons, so neither is instantaneous.

You can compare the results by inspecting parts of the distance matrices like this:

```
matlist ham[1..10,1..10]
matlist oma[1..10,1..10]
```

Note that in many cases the distances are the same. You can compare pairs visually using a string representation of the sequence:

```
stripe state*, gen(seqstr) symb("EFHSTU")
list seqstr if inlist(_n,1,10)
list seqstr if inlist(_n,6,10)
```

2.5 Clustering

When you have large distance matrices, simple inspection is not effective. We need to reduce the complexity of the data in some manner. The usual one is cluster analysis: group sequences with similar sequences in order to create a data-driven classification. This is readily carried out in Stata. Given distances in the matrix `oma`:

```
clustermat wards oma, add
cluster generate g8=groups(8)
```

```
tab g8
sort g8 seqstr
list g8 seqstr
chronogram state*, by(g8)
```

We can also create cluster-specific index plots:

```
cluster generate g999 = groups(800), ties(fewer)
reshape long state, i(id) j(t)
sqset state id t
sqindexplot, by(g8)            // by cluster, lexical order within
sqindexplot, order(g999)       // by dendrogram order
sqindexplot, by(g8) order(g999) // by cluster, sub-cluster order within
```

It can be useful to view the dendrogram, but for more than about 40 sequences it needs to be "trimmed" using the `cutnumber()` or `cutvalue()` options:

```
reshape wide
cluster dendrogram, cutnumber(40)
cluster dendrogram, cutvalue(15)
```

2.6 Comparing distance matrices

We can run the same analysis on OMA distances and on Hamming distances. The results will not be very different but will not be the same. First, we can compare the distances against each other, for instance by correlation:

```
corrsqm ham oma
```

As we see, the correlation is very high, partly because for many pairs the distances are the same. But if we cluster the Hamming distance and compare it with the OM cluster solution, small differences are amplified a little:

```
clustermat wards ham, add
cluster gen h8=groups(8)
permtab g8 h8
```

`permtab` permutes the second classification to maximise its match with the first, and then tabulates the result. As we see, a substantial minority of sequences are clustered differently.

2.7 Other linkages

Ward's "linkage" (i.e., method for deciding what elements to combine into clusters) is often used in cluster analysis, and it has a nice interpretation in terms of minimising intra-cluster variance. There are a number of other linkages in hierarchical agglomerative cluster analysis, as well as non-hierarchical cluster methods to consider. While non-hierarchical methods such as k-means and k-medians are not available for distance matrices in Stata, it is easy to replace Ward's with other methods. The choices available are:

- `singlelinkage` single-linkage cluster analysis

- `averagelinkage` average-linkage cluster analysis
- `completelinkage` complete-linkage cluster analysis
- `waveragelinkage` weighted-average linkage cluster analysis
- `medianlinkage` median-linkage cluster analysis
- `centroidlinkage` centroid-linkage cluster analysis
- `wardslinkage` Ward's linkage cluster analysis

Try `clustermat median oma, add` and so on, using e.g., `permtab` to compare solutions.

In my experience, Ward's is the most likely to yield a fairly even distribution of cases across clusters.

2.8 String representations and regular expressions

We have seen the utility of representing sequences as strings:

```
permtab g8 h8, newvar(ph8)
list seqstr if g8==1 & ph8==7 // View cases in disagreement
list seqstr if g8==1 & ph8!=7
```

"Regular expressions" are ways of defining patterns in strings. Regexes are implemented in many languages, including Stata. These can be very useful for exploring the structure of sequences.

