12-2013

Optimal Matching Distances between Categorical Sequences: Distortion and Inferences by Permutation

Juan P. Zuluaga

OPTIMAL MATCHING DISTANCES BETWEEN CATEGORICAL SEQUENCES: DISTORTION AND INFERENCES BY PERMUTATION

by

Juan P. Zuluaga

B.A. Universidad de los Andes, Colombia, 1995

A Thesis

Submitted to the Graduate Faculty

of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

St. Cloud, Minnesota

December, 2013

This thesis submitted by Juan P. Zuluaga in partial fulfillment of the requirements for the Degree of Master of Science at St. Cloud State University is hereby approved by the final evaluation committee.

Chairperson

Dean
School of Graduate Studies

OPTIMAL MATCHING DISTANCES BETWEEN CATEGORICAL SEQUENCES: DISTORTION AND INFERENCES BY PERMUTATION

Juan P. Zuluaga

Sequence data (an ordered set of categorical states) is a very common type of data in Social Sciences, Genetics and Computational Linguistics.

For exploration and inference of sets of sequences, having a measure of dissimilarities among sequences would allow the data to be analyzed by techniques like clustering, multimensional scaling analysis and distance-based regression analysis. Sequences can be placed in a map where similar sequences are close together, and dissimilar ones will be far apart. Such patterns of dispersion and concentration could be related to other covariates. For example, do the employment trajectories of men and women tend to form separate clusters?

Optimal Matching (OM) distances have been proposed as a measure of dissimilarity between sequences. Assuming that sequences are empirical realizations of latent random objects, this thesis explores how good the fit is between OM distances and original distances between the latent objects that generated the sequences, and the geometrical nature of such distortions.

Simulations show that raw OM dissimilarities are not an exact mirror of true distances and show systematic distortions. Common values for OM substitution and insertion/deletion costs produce dissimilarities that are metric, but not Euclidean. On the other hand, distances can be easily transformed to be Euclidean.

If differing values of a covariate lead to different latent random objects and thus different sequences, are there tests with enough power to catch such variability, among the natural intersequence random variation? Such tests should be robust enough to cope with the non-euclideanity of OM distances.

A number of statistical tests (Permutational MANOVA, MRPP, Mantel's correlation, and t-tests and median tests) were compared for statistical power, on associations between inter-item dissimilarities and a categorical explanatory variable. This thesis shows analytically that under simple conditions, the first four

tests are mathematically equivalent. Simulations confirmed that tests had the same power. Tests are less powerful with longer sequences.

Month Year

Approved by Research Committee:

Hui Xu Chairperson

ACKNOWLEDGMENTS

I would like to thank a number of people who provided essential support for the completion of this thesis: my advisor, Dr. Hui Xu, for his encouragement, interest, and patience; the members of my thesis committee; the faculty of the Statistics department in general; the Business Computing Research Laboratory (BCRL); the Office of Research and Sponsored Programs; Dr. Robert Johnson at Precollege Programs; the library of St. Cloud State University, especially Inter-Library Loan.

To Tina, thank you for your love and support all these years.

To my father.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter I

INTRODUCTION

SUBSTANTIVE MOTIVATION

The statistical examination in this thesis was motivated by a body of sociological and economic research on biographical trajectories. As an example of such empirical research work, McVicar & Anyadike-Danes (2002) tracked 712 young people in Northern Ireland for six years (72 months). During every month the person could be enrolled in school (of various kinds), in college, employed or jobless.

Their educational and work longitudinal trajectories were to be explained in terms of individual demographic characteristics.

For example, see raw data on the trajectories for the first six months for the first 10 people in the McVicar & Anyadike-Danes (2002) dataset mentioned above: in Figure 1, every horizontal row represent the trajectories of the first 10 people (in the dataset mentioned above); from left to right, colored tiles represent the state the person was for some of the 72 months[1] . (These plots and computations are made using the TraMineR package (Gabadinho, Ritschard, Muller, & Studer, 2011) from R (R Core Team, 2012)).

[1]In this dataset that comes from the educational system in Northern Ireland, young people still completing classes within their compulsory education are labeled as "In School" ; students may opt to enroll in some years of Further Education (equivalent to the last years of High School in the US); the equivalent to US College education would be Higher Education. Training is a government supported program of apprenticeships.

