University of Limerick
Department of Sociology Working Paper Series

Working Paper WP2014-03
April 2014

Brendan Halpin

Department of Sociology, University of Limerick

SADI: Sequence Analysis Tools for Stata

SADI: Sequence Analysis Tools for Stata

Brendan Halpin

3rd April 2014[*]

Contents

1 Introduction **2**
1.1 Referring to SADI 2

2 Installation **2**

3 Data requirements **3**

4 Worked example **3**
4.1 Quick start 3
4.2 Data 3
4.3 Pairwise distances 3
4.3.1 X/t 4
4.3.2 Data-driven substitution matrix 4
4.4 Examining distance matrices 5
4.4.1 Comparing distances 5
4.4.2 The triangle inequality 5
4.5 Cluster analysis 6
4.5.1 Comparing cluster solutions 6
4.5.2 Discrepancy 7
4.6 Summarising sequences and clusters 8
4.6.1 String representations of sequences 8
4.6.2 Medoids: typical sequences 8
4.6.3 Cumulated duration 9
4.6.4 Entropy 9
4.6.5 Number of spells 10
4.7 Graphics 10
4.7.1 Chronogram 10
4.7.2 Indexplot 10
4.7.3 Transition pattern graph 11
4.7.4 `maketrpr` 11

5 Compiling plugins **13**

6 Appendix: Help pages **15**

[*]documentation.org,v 1.2 2014/04/03 15:07:39 brendan Exp

1 Introduction

SADI is a suite of Stata tools for sequence analysis, with a particular focus on holistic comparisons of sequences using measures such as optimal matching distance. It provides a number of distance measures, including

- Optimal matching distance
- Hamming distance
- Dynamic Hamming distance
- Elzinga's combinatorial X/t measure and
- TWED, a time-warping distance measure.

It provides a number of utilities for graphing sequence-related data, for summarising sequences, and for handling sequences in general.

The main alternatives to SADI are the Stata SQ package (Brzinsky-Fay, Kohler & Luniak, 2006), and the R package TraMineR (Gabadinho, Ritschard, Studer & Müller, 2009). SADI provides some tools that are not in SQ, and is much faster for some important functions. TraMineR is pretty attractive for those working in R, but SADI makes it possible to do a lot in a Stata environment, and has distance measures that are not in TraMineR.

Since some of the distance measures are relatively intensive to calculate, they are implemented as C plugins, rather than pure Stata or Mata code. This means that they are available only for Windows and Linux, 32- and 64-bit. If you would like to compile them for another platform, please contact Brendan Halpin, `brendan.halpin@ul.ie`, or see section 5.

This document summarises the functionality offered by SADI, with worked examples, and reproduces the help files (see section 6).

Many of the measures in SADI are discussed in detail in Halpin (2014) and Halpin (2012).

1.1 Referring to SADI

If you use SADI and would like to acknowledge it, please refer to this document, as follows:

> Brendan Halpin (2014), SADI: Sequence analysis tools for Stata, Working Paper WP2014-03, Department of Sociology, University of Limerick, http://www.ul.ie/sociology/pubs/wp2014-03.pdf.

2 Installation

The SADI package is hosted at http://teaching.sociology.ul.ie/sadi and can be installed as follows:

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

Several commands in the package depend on the `mm_expand()` Mata function in the `moremata` package, so you must also do:

```
ssc install moremata
```

I also recommend looking at the SQ package for sequence analysis, not least for its effective implementation of indexplots:

```
ssc install sq
```

3 Data requirements

Sequence analysis works with linear structures, usually longitudinal in time, that are discrete in both the time dimension and the state space. Typically, each element represents a time period or event in sequential order, and contains an observation in a categorical state space. A typical example is monthly labour market status.

SADI expects sequences to be represented by a consecutive run of variables, where the categories are numbered from 1 up to the number of categories. Thus each case contains a complete sequence, in wide format. Missing values are not accommodated, unless missing is treated as a category in its own right. Sequences of different length should start at element 1, and have a variable indicating their length.

4 Worked example

In this section, the functionality of SADI is presented. All the steps presented are included in a Stata do-file available at http://teaching.sociology.ul.ie/sadi/distances.do.

4.1 Quick start

The following Stata commands will set up and run the example described in the following pages:

```
net from http://teaching.sociology.ul.ie/sadi
net install sadi
ssc install moremata
ssc install sq
do http://teaching.sociology.ul.ie/sadi/distances.do
```

4.2 Data

We use data from McVicar and Anyadike-Danes (2002), and set up a substitution matrix (i.e., a description of distances within the state space). The data consist of 72 monthly observations (`state1` to `state72`) in a six-element state space, to do with the transition from school to work.

```
set matsize 1000
use http://teaching.sociology.ul.ie/bhalpin/mvad
sort id

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 )
```

4.3 Pairwise distances

Sequence analysis proceeds by calculation distances between pairs of sequences, typically generating matrices of distances between all pairs.

Most distance measures work with the sequences as strings of state-variables, and have a relatively consistent format. This code creates six pairwise distance matrices, using six different distance measures:

```
oma        state1-state72, subsmat(mvdanes) pwd(omd) length(72) indel(1.5)
omav       state1-state72, subsmat(mvdanes) pwd(omv) length(72) indel(1.5)
hollister  state1-state72, subsmat(mvdanes) pwd(hol) length(72) timecost(0.5) localcost(0.5)
twed       state1-state72, subsmat(mvdanes) pwd(twd) length(72) lambda(0.5) nu(0.04)
```

```
hamming    state1-state72, subsmat(mvdanes) pwd(ham)
dynhamming state1-state72,                  pwd(dyn)
```

The commands start with a variable list which defines the sequence, and then have different options. Where relevant, `subsmat()` provides the substitution cost or state-space distance information. The mandatory `pwd()` options names the matrix in which the pairwise distances are returned. Where it is possible to compare sequences of different length, `length()` specifies the length either as a constant or a variable. Other options are command-specific.

The measure `omav` is described in Halpin (2010), `hol` in Hollister (2009), `dynhamming` in Lesnard (2008), and `twed` in Marteau (2007, 2008) and Halpin (2014).

4.3.1 X/t

The X/t measure, a duration-weighted, spell-oriented version of Elzinga's "number of matching subsequences" (NMS) similarity measure, is calculated with `combinadd`. It is described in Elzinga (2006) and discussed in Halpin (2014). It works with spells (consecutive runs of periods in the same state) weighted by duration, so we need to restructure the data (one observation per spell, with a state variable, and a length variable) . The `combinprep` command does the restructuring, and `combinadd` calculates the distances. We need to know the maximum number of spells in the data, which is returned as `r(maxspells)` by `combinprep`.

```
preserve
combinprep, state(state) length(len) idvar(id) nsp(nspells)
local spmax = r(maxspells)
combinadd state1-len`spmax', pwsim(xts) nspells(nspells) nstates(6) rtype(d)
restore
```

4.3.2 Data-driven substitution matrix

Sometimes researchers use theory or prior information to generate the substitution matrix. Other times they prefer to use the data to generate it, from transition rates (note that `dynhamming` does this automatically, but using time-varying transition rates). This may or may not be a good idea.

The command `trans2subs` creates a matrix of the transition-rate based distances. Typically transitions will occur much less often than once per time-unit, so the diagonal will be heavily populated. Thus the off-diagonal transition rates will be low, and distances will have low variability. If we exclude the diagonal, we get distances with greater variability.

Distances are defined as $2 - p_{ij} - p_{ji}$ where $p_{ij} = \frac{n_{ij}}{n_{i+}}$.