- `A*B`: `A` zero or more times followed by a `B`
- `A+B`: at least one `A` followed by a `B`
- `A?B`: `A` zero or one time 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$`: a string ending with `ABC`
- `^A+$`: a string composed entirely of {`A`}s
- `A[BCD]+E`: `A` followed by at least one of `B`, `C` or `D`, followed by `E`
- `A[^A]+A`: `A` followed by at least one element that is not `A`, followed by `A`

For instance, to ask how many sequences are 100% in non-employment:

```
clear
use http://teaching.sociology.ul.ie/oslo/bsseq
stripe state*, gen(seqstr) symb("FPun")
count if regexm(seqstr,"^n+$")
```

How many sequences do we observe entering unemployment from full-time:

```
count if regexm(seqstr,"Fu")
```

How many sequences do we observe entering unemployment from full-time, but later return to full or part-time:

```
count if regexm(seqstr,"Fu.+[FP]")
```

What clusters include sequences with 12 months part-time:

```
  // Generate clusters
oma state1-state72, subsmat(subs) indel(2) pwd(oma) len(72)
clustermat wards oma, add
cluster gen g8=groups(8)
  // Test patterns
gen pt12 = regexm(seqstr,"PPPPPPPPPPPP")
tab g8 pt12
```

What clusters include sequences that are never in non-employment:

```
gen nevern = regexm(seqstr,"^[FPu]+$")
tab g8 nevern
```

Which include sequences that experience at least 12 consecutive months of full-time, at least 6 of non-employment and at least 12 of full-time later:

```
gen fnf = regexm(seqstr,"FFFFFFFFFFFF.*nnnnnn.*FFFFFFFFFFFF")
tab g8 fnf
```

With either data set, explore this method to characterise as many of the clusters as possible. It is a very good way of assessing the coherence of the cluster solution in terms of substantive meaning. With luck and perseverance you may end up defining a set of ideal types that characterise the bulk of the data.

2.9 Other ways of characterising clusters: n-spells, cumdur, entropy, turbulence

The SADI package includes a number of other utilities that can be useful for summarising clusters or individual sequences.

```
cumuldur state*, cd(dur) nstates(4)
table g8, c(mean dur1 mean dur2 mean dur3 mean dur4)

nspells state*, gen(nspells)
table g8, c(mean nspells)

entropy state*, cd(rel) nstates(4) gen(entropy)
table g8, c(mean entropy)
```

Cumulated duration simply adds time-units in each state, creating one summary variable (dur1 to dur4 in this case) per state. nspells treats consecutive runs in the same state as spells (with missing counted as a state, if present). The entropy calculation calculates Shannon entropy, based on cumulated duration. Entropy is calculated as $-\sum p_i \log_2 p_i$ where p_i is the proportion of months in state i. It takes no account of order, only of cumulative duration, so is clearly inferior to Elzinga's *turbulence* measure, which is available in R but not in Stata.

3 Session 3

3.1 Setup: recreate oma matrix and cluster solution

```
use http://teaching.sociology.ul.ie/oslo/bsseq
matrix subs = (0,1,2,3\ ///
               1,0,1,2\ ///
               2,1,0,1\ ///
               3,2,1,0)
oma state1-state72, subsmat(subs) indel(2) pwd(oma) len(72)
clustermat wards oma, add
cluster gen g = groups(8 999), ties(fewer)
```

3.2 Multi-dimensional scaling

The other "obvious" thing to do with a pairwise distance matrix, apart from partitioning into clusters, is to use the information in it to attempt to construct the multi-dimensional space it implies. In the distances are meaningful, it is likely that they describe a space with much fewer dimensions than the number of trajectories. In fact, we often find that two to three dimensions will capture a large part of the structure.

This code does MDS on the `oma` distance matrix, and saves the first three dimensions (coordinate vectors) to variables:

```
set matsize 1000
mdsmat oma, dim(3)
matrix dim = e(Y)
svmat dim
```

The MDS output shows that three dimensions account for between 75% and 99% of the structure (there is ambiguity about whether we should calculate the recovered distances additively or in Euclidean fashion).