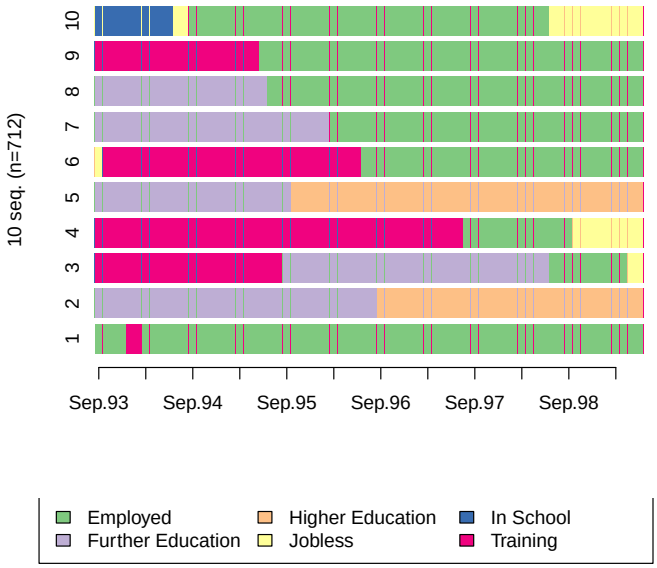


Figure 1

Education/Job trajectories of Northern Irish young adults

A trajectory is defined as an ordered (sequence AB is not the same as sequence BA) longitudinal set of categorical states in which the states are mutually exclusive (a person cannot be in both state A and B at the same time). (In this document I use *trajectories* and *sequences* as equivalents. Since I have in mind biographical trajectories I will be referring to individuals or subjects to the entity that goes through the states; but they should be thought of as any generic entity or item).

Traditional statistical methods have been employed to answer some questions about this type of data. Survival analysis (aka Event History Analysis in the Social Science literature) (Klein & Moeschberger, 2007; Yamaguchi, 1991; Blossfeld & Rohwer, 2002) can be used to explain the length of time to an event. Some hierarchical or repeated-measures models for categorical outcomes in a generalized linear frame have also been used (Ware & Lipsitz, 1988; Diggle, Liang, & Zeger, 1994). However, these traditional approaches have two major problems: they are based on assumptions about the data generating process (assumptions that may be unjustified), and their description is complex and even cumbersome due to their fine granularity (as they model specific stays or transitions through time).

An alternative approach, free of assumptions about the probability mechanism that generates the sequences, begins by condensing the information from each sequence by an indicator such as Elzinga's measure of complexity or turbulence (Elzinga, 2010). This indicator can be considered as the dependent variable in a regression, with individual level covariates as explanatory variables, to answer questions like "did people born before 1970 have more turbulent trajectories?".

Instead of directly analyzing instances of a random quantity (like observing a

sample of empirical values generated by a Gaussian generator with mean and variance parameters) [2], we observed quantities only indirectly tied with that generator:

- A sample of random objects (sequences) is generated,
- An algorithm that measures dissimilarity between every pair of random objects is defined and executed,
- such matrix of distances define a new kind of random object.

The approach presented here will be pertinent when there is no clear theoretical (probabilistic) model about how sequences are generated, nor a priori classification or typology of the sequences. Every sequence will be compared to every other one, and the resulting matrix of dissimilarities or distances will be the object of the statistical analysis.

DISSIMILARITY AMONG SEQUENCES BY OPTIMAL MATCHING

In Sociology, Andrew Abbott (Abbott & Forrest, 1986; Abbott & Hrycak, 1990; Abbott & Tsay, 2000) introduced a key methodological principle in the study of sequences: study their variability as represented by a matrix of distances among pairs of individual trajectories.

Second, Abbott proposed and used Optimal Matching distance as the algorithmic implementation of distance between two sequences. Optimal Matching (OM) has been a tool that the computational linguistics and the computational

[2]For our case, since a sequence is not a single scalar value like usual random quantities are, but a composite value, let us call it random object instead of random quantity

biology community have developed to compare strings of characters or genetic sequences (Sankoff & Kruskal, 1983; Gusfield, 1997). Given sequences represented as a string "ℵ∠\♣♢♭♡♭♮\" or "∠♭♣∇♭♡♭♮\", one can be tranformed into the other, by a number of elementary operations: deleting "letters", inserting new ones, or substituting new for old ones.