To calculate the transition rates, the data has to be in long format:

```
preserve
reshape long state, i(id) j(m)
trans2subs state, id(id) subs(tpr1)
trans2subs state, id(id) subs(tpr2) diag
restore
```

This yields:

```
. matrix list tpr1

symmetric tpr1[6,6]
           c1         c2         c3         c4         c5         c6
r1          0
r2  1.147539          0
r3  1.064734   1.849958          0
r4  1.643575   1.757525   1.671111          0
r5  1.182927   1.844291       1.96    1.90181          0
r6  1.207729   1.525335   1.831594   1.803575   1.608297          0
```

```
. matrix list tpr2

symmetric tpr2[6,6]
          c1        c2        c3        c4        c5        c6
r1         0
r2  1.967601         0
r3   1.98727  1.993341         0
r4  1.984684  1.987531  1.982969         0
r5  1.959993  1.992045  1.999488  1.994867         0
r6  1.951231   1.96336  1.996033  1.985649  1.972029         0
```

We can then calculate OMA distances using the transition-derived substitution costs, excluding the diagonal:

```
oma        state1-state72, subsmat(tpr1) pwd(tpr) length(72) indel(1.5)
```

4.4 Examining distance matrices

4.4.1 Comparing distances

Between different distance measures and different parameterisations (substitution costs) we have now eight pairwise distance matrices. The simplest way to compare them is correlation. The command `corrsqm` reports the Pearson correlation between the lower triangles of two square (symmetric) matrices, optionally excluding the diagonal (which, for distance matrices, is filled with zeros for all measures).

```
foreach dist in dyn ham twd hol omv xts tpr {
  corrsqm omd ‘dist’, nodiag
}
```

This yields:

```
VECH correlation between omd and dyn: 0.7915
VECH correlation between omd and ham: 0.9856
VECH correlation between omd and twd: 0.8065
VECH correlation between omd and hol: 0.9898
VECH correlation between omd and omv: 0.9197
VECH correlation between omd and xts: 0.1135
VECH correlation between omd and tpr: 0.7701
```

Note the very high correlation with OMA of the Hamming and Hollister measure, the very low correlation of the combinatorial X/t measure, and the relatively big difference between OMA with the original substitution cost matrix and OMA with the transition-rate based matrix.

4.4.2 The triangle inequality

For many of the uses to which these measures will be put, it is necessary that they imply a metric space. This requires, inter alia, that the distances obey the triangle inequality: for all A and B, there is no C such that $d(A,B) > d(A,C) + d(C,B)$. The `omav` and `hollister` distances do not fulfill this requirement (see Halpin, 2014).

```
foreach dist in dyn ham twd hol omv xts tpr {
  metricp ‘dist’
}
```

This results in the following output:

```
Matrix dyn is consistent with a metric space
Matrix ham is consistent with a metric space
Matrix twd is consistent with a metric space
Shorter route exists between seq   1 and seq 210 -- 2.056 > 2.042
Shorter route exists between seq   2 and seq  12 -- 1.556 > 1.542
Shorter route exists between seq   2 and seq  28 -- 1.528 > 1.521
Shorter route exists between seq   2 and seq  56 -- 0.931 > 0.924
Shorter route exists between seq   2 and seq  64 -- 0.701 > 0.694
Shorter route exists between seq   2 and seq  71 -- 1.361 > 1.347
Shorter route exists between seq   2 and seq  77 -- 0.889 > 0.882
Shorter route exists between seq   2 and seq  81 -- 0.903 > 0.896
Shorter route exists between seq   2 and seq 113 -- 2.389 > 2.375
Shorter route exists between seq   2 and seq 142 -- 0.486 > 0.472
Matrix hol is NOT consistent with a metric space
Shorter route exists between seq   1 and seq   2 -- 0.161 > 0.131
Shorter route exists between seq   1 and seq   3 -- 0.143 > 0.121
Shorter route exists between seq   1 and seq   5 -- 0.165 > 0.131
Shorter route exists between seq   1 and seq   6 -- 0.097 > 0.072
Shorter route exists between seq   1 and seq   7 -- 0.093 > 0.065
Shorter route exists between seq   1 and seq   8 -- 0.065 > 0.043
Shorter route exists between seq   1 and seq   9 -- 0.069 > 0.049
Shorter route exists between seq   1 and seq  10 -- 0.150 > 0.108
Shorter route exists between seq   1 and seq  11 -- 0.132 > 0.106
Shorter route exists between seq   1 and seq  12 -- 0.163 > 0.139
Matrix omv is NOT consistent with a metric space
Matrix xts is consistent with a metric space
Matrix tpr is consistent with a metric space
```

The `hol` and `omv` distance matrices are not metric, and are hence of limited value. Only the first ten exceptions are printed, unless the option `detailed` is given.

4.5 Cluster analysis

Very often, sequence analysis proceeds by conducting cluster analysis on the pairwise distance matrix. Here we do it for the `oma` and `twed` distances, generating cluster solutions with 8 and 12 clusters in each case.

```
clustermat wards omd, name(oma) add
cluster generate o=groups(8 12)

clustermat wards twd, name(twd) add
cluster generate t=groups(8 12)
```

4.5.1 Comparing cluster solutions

We can compare the cluster solutions for the two measures in a number of ways. The Adjusted Rand Index (Hubert & Arabie, 1985; Vinh, Epps & Bailey, 2009) reflects agreement defined as the extent to which the members of a pair of cases, if in the same cluster in one solution, are in the same cluster in the other:

```
. ari o8 t8
Adjusted Rand Index:  0.5977
```

Clusterings are "unlabelled classifications", in that clusters can only be identified by reference to the cases they contain. In this sense, a cluster in a clustering based on one distance matrix is "the same" or similar to a cluster in a clustering based on another matrix only to the extent that they contain (mostly) the same cases. The `permtab` command crosstabulates two (equal-sized) solutions, permuting the values of one to maximise the agreement. The permuted classification can be saved as a new variable:

```
permtab o8 t8, gen(pt8)
tab o8 pt8
```

The permutation seeks to maximise Cohen's κ as an index of agreement (Reilly, Wang & Rutherford, 2005), and reports the κ_{max} to be 0.7346.

```
. tab o8 pt8

           |                               pt8
        o8 |         1          2          3          4          5          6          7          8 |     Total
-----------+----------------------------------------------------------------------------------------+----------
         1 |        92          1          0          0          0          0          0          0 |        93
         2 |        41         96          0          2          0          0          0          0 |       139
         3 |         0          0         57          4          0          0          0          1 |        62
         4 |         0          4          2        123          0          0         16          1 |       146
         5 |        11         19          0          2         39         13          9          0 |        93
         6 |         0          0          0          0          2         28          0          0 |        30
         7 |         2          5          0          1          4          0         28          7 |        47
         8 |         0          0          0          0          1          0         14         87 |       102
-----------+----------------------------------------------------------------------------------------+----------
     Total |       146        125         59        132         46         41         67         96 |       712
```

Permutation is simple but expensive if there are many categories. For 12 clusters, permutation takes $9 \times 10 \times 11 \times 12 = 11880$ times as long as for 8. To deal with this, `permtabga` yields an approximate-best permutation using a genetic algorithm:

```
permtabga o12 t12, gen(pt18)
```