We can examine the relationship between the MDS and cluster solutions:

```
separate dim2, by(g8)
scatter dim21-dim28 dim1
separate dim3, by(g8)
scatter dim31-dim38 dim1
```

We can print sequence info on the scatterplot:

```
stripe state*, gen(seqstr)
scatter dim2 dim1, mlabel(seqstr)
scatter dim2 dim1 if inrange(dim1,0,1), mlabel(seqstr)
```

However, that can be hard to read. It can be more convenient in the results window:

```
sort dim1
list dim1 dim2 dim3 seqstr, clean
```

This shows the relationship between the first dimension and the sequences. Note how it starts at 100% A, and ends up in 100% B. Note the locations on `dim1` of the four types of single-spell sequences, and relate that to the substitution costs.

You can do the same sorting by `dim2`, etc, to help interpret the other dimensions.

We can also use indexplots in this context:

```
reshape long state, i(pid) j(t)
sqset state pid t
sqindexplot, order(dim1)
```

Indexplots ordered by the first MDS dimension are used by Piccarreta and Lior (2010).

We can extend on this by partitioning the MDS space mechanically (note that cluster analysis can be thought of as partitioning it algorithmically). This approach cuts each dimension at the mean, creating $8 = 2^3$ partitions:

```
reshape wide
centile dim1
gen d1 = dim1>`r(ub_1)'
centile dim2
gen d2 = dim2>`r(ub_1)'
centile dim3
gen d3 = dim3>`r(ub_1)'
gen dx = d3+2*(d2 + 2*d1)
reshape long
sqindexplot, order(dim1) by(dx, legend(off) cols(4))
```

As we can see, the partitions are very distinct, and this offers an attractive alternative to clustering.

3.3 Optimal matching and costs

A lot of criticism of OM hinges on the lack of understanding of the effect of substitution and indel costs. However, we can experiment and get insight that way.

3.3.1 Indel, OM and Hamming

First, note that if you raise indel costs enough, OM yields Hamming distances:

```
reshape wide
hamming state*, subs(subs) pwd(ham)
oma state*, subs(subs) indel(1.5)  length(72) pwd(om1)
corrsqm ham om1
oma state*, subs(subs) indel(1.75) length(72) pwd(om2)
corrsqm ham om2
oma state*, subs(subs) indel(2.0)  length(72) pwd(om3)
corrsqm ham om3
return list
oma state*, subs(subs) indel(2.5)  length(72) pwd(om4)
// . . .
```

How high do you have to raise *indel* before you get a correlation of exactly 1? (`return list` after `corrsqm` shows you the unrounded result).

3.3.2 Different substitution matrices

The substitution matrix we have been using is very simple – it puts the four states at equal intervals along a single dimension:

```
matrix linear = (0,1,2,3\ ///
                 1,0,1,2\ ///
                 2,1,0,1\ ///
                 3,2,1,0)
```

An equally simple position would be to put all states equally different from each other:

```
matrix linear = (0,1,1,1\ ///
                 1,0,1,1\ ///
                 1,1,0,1\ ///
                 1,1,1,0)
```

Generate and compare cluster solutions for these two distance structures (note that you should set the indel cost relative to the maximum substitution cost). Can you see where the differences come from?

Experiment also with this matrix, which marks non-employment as more distinct in a linear framework:

```
matrix nonemp = (0,1,2,5\ ///
                 1,0,1,2\ ///
                 2,1,0,1\ ///
                 5,2,1,0)
```

Consider also a matrix that puts F, P and U on a single dimension with N equally different from them all:

```
matrix twodim = (0,1,2,2\ ///
                 1,0,1,2\ ///
                 2,1,0,2\ ///
                 2,2,2,0)
```

This approximates two dimensions (see this fact with `mdsmat twodim`)

Note: compare the results using `corrsqm`, and indexplots and permutation tables of the cluster results.