As an example, to get from "ℵ∠\♣♢♭♡♭♮\" to "∠♭♣∇♭♡♭♮\" reader can see these two ways, among many others:

ℵ	∠	\	♣	♢	♭	♡	♭	♮	\
∅	∠	♭	♣	∇	♭	♡	♭	♮	\

in which the ℵ has been deleted ($\aleph \to \emptyset$), the \ has been substituted by ♭ and a ♢ by a ∇.

Another possible way to align the two sequences is

ℵ	∠	\	♣	♢	♭	♡	♭	♮	\
∠	♭	♣	∇	∅	♭	♡	♭	♮	\

where $\aleph \to \angle, \angle \to \flat, \backslash \to \clubsuit, \clubsuit \to \nabla$ and ♢ gets deleted.

If one assumes that every insertion, deletion and subtitution are costly, which transformation should be considered as the one that happened? Intuitively, changes would occur along a path of least resistance, so the least costly transformation is the one that will be recorded.

For the sake of the example, let us assume that all substitutions have a cost of 2 units, and that all insertions or deletions have a cost of 1.

The first transformation used one indel and two substitutions, for a total cost

of $1 + 2 + 2 = 5$, while the second trasformation used one indel and four substitutions, for a total cost of 9. If 5 is really the least costly transformation, we can assign that 5 as a dissimilary measure.

For example, let us calculate a distance matrix with Optimal Matching for the first 10 people in the `mvad` dataset used before[3]:

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0  140  116  108  140   64   60   44   38    48
 [2,]  140    0   72  140   22  140   80   96  140   140
 [3,]  116   72    0   68   90   72   60   76   78   116
 [4,]  108  140   68    0  140   46  112  112   70    94
 [5,]  140   22   90  140    0  140   90   96  140   140
 [6,]   64  140   72   46  140    0   68   68   26    66
 [7,]   60   80   60  112   90   68    0   16   60    60
 [8,]   44   96   76  112   96   68   16    0   44    48
 [9,]   38  140   78   70  140   26   60   44    0    48
[10,]   48  140  116   94  140   66   60   48   48     0
```

See how the distance between sequence 7 and sequence 8 is quite small, while the distance between sequence 1 and 2 is large. Distance numbers do agree with the intuitive sense derived from observing Figure 1.

At this point, the reader will certainly guess that the distance measure depends on a prior matrix of subtitution and insertion–deletion costs. Such prior values, where do they come from?

In the empirical tradition of sociological studies, researchers can make a guess at costs based on substantive knowledge, for instance, by using information about ordination of states, or they can assume explicit ignorance, by making all substitution costs have the same value.

[3]Distance to be calculated with a substitution matrix set at equal costs (2) for all, and an insertion–deletion (indel) cost set at 1.

Choices of prior costs has been subject to strong criticism and debate (Levine, 2000; Wu, 2000; Abbott, 2000) and has opened a number of developments, such as refinements for Optimal Matching distance measures (Lesnard, 2010; Gauthier, Widmer, Bucher, & Notredame, 2009).

FROM EXPLORATION TO EXPLANATION

Clustering

The matrix of inter-sequence distances can be subject to clustering procedures [4], allowing the allocation of individual trajectories into a few archetypical trajectories. Such multinomial outcome can be explained by subject attributes by fitting a multinomial logistic model. As an example, if 200 academic trajectories of college students could be classed into 5 groups, could we use the race or income of each student as a predictor of its trajectory being of a type 3 instead of a type 1? McVicar & Anyadike-Danes (2002) is a good example of such approach.

However, going from similarity matrices to a mapping of clusters loses information, since within each cluster all trajectories are assumed to be the same. Instead, alternative approches use the pairwise distances explicitly as a measure of variability (or *discrepancy* as Studer, Ritschard, Gabadinho, & Muller (2011) call it) to be explained by covariates.

[4] Deciding which clustering procedure makes sense, among the many available is not a trivial problem (Kaufman & Rousseeuw, 1990)

Extracting Dimensions

A set of coordinates in a lower dimensional space can be found that approximates the raw matrix of inter-item distances, by means of Multidimensional Scaling Algorithms (either non-metric or metric – Principal Coordinates being a classical metric kind).

While we can assume that reducing raw distances to a lower space causes some information loss, a clear advantage is the ability to work with much less parameters.

Coordinates in a lower dimensional space can be used to make a 2 or 3 dimensional plot of items, and perhaps adding colors or shapes to items by their attributes.

Figure 2 has a plot of the space of dissimilarities among the 10 first sequences (in a two-dimensional plane that, hopefully, is not putting too much stress on the distances).

The coordinates in a space can be used not just to plot, but can also be understood as dependent variables mesured at the item level to be analyzed with multivariate linear models (Multiple Anova if explanatory variable is categorical, canonical correlation if explanatory variables are quantitative).

However, the extension of such linear approaches to statistical inference on matrices of OM distances is still a work in progress and has not entered the statistical toolbox at the elementary level.

There may be some reasons for the difficulty:

- Optimal Matching (or whatever algorithm is used to compute a measure of

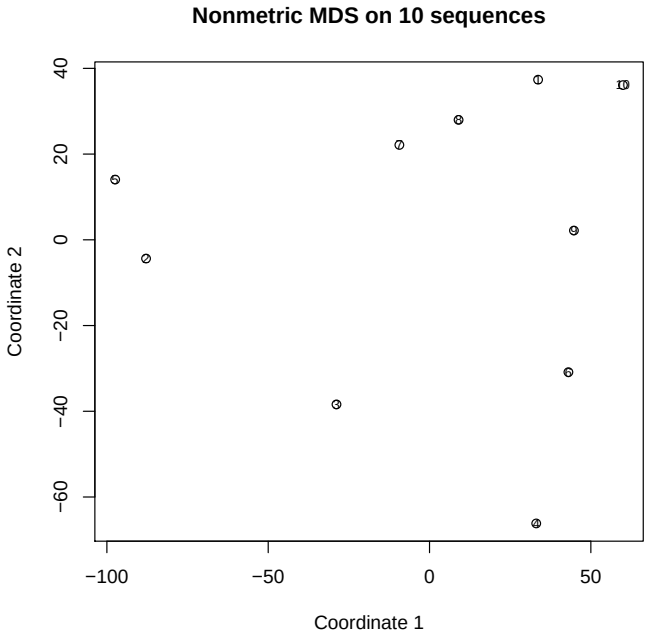


Figure 2

MDS plot of 10 sequences

dissimilarity between sequences) may be a biased representation of the dissimilarities among the true entities behind the sequence.

- the raw dissimilarities among items may not be metric or Euclidean at all, some mathematical operations with them may not be appropriate, and some conclusions may be suspicious.

- the distributions of the items (or a monotone transformation of them) may not follow a multivariate normal shape (the most tractable multivariate distribution) and MANOVA may not be robust enough against violations of the assumptions.

Permutational Approaches on Raw Dissimilarities

A way to bypass the concern with the geometric and distributional properties of inter-item distances for statistical inference is by simulation, specifically permutational strategies.

In the permutational strategy a Null Hypothesis is proposed and operationalized as a baseline, as the theoretically possible set of variations by permutation of the original matrix of distances.

Furthermore, as in Permutational MANOVA (Anderson, 2001), a measure of variability and discrepancy among items is operationalizaed, to be partitioned by covariates (perhaps in a regression tree).

A number of permutational approaches have been proposed by applied researchers in dissimilar fields like Spatial Statistics (Mantel, 1967; Mielke & Berry, 2001), Psychometrics (Hubert & Schultz, 1976), and Ecology (Manly, 1997;

Anderson, 2001). It seems worthwhile to clarify the differences and commonalities among these various approaches (Reiss et al., 2010).

RESEARCH QUESTIONS

First, I will describe the sequence generator to be used. Of the very many possible generators, this one was chosen because it was used by Studer et al. (2011).

Second, I study properties of such Optimal Matching distances, particularly what I call representational, geometric and distributional characteristics.

These two items will be contained in chapter II of this thesis.

Iin the third chapter I will describe and compare the power of some statistical methods for inference on Optimal Matching distance matrices: Mantel's test, permutational MANOVA (Permanova), Multiresponse permutational procedure (MRPP) and a test that uses Principal Coordinates (PCoA) from the matrix of distances for simple one way analysis. These tests make almost no assumptions about the process that generated the sequences. Some additional tests, that "cheat" by incorporate some knowledge about the sequence generation, will be run and their power compared.

In the fourth chapter I will explain the similarities between Mantel's test, PerManova and MRPP, by studying their algebraic formulation.

Chapter II

EXPLORING INTERSEQUENCE DISTANCES

GENERATING SEQUENCES

Many sequence generators could be imagined; I chose the generators in Studer et al. (2011) for being a key paper in the literature. A good alternative could be the ones described in Wilson (2006).

Studer et al. (2011), in the simulations they run, present three extremely simple mechanisms to generate sequences. Their simplicity consists in requiring few parameters (unlike, say, Markov chains, that would require $n(n-1)$ parameters to describe a transition matrix among n states). The sequences are both simple but still interesting.

In this simulation I will be using the third generator mechanism from Studer et al. (2011). A triplet is randomly generated, from a generator like