4.5.2 Discrepancy

Studer et al's discrepancy measure brings a pseudo-ANOVA perspective to distance matrices (Studer, Ritschard, Gabadinho & Müller, 2011). If we partition the matrix using a cluster solution, or a pre-existing observed characteristic, we can compare the average distance to the centre of the partition to the average distance to the overall centre, and generate a pseudo-R^2 measure. The approach uses bootstrapping to generate p-values, and increasing the `niter()` option from the default 100 increases precision.

```
. discrepancy o8, dist(omd) id(id)

Discrepancy based R2 and F, 100 permutations for p-value

             | pseudo R2   pseudo F    p-value
-------------+--------------------------------
          o8 |  .5310534    113.891        .01

. discrepancy o12, dist(omd) id(id)

Discrepancy based R2 and F, 100 permutations for p-value

             | pseudo R2   pseudo F    p-value
-------------+--------------------------------
         o12 |  .5990087   95.06125        .01

. discrepancy grammar, dist(omd) id(id)
```

```
Discrepancy based R2 and F, 100 permutations for p-value

             | pseudo R2   pseudo F    p-value
-------------+---------------------------------
     grammar |  .0272244   19.87031        .01

. discrepancy grammar, dist(omd) id(id) niter(1000)

Discrepancy based R2 and F, 1000 permutations for p-value

             | pseudo R2   pseudo F    p-value
-------------+---------------------------------
     grammar |  .0272244   19.87031       .001
```

4.6 Summarising sequences and clusters

4.6.1 String representations of sequences

We can create string representations of sequences, which makes it much easier to get a visual overview of the data, and allows searching for patterns:

```
. stripe state1-state72, gen(seqstr) symbols("EFHSTU")

. list seqstr in 1/5, clean

                                                                             seqstr
  1.    TTEEEETTEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
  2.    UUFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
  3.    UUTTTTTTTTTTTTTTTTTTTTTTTTFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEEEEEEEEEEUU
  4.    TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTEEEEEEEEEEEEEUUUUUUUUU
  5.    UUFFFFFFFFFFFFFFFFFFFFFFFFFHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
```

Stata's `regex` system makes it easy to search for patterns in these representations. For instance, `count if regexm(seqstr, "^E+$")` will count sequences 100% in employment, while `count if regexm(seqstr, "U[^U]")` will count sequences where we observe an exit from unemployment.

4.6.2 Medoids: typical sequences

We can characterise clusters in many ways (see below for graphics, `chronogram` and `sqindexplot`). One way is to pick a "medoid", the sequence nearest the centre of the cluster. The `discrepancy` command has an option to save this distance as a variable, which allows us to identify the medoid:

```
discrepancy o8, dist(omd) id(id) dcg(dx)
sort o8 dx
by o8: gen medoid = _n==1
```

The medoids are all pretty simple, and quite distinct:

```
. list o8 seqstr if medoid, clean

       o8                                                                     seqstr
  1.    1   EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
 94.    2   TTTTTTTTTTTTTTTTTTTTTTTTEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
233.    3   FFFFFFFFFFFFFFFFFFFFFFFFFFFHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
295.    4   SSFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
441.    5   TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTEEEEEEEEEEEEEUUUUUUUUU
534.    6   TTTTTTTTTTTTTTTTTTTTUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
564.    7   SSSSSSSSSSSSSSSSSSSSSSSSSSEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
611.    8   SSSSSSSSSSSSSSSSSSSSSSSSSSHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
```

4.6.3 Cumulated duration

The sequence-wise total duration in each state is also an interesting summary:

```
cumuldur state1-state72, cd(dur) nstates(6)
```

Even though cumulated duration discards all order information, it differentiates the clusters very strongly:

```
. table o8, c(mean dur1 mean dur2 mean dur3) format(%5.2f)

----------------------------------------------
       o8 | mean(dur1)  mean(dur2)  mean(dur3)
----------+-----------------------------------
        1 |      67.99        2.74        0.00
        2 |      43.86        2.58        0.00
        3 |       6.34       27.34       36.35
        4 |      34.22       31.15        1.08
        5 |      27.55        9.23        0.00
        6 |       5.30        3.57        0.00
        7 |      32.94        4.79        3.79
        8 |       4.59        4.35       33.24
----------------------------------------------

. table o8, c(mean dur4 mean dur5 mean dur6) format(%5.2f)

----------------------------------------------
       o8 | mean(dur4)  mean(dur5)  mean(dur6)
----------+-----------------------------------
        1 |       0.04        0.52        0.71
        2 |       1.25       22.50        1.82
        3 |       0.77        0.00        1.19
        4 |       0.95        2.25        2.35
        5 |       1.14       11.00       23.07
        6 |       4.47        7.93       50.73
        7 |      21.64        3.51        5.34
        8 |      26.68        0.66        2.49
----------------------------------------------
```

4.6.4 Entropy

We can look at the entropy of cumulated duration. The `entropy` command calculates a simple measure of Shannon entropy (maximal if all states are equally likely, minimal if only one state is visited):

```
// first drop the cumulated duration variables as the
// entropy command will recreate these
drop dur1-dur6

entropy state1-state72, gen(ent) cd(dur) nstates(6)
```

Because it completely ignores order, this is not an entirely appropriate measure of sequence complexity. However, entropy levels differ greatly by cluster:

```
. table o8, c(mean ent) format(%5.2f)

----------------------
       o8 |  mean(ent)
```

```
----------+-----------
        1 |       0.26
        2 |       1.02
        3 |       1.30
        4 |       1.18
        5 |       1.39
        6 |       0.94
        7 |       1.38
        8 |       1.32
----------------------
```

4.6.5 Number of spells

The total number of spells in a sequence is a measure of its volatility:

```
. nspells state1-state72, gen(nsp)
. table o8, c(mean nsp) format(%5.2f)

----------------------
       o8 |  mean(nsp)
----------+-----------
        1 |       2.14
        2 |       3.39
        3 |       3.73
        4 |       3.99
        5 |       4.97
        6 |       3.00
        7 |       3.94
        8 |       3.37
----------------------
```

4.7 Graphics

There are two key graphics associated with sequence analysis, the state-distribution plot (or chronogram) and the indexplot. I also present a representation of the time-structure of transition rates.

4.7.1 Chronogram

The chronogram represents the distribution of states at each time unit, hiding individual continuity but yielding a more digestible summary:

```
chronogram state*, id(id) by(o8, legend(off)) name(chronogram, replace)
```

See Figure 1.

4.7.2 Indexplot

The indexplot plots each sequence as a line, and thus reproduces the sequence data in full. The `sqindexplot` command from the `SQ` package makes a good job of this task, so I haven't re-implemented it for `SADI`.

Do `ssc install sq` if necessary.

To make the full sequence data visually digestible, it needs to be grouped and ordered carefully. If we plot by cluster, the order within cluster is critical. My preference is to generate a maximal clustering (as many clusters as distinct sequences). This allows us to order sequences within clusters such that subcluster-structure is preserved (such that the sequences are in dendrogram order). It makes clustered indexplots more readable, and less dependent on cutting at an arbitrary number of clusters.