3.3.3 MDS mat on substitution matrices

As we have seen, MDS can be used to make sense of substitution matrices as well as distance matrices. It is a useful way to think about the structure of the "state" state-space, which is projected onto the "trajectory" state-space. See what structure you find in these two substitution cost matrices, taken respectively from Halpin and Chan (1998) and McVicar and Anyadike-Danes (2002):

```
matrix hc = (0, 2, 2, 2, 2, 3, 3\ ///
             2, 0, 1, 1, 1, 2, 2\ ///
             2, 1, 0, 1, 1, 2, 2\ ///
             2, 1, 1, 0, 1, 2, 2\ ///
             2, 1, 1, 1, 0, 2, 2\ ///
             3, 2, 2, 2, 2, 0, 1\ ///
             3, 2, 2, 2, 2, 1, 0)
matrix rownames hc = I-II III IVab IVcd V-VI VIIa VIIb
matrix colnames hc = I-II III IVab IVcd V-VI VIIa VIIb

matrix md = (0, 1, 1, 2, 1, 3\ ///
             1, 0, 1, 2, 1, 3\ ///
             1, 1, 0, 2, 1, 2\ ///
             2, 2, 2, 0, 1, 1\ ///
             1, 1, 1, 1, 0, 2\ ///
             3, 3, 2, 1, 2, 0)
matrix rownames md = E F H S T U
matrix colnames md = E F H S T U

mdsmat hc, dim(3)
mat hcd = e(Y)
svmat hcd
graph matrix hcd*
```

3.4 Other distance measures

OM is the default distance measure in much sequence analysis literature but there are a number of alternatives with different characteristics. This include:

- methods that take account of context (neighbouring states, spell length): Hollister's localised OM and my duration-adjusted OMv (both are non-metric, unfortunately)
- Lesnard's Dynamic Hamming distance
- Time-warping measures such as Marteau's Time-Warp Edit Distance
- Elzinga's combinatorial approaches.

On one or both of the example data sets, fit each of the measures and compare the results, using `corrsqm`, comparing indexplots and `permtab`.

LOM and OMv:

```
hollister state*, subs(subs) time(0.5) local(0.5) pwdist(hol) length(72)
omav state*, subs(subs) indel(1.5) pwdist(omv) length(72)
metricp hol
metricp omv
```

Dynamic Hamming:

```
dynhamming state*, pwdist(dyn)
```

TWED:

```
twed state8, subsmat(subs) lambda(0.5) nu(0.15) pwdist(twd) len(72)
```

Elzinga's duration-weighted spell-focused Number of Matching Subsequences measure (`combinprep` reorganises the data substantially first):

```
combinprep, state(state) length(l) idvar(pid) nsp(nspells)
combinadd state1-l`r(maxspells)', pwsim(xtd) nspells(nspells) nstates(`r(nels)') rtype(d)
```

3.5 Post hoc analysis

Use the MVAD data to explore the association between the sequences and other covariates:

```
use http://teaching.sociology.ul.ie/oslo/mvad, clear

matrix md = (0, 1, 1, 2, 1, 3\ ///
             1, 0, 1, 2, 1, 3\ ///
             1, 1, 0, 2, 1, 2\ ///
             2, 2, 2, 0, 1, 1\ ///
             1, 1, 1, 1, 0, 2\ ///
             3, 3, 2, 1, 2, 0)
matrix rownames md = E F H S T U
matrix colnames md = E F H S T U


oma state*, subs(md) indel(1.5) pwd(oma) length(72)

clustermat wards oma, add
cluster gen g8=groups(8)

// Predict the outcome classification using covariates
mlogit g8 funemp grammar gcse

// Predict the classification using cumulated duration
cumuldur state*, cd(cd) nstates(6)
mlogit g8 cd1-cd5
est store base
mlogit g8  cd1-cd5 grammar
lrtest base

// Does the clustering "explain" the covariates, over and above the
// simpler summary of cumulated duration?
logit gcse cd1-cd5
logit gcse cd1-cd5 i.g8
```

3.6 Discrepancy

Studer et al (2011) describe a way of treating distance matrices in a way analogous to sums of squares in linear models. It is useful for assessing the association of a covariate with the distance structure. For instance, are members of groups within a variable statistically more like each other?