```
> library(actuar)# provides the loglogistic distribution rllogis.
> t1 <- 0 +  rllogis(1,shape=2.364,rate=0.078)
> t2 <- 10 + rllogis(1,shape=2.364,rate=0.126)
> t3 <- 20 + rllogis(1,shape=2.364,rate=0.078)
> c(t1,t2,t3)
[1] 42.18953 19.25333 31.05743
```

(Package `actuar` is described in Dutang, Goulet, & Pigeon (2008)). Each value in the triplet defines a turning point in each of three longitudinal vectors – vectors that

correspond to columns `S1, S2, S3` as seen in computer output; see how they usually start (at row 1) being 0, and then turn to 1 at a row that depends on the value of `t1, t2, t3`. We can think of every row representing a point in time; at each point there is a triple of three binary states. There can be 8 (2^3) possible triples (000, 001, ..., 111) that can be finally represented, as an example, by their decimal equivalent. This final digit ("0" to "7") represents a category, not a numerically valued dimension (state "7" is not seven times whatever state "1" is, it just represents a state 111 versus state 001). (Time sequence goes from top to bottom. Hidden presequences are `S1, S2, S3`, final categorical sequence is `s.final`.)

	S1	S2	S3	s.final
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	1	0	2
21	0	1	0	2
22	0	1	0	2
23	0	1	0	2
24	0	1	0	2
25	0	1	0	2
26	0	1	0	2
27	0	1	0	2
28	0	1	0	2
29	0	1	0	2
30	0	1	0	2
31	0	1	0	2
32	0	1	1	3
33	0	1	1	3
34	0	1	1	3
35	0	1	1	3
36	0	1	1	3
37	0	1	1	3
38	0	1	1	3
39	0	1	1	3
40	0	1	1	3

Using the same parameters for shape and intensity that were used by Studer et al. (shape=2.364, intensity=0.078, start={0,10,20}), results in a last column that is the combined outcome of the three preliminary sequences generated from three log-logistic points of transition.

Notice that if the value of the random quantity t_i generated by the log-logistic model exceeds the length of the vector (40 in this case), t_i will have no effect whatsoever in the final sequence: a triple like (10, 20, 42) will lead to the same sequence as (10, 20, 100).