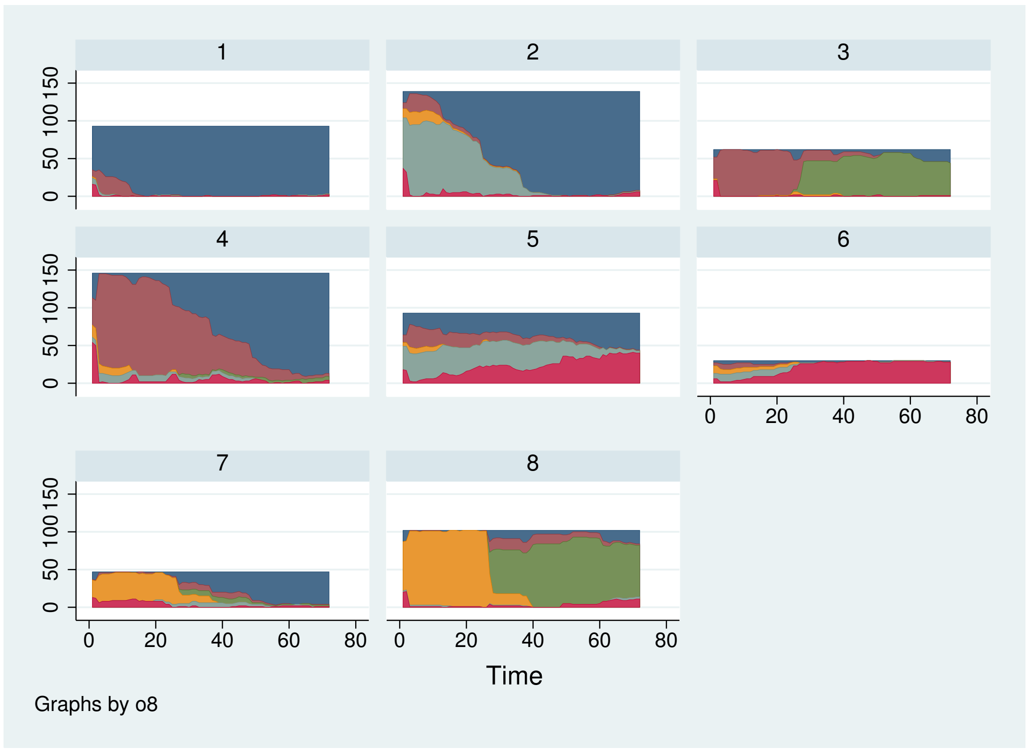

Figure 1: Chronogram, by 8-cluster OMA solution

```
cluster generate o999 = groups(750), name(oma) ties(fewer)
```

SQ wants sequence data in long format, and sqset:

```
preserve
reshape long state, i(id) j(m)
sqset state id m
sqindexplot, by(o8, note("") legend(off)) order(o999) name(indexplot, replace)
restore
```

See Figure 2.

4.7.3 Transition pattern graph

The trprgr command creates a composite graphic, with a column of graphs (chronograms) representing the 6 states over time, and a 6x6 grid representing the transition rates between states over time.

```
trprgr state*, id(id) gmax(485)
```

By design, this command shows transition rates on the diagonal on the range 0.9-1.0, and those off the diagonal on the range 0.0-0.1, but in practice these ranges are often exceeded. See the ceiling and floor options.

See Figure 3.

4.7.4 maketrpr

The maketrpr generates the matrix of transition rates that is used by dynhamming and trprgr, using tssmooth to average over a moving window of successive transitions. It may be of interest to inspect this data in matrix form as much as in the trprgr graph.

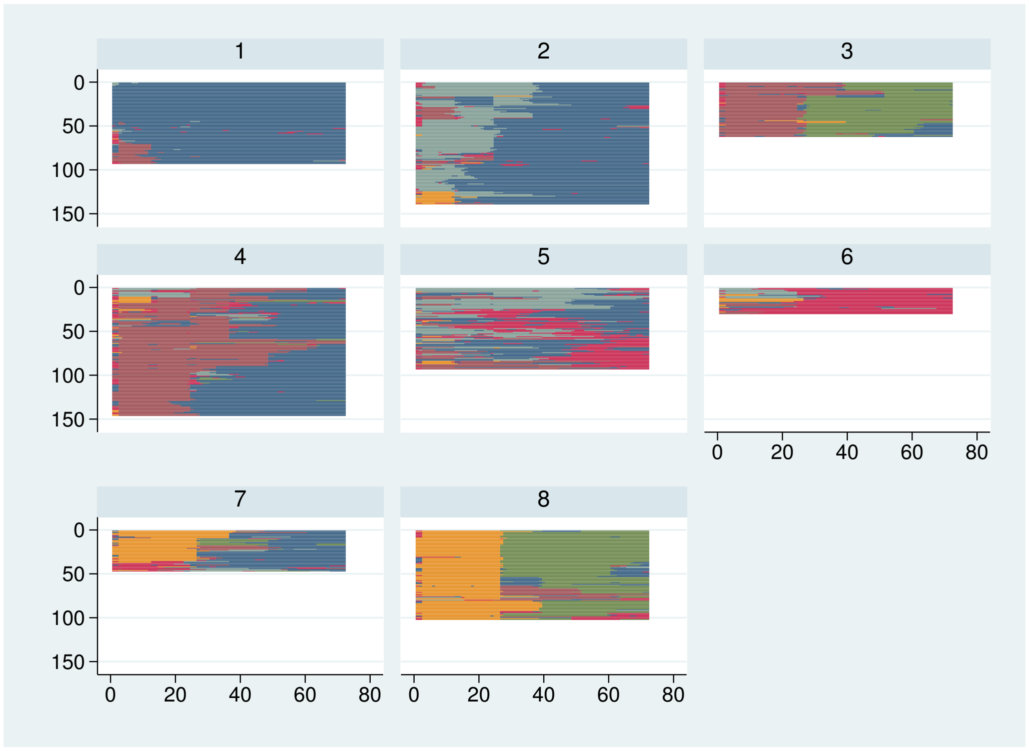

Figure 2: Indexplot, by 8-cluster OMA solution

```
maketrpr state*, mat(mkt) ma(5)
```

For m categories and t time points, this creates a $(t-1)m \times m$ matrix, where each successive $m \times m$ panel represents a time-specific pattern of transitions (smoothed). In this example there are no early observations in state 3 (higher education) so its exit rate is undefined:

```
. matlist mkt[1..6,.]

             | __0000071   __0000072   __0000073   __0000074   __0000075
-------------+-------------------------------------------------------
          r1 |  .8853028    .0684874           0    .0297906    .0101547
          r2 |  .0054704    .9890236           0           0    .0024511
          r3 |         .           .           .           .           .
          r4 |  .0050338    .0228499           0    .9606053     .009611
          r5 |  .0129851    .0049919           0    .0026247    .9770244
          r6 |  .0473975    .1041914           0    .0354406    .0490166

             | __0000076
-------------+-----------
          r1 |  .0062645
          r2 |  .0030549
          r3 |         .
          r4 |  .0019001
          r5 |  .0023739
          r6 |  .7639539

. matlist mkt[25..30,.]

             | __0000071   __0000072   __0000073   __0000074   __0000075
-------------+-------------------------------------------------------
```