3.6.1 Pseudo-R^2 and pseudo-F tests

With the MVAD data:

```
oma state*, subs(md) indel(1.5) pwd(oma) length(72)
discrepancy funemp, dist(oma) idvar(id)
```

Average distance to the centre of the group when broken down by `funemp` (father unemployed) is less than that to the centre of the whole data set, but not much: it doesn't appear to be statistically significant according to the permutation-test p-value.

```
discrepancy gcse5eq, dist(oma) idvar(id)
```

For `gcse5eq` (5 or more GCSEs) the result is significant. When the permutation-test p-value is in the tail like this, it is worth testing with more permutations, 1000 to 5000 (this is a little slow):

```
discrepancy gcse5eq, dist(oma) idvar(id) niter(1000)
```

If you want to test the effect of multiple variables, you need to create their cross-tabulation:

```
gen cross = gcse5eq+funemp*2
discrepancy cross, dist(oma) idvar(id) niter(1000)
```

3.6.2 Comparing F-tests with cluster solutions

The alternative way to assess association between distances and covariates is via the cluster solution (by tabulating or modelling):

```
tab g8 funemp, chi
tab g8 gcse5eq, chi
tab g8 cross, chi
```

Compare the results with those from the discrepancy pseudo-F test.

3.6.3 Assessing cluster solutions

We can also use discrepancy to assess how homogeneous cluster solutions are, to compare different clusterings of the same data:

```
discrepancy g8, dist(oma) id(id)
clustermat waver oma, add
cluster gen w8=groups(8)
discrepancy w8, dist(oma) id(id)
```

We can see how clusters differ in homogeneity, by saving as a variable the distance to the cluster centre of gravity:

```
discrepancy g8, dist(oma) id(id) dcg(d2c)
```

Use an indexplot to see how the different mean distances relate visually to cluster homogeneity.

3.6.4 Medoids

Finally, we can use discrepancy to identify "medoids", the sequences in each cluster closest to its centre.

```
bysort g8: egen mindist = min(d2c)
gen medoid = d2c == mindist
tab g8 medoid
list g8 stripe if medoid
gsort g8 -medoid
by g8: gen first = _n==1
list g8 medoid stripe if first
```

Note that more than one sequence can be the medoid.

Compare the indexplot and/or chronogram with the medoid. Is it a good summary?

4 Session 4

4.1 MCSA

Use the data in http://teaching.sociology.ul.ie/oslo/mcsa18.dta and the example code in http://teaching.sociology.ul.ie/oslo/mcsa18run.do to set up an MCSA example. The data contains 18 observations of the state at interview on employment status (employed/unemployed/not employed) and vote (4 categories). The code creates a combined variable and a combined substitution cost matrix.

```
do http://teaching.sociology.ul.ie/oslo/mcsa18run.do
```

Do OMA on the `eun` variable, and on the `vote` variable. Combine the distances as follows:

```
mata: st_matrix("additive", st_matrix("deun") :+ st_matrix("dvote"))
```

This simply adds the 1-domain distances together in a new distance matrix, `additive`. Do a cluster analysis on this distance: it takes account of the two dimensions without accounting for their interacting with each other.

Then do OMA on the combined variable, using the combined substitution cost matrix. Do a cluster analysis on this and compare your results. This takes account of the domains' interaction.

```
oma cross1-cross18, subs(mcsa) indel(3) pwd(dist) length(18)
clustermat wards dist, add
cluster generate g = groups(4 8 16 32 500), ties(fewer)
```

4.1.1 Viewing results

The following code will graph the results together

```
reshape long cross vote eun, i(pid) j(t)
sqset eun pid t
sqindexplot, by(g8, legend(off) note("Employment")) order(g500) name(emplot, replace)
sqset vote pid t
sqindexplot, by(g8, legend(off) note("Voting intention")) order(g500) name(votplot, replac
```

4.2 Dyads

See `http://teaching.sociology.ul.ie/dyadoma.do` for a worked example