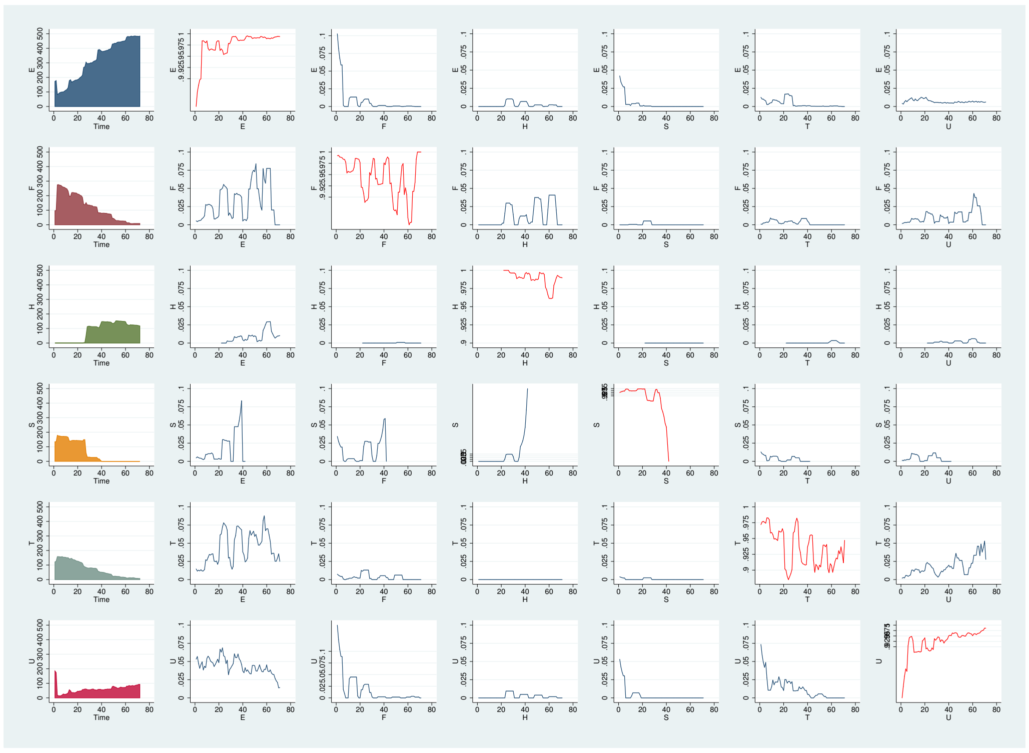

Figure 3: Transition pattern

```
         r25 |  .9244586   .0410925           0   .0188743   .0070362
         r26 |  .0071804   .9871928           0          0   .0026261
         r27 |         .          .           .          .          .
         r28 |  .0030203   .0137099           0    .975197   .0057666
         r29 |  .0136273   .0029951           0   .0015748   .9771932
         r30 |  .0481792   .0625148           0   .0212644   .0368173

             | __0000076
-------------+-----------
         r25 |  .0085383
         r26 |  .0030007
         r27 |         .
         r28 |  .0023062
         r29 |  .0046096
         r30 |  .8312242
```

5 Compiling plugins

The C code for the distance measures resides in two main files, `omamatv3.c` and `elzspelladd.c`, both available at http://teaching.sociology.ul.ie/sadi. These need to be compiled with `uthash.h`, by Troy D. Hanson (http://uthash.sourceforge.net), which provides hash functions used in `elzspelladd.c`, and with `stplugin.c` and `stplugin.h`. The latter two files are provided by Stata. See Stata Corp's instructions for compiling plugins at http://www.stata.com/plugins/.

All five files are available at http://teaching.sociology.ul.ie/sadi:

- omamatv3.c: http://teaching.sociology.ul.ie/sadi/omamatv3.c
- elzspellad.c: http://teaching.sociology.ul.ie/sadi/elzspellad.c
- uthash.h: http://teaching.sociology.ul.ie/sadi/uthash.h

- stplugin.c: http://teaching.sociology.ul.ie/sadi/stplugin.c
- stplugin.h: http://teaching.sociology.ul.ie/sadi/stplugin.h

You may find updated versions of `stplugin.c` and `stplugin.h` at the Stata site, and of `uthash.h` at http://uthash.sourceforge.net, but these are the versions used to create the published SADI plugins.

The published plugins are compiled for Windows and Linux, 32- and 64-bit, by cross-compilation on a 64-bit Linux system.

References

Brzinsky-Fay, C., Kohler, U. & Luniak, M. (2006). Sequence analysis with Stata. *Stata Journal*, *6*(4), 435–460.

Elzinga, C. H. (2006). *Sequence analysis: metric representations of categorical time series*. Amsterdam: Free University of Amsterdam.

Gabadinho, A., Ritschard, G., Studer, M. & Müller, N. S. (2009). *Mining sequence data in R with the TraMineR package: a user's guide for version 1.2*. University of Geneva.

Halpin, B. (2010). Optimal matching analysis and life course data: the importance of duration. *Sociological Methods and Research*, *38*(3), 365–388.

Halpin, B. (2012). *Sequence analysis of life-course data: a comparison of distance measures* (Working Paper No. WP2012-02). Dept of Sociology, University of Limerick. Ireland. Retrieved from http://www.ul.ie/sociology/pubs/wp2012-02.pdf

Halpin, B. (2014). Three narratives of sequence analysis. In P. Blanchard, F. Bühlmann & J.-A. Gauthier (Eds.), *Advances in sequence analysis: theory, method, applications*. Berlin: Springer.

Hollister, M. (2009). Is optimal matching suboptimal? *Sociological Methods and Research*, *38*(2), 235–264.

Hubert, L. & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(1), 193–218. Retrieved from http://www.springerlink.com/index/x64124718341j1j0.pdf

Lesnard, L. (2008). Off-scheduling within dual-earner couples: an unequal and negative externality for family time. *American Journal of Sociology*, *114*(2), 447–90.

Marteau, P.-F. (2007). Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. *ArXiv Computer Science e-prints*. eprint: cs/0703033. Retrieved January 26, 2008, from http://arxiv.org/abs/cs/0703033

Marteau, P.-F. (2008). Time Warp Edit Distance. *ArXiv e-prints*. Retrieved June 8, 2008, from http://arxiv.org/abs/0802.3522

McVicar, D. & 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.

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

Studer, M., Ritschard, G., Gabadinho, A. & Müller, N. S. (2011). Discrepancy analysis of state sequences. *Sociological Methods and Research*, *40*(3), 471–510.

Vinh, N. X., Epps, J. & Bailey, J. (2009). Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th international conference on machine learning*. Montreal, Canada.

6 Appendix: Help pages

The following pages reproduce the Stata help files for this list of commands:

- `ari`
- `chronogram`
- `combinadd`
- `combinprep`
- `corrsqm`
- `cumuldur`
- `discrepancy`
- `dynhamming`
- `entropy`
- `hamming`
- `hollister`
- `maketrpr`
- `metricp`
- `nspells`
- `oma`
- `omav`
- `permtab`
- `stripe`
- `trans2subs`
- `trprgr`
- `twed`

help ari

Title

ari —— Calculate the Adjuted Rand Index for a pair of unlabelled classifications

Syntax

ari *var1 var2* [if] [in]

Description

ari takes a pair of unlabelled classifications (e.g., two cluster solutions) and returns the Adjusted Rand Index, which has a maximum of 1 for perfect agreement, and where zero means no relationship. Returns r(ari).

References

N Xuan Vinh, J Epps and J Bailey (2009), Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?, *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada

L. Hubert and P Arabie (1985), Comparing Partitions, *Journal of Classification* 2(1), pp 193–218

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. ari a8 b8
```

help chronogram

Title

chronogram ——
Graph the time−dependent state distribution

Syntax

chronogram *varlist* (min=2) [if] [in] **,** *options* [options]

options	Description
ID	
id(*varname***)**	A unique case−id variable. Required.
Optional	
by(*string***)**	Graph by varlist, allows options.
textsize(*string***)**	Text size of labels.
proportional	Graph proportional distribution (useful with by).
*****	Accepts many graph options.

Description

chronogram takes a set of sequences described by *varlist* in wide format and graphs the time−dependent distribution of the state variable. This is sometimes called a chronogram, or the transversal state distribution.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. chronogram mon1−mon36, id(id) by(sex, legend(off))
```

help combinadd

Title

combinadd ―― Calculate inter-sequence distances using Elzinga's duration-weighted subsequence counting

Syntax

combinadd *varlist* **,** *options* [option]

options	Description
Required	
nspells(*string*)	Name the variable which stores the number of spells
nstates(*string*)	Name the variable which stores the number of states
pwsim(*string*)	Name the matrix which will store the similarities or distances
Optional	
rtype(*string*)	Generate similarities or distances: "s" for similarities, "d" for distances, "r" for raw SXY values. Defaults to similarities.
workspace	Show workings
maxtuples(*integer*)	Maximum number of tuples (subsequences) to count in one sequence (default 40000)

Description

combinadd calculates a version of Elzinga's duration-weighted number of common subsequences measure for spell-structured data. The variable list must identify the spell structure in the form state-1, duration-1, state-2, duration-2, ... state-X, duration-X where X is the maximum number of spells observed. The nspells option identifies the variable that stores the case-specific number of spells.

States must be numbered as integers from 1 up.

The measure counts the number of spell sub-sequences common to each pair of sequences, weighted by the combined duration of the subsequence. The number of subsequences in a sequence increases very rapidly with the length of the sequence, with major consequences for memory demands. This implementation can handle sequences with of the order of 15 spells without too much difficulty. The maxtuples limit causes the command to stop if too many subsequences are observed, in order to avoid running out of memory. The maxtuples option can be used to judiciously raise this limit.

It uses a Stata plugin implementation.

See combinprep for a way of converting calendar representations to spell representations.

References

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.

Elzinga, C. H. (2006). Sequence analysis: Metric representations of categorical time series. Technical report, Free University of Amsterdam.

Halpin, Brendan. (2014). Three narratives of sequence analysis, Bühlmann et al (eds), {it: Advances in Sequence Analysis. Beyond the Core Program}, Springer

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. combinprep, state(m) length(l) idvar(id) nsp(nspells)

. local nsp = r(maxspells)
. local nel = r(nels)

. combinadd m1-l‘nsp’, pwsim(xtd) nspells(nspells) nstates(‘nel’)
    rtype(d)
```

help combinprep

Title

combinprep —— Transform sequences from wide calendar format to wide spell format

Syntax

combinprep, options

options	Description
Options	
state(*string***)**	Stub of state variable name, in reshape fashion
length(*string***)**	Stub of spell−length variable name (will be created)
idvar(*varname***)**	ID variable
nspells(*varname***)**	number−of−spells variable (will be created)

Description

combinprep takes sequence data in wide calendar format (i.e., a consecutive string of numbered state variables representing state in each time unit, with one case per sequence) and turns it into wide spell format (consecutive pairs of numbered state and duration variables) with a separate variable indicating the number of spells.

It returns the maximum number of spells observed in r(maxspells) and the range of the state variable in r(nels).

This can be used to prepare the data for {help:combinadd} and other techniques that focus on spell history rather than state history.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

Given sequences represented as consecutive variables s1−s40:

```
. combinprep, state(s) length(dur) nspells(nsp)
```

will generate a new structures with variable pairs s1, dur1 to sX, durX where X is the maximum number of spells observed. The spells are defined as consecutive runs in the same state, and their duration is recorded in the dur variable. The observed number of spells in each case is recorded in nsp.

help corrsqm

Title

corrsqm —— Calculate the correlation between the lower triangle of two symmetric matrices

Syntax

```
        corrsqm var1 var2 [if] [in] [,NODiag]
```

Description

corrsqm takes two symmetric matrices of the same dimension (e.g., distance matrices) and returns their correlation. More specifically, it returns the correlation between their lower triangles, including the diagonal by default. It fails if one or both matrix is not symmetric, or if the matrices are not the same size. It prints the correlation but also returns it in r(rho).

The option **nodiag** suppresses the diagonal, so that the correlation is between the lower triangles excluding the main diagonal.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
    . corrsqm dist1 dist2
```

help cumuldur

Title

cumuldur —— Calculate cumulated duration in states of a sequence

Syntax

cumuldur *varlist* , **cdstub(***string***) nstates(***int***)**

Description

cumuldur creates variables holding the cumulative duration in each state in a sequence described by the *varlist*. The **cdstub** option gives the prefix of the new variables, and **nstates** enumerates how many states there are. States must be numbered from 1 up. A warning is issued if the total duration is less than the sequence length (e.g., if the number of states is actually larger than that given in the option, or if there are missing values or values less than or equal to zero).

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. cumuldur m1-m40, cd(dur) nstates(3)
```

help discrepancy

Title

discrepancy —— Calculate Studer et al’s discrepancy measure

Syntax

```
        discrepancy groupvar , DISTmat(string) IDvar(varname)
                 [NITer(integer 100) DCG(string)]
```

options	Description
Required	
distmat(*matname***)**	names the distance matrix
idvar(*varname***)**	identifies the variable that links the sort−order of the distance matrix to the sort−order of the data
Optional	
niter(*interger***)**	number of permutations used to calculate p−value, defaults to 100
dcg(*string***)**	variable in which to store the distance to the group centre

Description

discrepancy calculates Studer et al’s measure of the *discrepancy* of a distance matrix, grouped by a categorical variable *groupvar*. The pseudo−R−squared and pseudo−F statistic are based on the extent to which the average distance to the centres of the groups are less than the average distance to the centre of the ungrouped distance matrix. The p−value is based on permutations (100 by default, but Studer et al recommend 1000 to 5000; set it to 1 for speed if you are not interested in the p−value).

The distance to the centre of the group can optionally be saved in a variable. This can be used to identify group medoids.

Returns:
r(p_perm)
r(pseudoF)
r(pseudoR2)

References

M Studer, G Ritschard, A Gabadinho and NS Müller, Discrepancy analysis of state sequences, *Sociological Methods and Research*, 40(3):471−510

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. discrepancy sex, dist(d) id(id) dcg(dsex)
. bysort sex: egen mindist = min(dsex)
. gen medoid = mindist == min(dsex)
```

help dynhamming

Title

dynhamming ——
Calculate inter−sequence distances using dynamic Hamming distance

Syntax

dynhamming *varlist* **,** *options* [option]

options	Description
Distances	
pwdist(*matname***)**	store the pairwise distances in *matname*, as a symmetric matrix. Will be created or overwritten.

Description

dynhamming calculates Lesnard’s dynamic Hamming distances between all pairs of sequences in the data, where *varlist* is a consecutive set of variables describing the elements of the sequence. Dynamic Hamming distances compare sequences element by element such that the inter−sequence distance is the sum of the element−wise distances. The element−wise distances are dynamic, based on the time−dependent structure of transition rates. The procedure uses maketrpr to calculate the transition rates, smoothing over a rolling seven (3+1+3) observations. See also trprgr which uses **maketrpr** to graph the time−dependent transition structure.

States must be numbered as integers from 1 up.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. dynhamming mon1−mon36, pwdist(dist)
. matrix list dist
```

help entropy

Title

entropy —— Calculate the Shannon entropy of a sequence

Syntax

```
entropy varlist , generate(string) cdstub(string) nstates(int)
```

Description

entropy creates a new variable holding the Shannon entropy of the sequence, given by the **generate()** option. As a side effect, it creates variables containing the relative cumulated duration (named by the **cdstub()** option, as in cumuldur). **nstates** tells Stata how many states there are. States must be numbered from 1 up.

Shannon entropy takes no account of sequence order, and is just based on the relative cumulated duration in the different states, with the formula:

− Sum [p_i * log_2(p_i)]

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. entropy m1-m40, gen(ent) cd(dur) nstates(3)
```

help hamming

Title

hamming —— Calculate inter−sequence distances using Hamming distance

Syntax

hamming *varlist* **,** *options* [option]

options	Description
Cost structure	
subsmat(*matname*)	use *matname* as the substitution cost matrix
Distances	
pwdist(*matname*)	store the pairwise distances in *matname*, as a symmetric matrix. Will be created or overwritten.

Description

hamming calculates Hamming distances between all pairs of sequences in the data, where *varlist* is a consecutive set of variables describing the elements of the sequence. Hamming distances compare sequences element by element such that the inter−sequence distance is the sum of the element−wise distances. The element−wise distances are given in the *subsmat()* substitution matrix.

States must be numbered as consecutive integers from 1 up, and the substitution cost matrix must be square, with dimension equal to the number of states. States must not be missing.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. matrix scost = (0,1,2,3\1,0,1,2\2,1,0,1\3,2,1,0)
. hamming mon1−mon36, subsmat(scost) pwdist(dist)
. matrix list dist

. hamming mon1−mon72, subsmat(scost) pwdist(dist)
. matrix list dist
```

help hollister

Title

hollister —— Calculate inter−sequence distances using Hollister’s Localized OM

Syntax

hollister *varlist* , *options* [option]

options	Description
Cost structure	
subsmat(*matname***)**	use *matname* as the substitution cost matrix
TIMEcost(*#***)**	use *#* as the time cost
LOCalcost(*#***)**	use *#* as the local cost
Sequence length	
length(*var***)**	sequence length, a variable or a constant if sequence length is fixed
Distances	
pwdist(*matname***)**	store the pairwise distances in *matname*, as a symmetric matrix. Will be created or overwritten.
Work−space	
workspace	(Optional) Causes the internal workspace matrices to be shown for each sequence comparison.
Normalisation	
STAndard	(Optional) If "longer", normalise by the length of the longer sequence, if "none" do no normalisation. Defaults to "longer".

Description

hollister calculates localised Optimal Matching distances between all pairs of sequences in the data, where *varlist* is a consecutive set of variables describing the elements of the sequence. It uses a Stata plugin implementation of Mattissa Hollister’s adaptation of the Needleman−−Wunsch algorithm. Thanks to Mattissa for help in figuring out to code it, but if I have introduced any errors they are my own.

Hollister’s measure differs from conventional OM by taking account of the neighbours of tokens involved in OM’s elementary operations. While this is attractive from a sociological point of view, it means the dissimilarity measure is not guaranteed to be metric. This is also true of omav.

States must be numbered as consecutive integers from 1 up, and the substitution cost matrix must be square, with dimension equal to the number of states. States must not be missing.

References

Halpin, Brendan. (2014). Three narratives of sequence analysis, Bühlmann et al (eds), {it: Advances in Sequence Analysis. Beyond the Core Program}, Springer

Hollister, M. (2009). Is optimal matching suboptimal? *Sociological Methods and Research*, 38(2):235--264.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. matrix scost = (0,1,2,3\1,0,1,2\2,1,0,1\3,2,1,0)
. hollister mon1-mon36, subsmat(scost) time(0.5) local(0.5)
    pwdist(dist) len(36)
. matrix list dist

. hollister mon1-mon36, subsmat(scost) time(0.5) local(0.5)
    pwdist(dist) len(dur)
. matrix list dist
```

Title

maketrpr –– Create a matrix containing transition rates from sequences

Syntax

maketrpr *varlist* (min=2) , *options* [option]

options	Description
Matrix	
MATrix(*matname***)**	store the transition rates in *matname*. Will be created or overwritten.
Moving average	
MA(*int***)**	Calculate a moving average over int+1+int periods. Defaults to 3.

Description

maketrpr takes a set of sequences described by *varlist* in wide format and creates an n by (n times t) matrix where each n by n section contains the smoothed transition rates for the corresponding time period. It uses tssmooth to create the smoothed rates, defaulting to a 3-unit look-head and look-back (i.e., a 7-wide moving average). If the number of states is 4 and there are 10 periods, it generates a (4x(10-1))x4 or 36x4 matrix, where T[1..4,1..4] contains the transition rates for time 1-2, T[5..8,1..4] for time 2-3 and so on.

This is essentially a utility program, and is used by dynhamming and trprgr.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. maketrpr mon1-mon36, mat(trp)
. matrix list trp
```

help metricp

Title

metricp —— Test a symmetric matrix of pairwise distances for the triangle inequality

Syntax

metricp *matname* [,**cou**ntlimit(*int*) **det**ailed]

Options	Description
Count limit	
countlimit(*int*)	Number of triangle−inequality infringements to report (defaults to 10, 0 means no limit)
detailed	Slowly identify the problem cases, not just the fact they exist.

Description

metricp takes a matrix of pairwise distances and tests that the triangle inequality is observed. If it finds triads infringing on the inequality it reports at most 10 before stopping (this is changed with the option **countlimit**; set that to zero for no limit). If there are no infringing cases and the matrix is large, it can be a little slow (tens of seconds). It is even slower with the **detailed** option (minutes), which identifies the infringing trio of sequences; without this option only the fact that there is a shorter route between sequence i and sequence j is reported.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. metricp pwd
```

Version

help nspells

Title

nspells —— Calculate number of spells in a sequence

Syntax

nspells *varlist* , **<u>gen</u>erate(***string***)**

Description

nspells creates a variable holding the number of spells in a sequence described by the *varlist*. The **generate** option names the variable, which will be created. Spells are defined as consecutive runs of the same value. Runs of missing values are counted as spells.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. nspells m1-m40, gen(nsp)
```

help oma

Title

oma —— Calculate inter-sequence distances using Needleman−−Wunsch algorithm

Syntax

oma *varlist* , *options* [option]

options	Description
Cost structure	
subsmat(*matname***)**	use *matname* as the substitution cost matrix
indel(*#***)**	use *#* as the indel cost
Sequence length	
length(*var***)**	sequence length, a variable or a constant if sequence length is fixed
Distances	
pwdist(*matname***)**	store the pairwise distances in *matname*, as a symmetric matrix. Will be created or overwritten.
Work−space	
workspace	(Optional) Causes the internal workspace matrices to be shown for each sequence comparison.
Duplicates	
DUps	(Optional) Force calculation of duplicate distances.
Normalisation	
STAndard	(Optional) If "longer", normalise by the length of the longer sequence, if "none" do no normalisation. Defaults to "longer".

Description

oma calculates Optimal Matching distances between all pairs of sequences in the data, where *varlist* is a consecutive set of variables describing the elements of the sequence. It uses a Stata plugin implementation of the Needleman−−Wunsch algorithm.

States must be numbered as consecutive integers from 1 up, and the substitution cost matrix must be square, with dimension equal to the number of states. States must not be missing.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. matrix scost = (0,1,2,3\1,0,1,2\2,1,0,1\3,2,1,0)
. oma mon1-mon36, subsmat(scost) indel(2) pwdist(dist) len(36)
. matrix list dist

. oma mon1-mon72, subsmat(scost) indel(2) pwdist(dist) len(dur)
. matrix list dist
```

help omav

Title

omav — Calculate inter-sequence distances using duration-compensated Needleman--Wunsch algorithm

Syntax

omav *varlist* , *options* [option]

options	Description
Cost structure	
subsmat(*matname***)**	use *matname* as the substitution cost matrix
indel(*#***)**	use *#* as the cost for insertions/deletions
Sequence length	
length(*var***)**	sequence length, a variable or a constant if sequence length is fixed
Distances	
pwdist(*matname***)**	store the pairwise distances in *matname*, as a symmetric matrix. Will be created or overwritten.
Duration adjustment	
facexp(*real***)**	(Optional) Exponent by which to adjust costs for duration (defaults to 0.5)
Work-space	
workspace	(Optional) Causes the internal workspace matrices to be shown for each sequence comparison.

Description

omav calculates duration-adjusted Optimal Matching distances between all pairs of sequences in the data, where *varlist* is a consecutive set of variables describing the elements of the sequence. It uses a Stata plugin implementation of an adapted Needleman–Wunsch algorithm. It differs from the standard **oma** command in that the costs of elementary operations are reduced for tokens that are elements of runs of the same value. By default, the cost of an operation on an element of an n-element sequence is changed by a factor of 1/n^f where f is given by the {opt:fac:exp} option. The value of f defaults to 0.5. A value of f of zero produces the same result as **oma** and a value of f of 1.0 weights all spells the same regardless of length.

Note: this measure is not guaranteed to be metric.

States must be numbered as consecutive integers from 1 up, and the substitution cost matrix must be square, with dimension equal to the number of states. States must not be missing.

References

Halpin, Brendan. (2010). Optimal Matching Analysis and Life Course Data: the importance of duration *Sociological Methods and Research*, 38(3)

Halpin, Brendan. (2014). Three narratives of sequence analysis, Bühlmann et al (eds), {it: Advances in Sequence Analysis. Beyond the Core Program}, Springer

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. matrix scost = (0,1,2,3\1,0,1,2\2,1,0,1\3,2,1,0)
. omav mon1-mon36, subsmat(scost) indel(2) pwdist(dist) len(36)
. matrix list dist

. omav mon1-mon72, subsmat(scost) indel(2) pwdist(dist) len(dur)
    facexp(0.75)
. matrix list dist
```

help permtab, permtabga

Title

permtab —— Rearrange columns of square table to maximise kappa

Syntax

permtab *rowvar colvar* [if] [in] [, gen(newvarname)]

permtabga *rowvar colvar* [if] [in] [, gen(newvarname)]

Description

permtab permutes the columns of the square crosstabulation of rowvar by colvar to maximise kappa. It is intended for use in comparing cluster solutions where the identity of categories from one solution to the other is only defined in terms of membership. Kappa measures the excess of observed over expected on the diagonal. Kappa_max is the Kappa of the best solution, and is reported.

A permuted version of *colvar* is created by the **gen** option.

Returns kappa_max as r(kappa).

Note: For numbers of categories much above 8 this procedure is slow and inefficient. For such cases **permtabga** uses a genetic algorithm approach to find an approximate solution.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. permtab a8 b8
```

help stripe

Title

stripe —— Create a single string variable representing the sequence

Syntax

stripe *varlist*, GENerate(newvarname) [SYMbols(string)]

Description

stripe Create a single string variable representing a sequence. Option **symbols** allows replacement of the default symbol series (the uppercase alphabet). This makes sequences easier to view, and enables one to use regular expressions to group sequences **[M-5] regexm()**.

Note: Assumes sequences are represented by consecutive variables containing numeric values.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. stripe state1-state40, gen(seqstr)
. stripe state1-state40, gen(seqstr) symbols("FPun")
. list seqstr if regexm(seqstr,"FFFF+.+nnnn")
. list seqstr if regexm(seqstr,"^F+n+$")
```

help trans2subs

Title

trans2subs ––
Create substitution matrix based on observed transitions

Syntax

trans2subs *state* [*if*] [*in*], *IDvar(id) SUBSmat(subsmat)* [*DIAGincl*]

Description

trans2subs calculates a substitution matrix based on observed transitions in the *state* variable, and puts it in the *subsmat* matrix. The data must be in long format, with *idvar* identifying the groups, and must be sorted.

Transitions are tabulated from period to period, and the substitution cost is defined as 2 − p_{a,b} − p{b,a} for off−diagonal cells, and 0 for diagonal cells. p_{a,b} is defined as the proportion of transitions from a in t which are to b in t+1. Note that, by default, cases which do not have a transition from one period to the next do not enter the calculation.

Options

IDvar(*idvar***)** specifies the ID variable.

SUBSmat(*mat***)** specifies the Stata matrix to which to write the substitution costs.

DIAGincl causes the cells on the diagonal to be used in the calculation.

Comments

One way to define substition costs for optimal matching is to use observed transition rates between states. Higher probabilities of transition imply greater similarity. This *may* often be a good idea, but it is not always the case. It is plausible that in some domains we will see high probabilities of transition between states which are substantively quite dissimilar, for instance between never−married and married.

The procedure expects the data in long calendar format, that is with each record representing a person−−month or case−−time−unit, sorted in temporal order within IDvar, the variable identifying the person or case. The resulting matrix is based on a cross−tabulation of state at t and t−1.

In this format only off−diagonal cases represent transitions: the diagonal represents months where the state is the same as the previous month. In the default, the diagonal cases are excluded, but the option DIAGincl causes them to be included in the calculation. Including them reduces the range of the substitution costs.

The strategy is based in part on that described in Rowher and Potter’s TDA manual, section 6.7.2.5, http://www.stat.ruhr−uni−bochum.de/pub/tda/doc/tman63/d06070205.zip

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
If your sequences are represented by consecutive variables s1-s50
    with ID id, first reshape long:

. reshape long s, i(id) j(m)
. trans2subs s, id(id) subs(smat)
. matrix list smat
. trans2subs s, id(id) subs(smat) diag
. matrix list smat2
```

help trprgr

Title

trprgr —— Graphically present transition rates from sequences

Syntax

trprgr *varlist* (min=2) , *options* [option]

options	Description
ID	
ID(*varname***)**	A unique case-id variable. Required.
Optional	
FLoor(*real***)**	Lowest transition rate for diagonal graphs.
CEIling(*real***)**	Highest transition rate for off-diagonal graphs.
GMax(*int***)**	Highest number of cases in any state at any time.
MOVingaverage(*int***)**	Look-back and look-ahead for moving average, default 3.
TEXtsize(*string***)**	Text size of labels.

Description

trprgr takes a set of sequences described by *varlist* in wide format and graphs the time-dependent transition rate structure. The graphic consists of m rows and m+1 columns, where m is the number of states. The first column displays the time-dependent distribution of states, and the remaining m by m structure reproduces an m by m transition table but with graphs of time-series of transition rates instead of single values.

Time series on the diagonal are plotted on the y-axis with a range of FLOOR to 1, those off the diagonal on the range 0 to CEILING. This assumes that retention in a state is more common than transitions between states, but setting FLOOR and CEILING respectively to 0 and 1 will give a common y-axis. The option GMAX sets the range for the state-distribution graphs, and should be set slightly greater than the maximum to make the state distribution graphs comparable.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. trprgr mon1-mon36, id(id)
```

help twed

Title

twed —— Calculate inter−sequence distances using Time−Warp Edit Distance

Syntax

twed *varlist* , *options* [option]

options	Description
Cost structure	
subsmat(*matname***)**	use *matname* as the substitution cost matrix
lambda(*#***)**	use *#* as the lambda parameter
nu(*#***)**	use *#* as the nu parameter
Sequence length	
length(*var***)**	sequence length, a variable or a constant if sequence length is fixed
Distances	
pwdist(*matname***)**	store the pairwise distances in *matname*, as a symmetric matrix. Will be created or overwritten.
Work−space	
workspace	(Optional) Causes the internal workspace matrices to be shown for each sequence comparison.
Normalisation	
STAndard	(Optional) If "longer", normalise by the length of the longer sequence, if "none" do no normalisation. Defaults to "longer".

Description

twed calculates Marteau's Time−Warp Edit Distance (TWED) between all pairs of sequences in the data, where *varlist* is a consecutive set of variables describing the elements of the sequence. Time−warping stretches and compresses the time dimension to achieve alignment in a manner similar but not identical to oma's insertion and deletion. Marteau (2007) describes a time−warping algorithm with a stiffness parameter (nu) and a gap penalty (lambda) which is metric as long as nu>0 (many time−warping distances are not metric). Because it uses compression instead of deletion, it respects the spell structure of the trajectory more than oma does. It uses a matching cost operation that is very close to OMA's substitution operation. The algorithm also differs by comparing adjacent pairs of elements in each sequence, rather than single elements.

It uses a Stata plugin implementation.

States must be numbered as consecutive integers from 1 up, and the substitution cost matrix must be square, with dimension equal to the number of states. States must not be missing.

References

Halpin, Brendan. (2014). Three narratives of sequence analysis, Bühlmann et al (eds), {it: Advances in Sequence Analysis. Beyond the Core Program}, Springer

Marteau, P.−F. (2007). Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. *ArXiv Computer Science e−prints*.

Marteau, P.−F. (2008). Time Warp Edit Distance. *ArXiv e−prints*, 802.

Author

Brendan Halpin, brendan.halpin@ul.ie

Examples

```
. matrix scost = (0,1,2,3\1,0,1,2\2,1,0,1\3,2,1,0)
. twed m1−m36, subsmat(scost) lambda(0.5) nu(0.15) pwdist(dist)
    len(36)
. matrix list dist

. twed m1−m72, subsmat(scost) lambda(0.5) nu(0.15) pwdist(dist)
    len(dur)
. matrix list dist
```