```
   Sequence
10 0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-4-4-4-4-4-4-4-4-4-4
9  0-0-0-0-0-0-0-0-0-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4
8  0-0-0-0-0-0-0-0-0-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-6-6-6-6-7-7
7  0-0-0-0-0-0-0-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-6-6-6-6-6-6-6-6-6
6  0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-4-4-4-4-4-4-4-4-4-4-4-6-7-7-7-7
5  0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-2-2-6-6-6-6-7-7-7-7-7-7-7-7-7
4  0-0-0-0-0-0-0-0-0-0-0-0-4-6-6-6-6-6-6-6-6-6-6-6-6-6-6-6-6-6-6
3  0-0-0-0-0-0-0-0-0-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-6-6-6-6
2  0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-2-2-2-2-6-6-6-6-6-6-6-6-6-6
1  0-0-0-0-0-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-4-6-6-6-6-6-6-6-7
```

As an example, let us generate 5 sequences from baseline values (start parameter of 10), and 5 from an alternate model with a start of 20 (fixing rate aka intensity parameter at .0126). First a text representation of sequences, running horizontally from left to right, and then a graph in Figure 3 (where colors are equivalent to numbers in text representation).

DOES OM REPRESENT TRUE DISSIMILARITIES?

It must be evident that in our generator, a triple of values $\mathbf{t} = (t_1, t_2, t_3)$ defines a sequence, or that $\mathbf{t}_i \mapsto s_i$.

Since sequence s_i has a fixed length, for values $t_{i,k} > SequenceLength$, vector will keep 0 values – so a triple (12,23,45) will result in the same sequence as (12,23,50). Thus, s_i does not map to a unique $\mathbf{t}_i$.

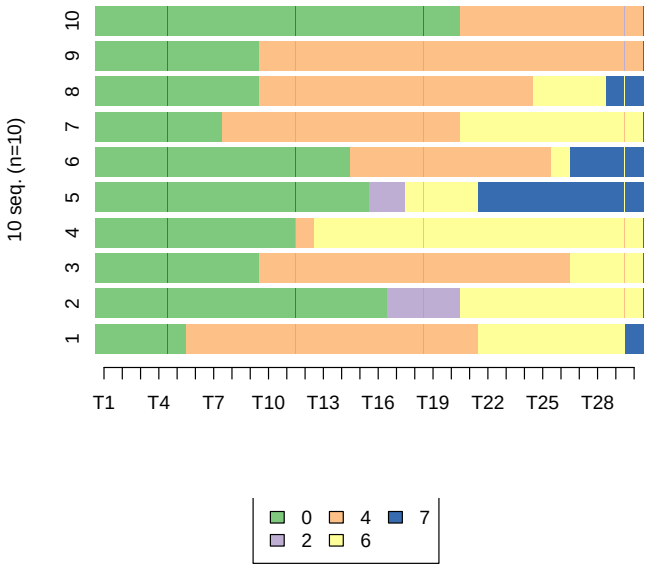


Figure 3

5 sequences from a population, 5 from other.

As a triple $\mathbf{t}_i = (t_{i1}, t_{i2}, t_{i3})$ lives in a 3-dimensional space, we can calculate a true distance among them, using a straightforward Euclidean distance.

$$\delta(\mathbf{t}_i, \mathbf{t}_j) = \sqrt{(t_{i1} - t_{j1})^2 + (t_{i2} - t_{j2})^2 + (t_{i3} - t_{j3})^2}$$

For notation, define Optimal Matching distance between sequences s_i and s_j as $d^{OM}(s_i, s_j)$.

If Optimal Matching is an adequate representation of the true distances between true causal vectors, the matrix $[d^{OM}(s_i, s_j)]$ should approximate the matrix $[\delta(\mathbf{t}_i, \mathbf{t}_j)]$. See Figure 4.

Do they correspond? Let us simulate some triples and sequences from them:

```
             t1       t2       t3
 [1,] 28.463043 18.11236 28.66758
 [2,]  9.610471 15.28498 42.25971
 [3,] 15.486113 19.25317 32.03188
 [4,]  2.847647 32.91659 35.83137
 [5,] 13.804390 22.62734 30.70195
 [6,]  1.955029 14.88841 30.82837
 [7,] 14.775725 14.50052 34.86750
 [8,]  9.450950 10.50209 47.06760
 [9,] 22.033764 21.23907 24.37952
[10,] 11.519529 11.83280 31.30503
[11,]  9.185978 16.50238 42.09373
[12,] 18.845659 19.00234 42.88460
[13,] 39.403860 17.08331 56.09644
[14,] 16.985166 14.78902 25.91499
[15,] 18.439759 19.25121 25.31387
```

Given such triples,

- calculate Euclidean distances among them,
- from each triple, generate the corresponding sequence,

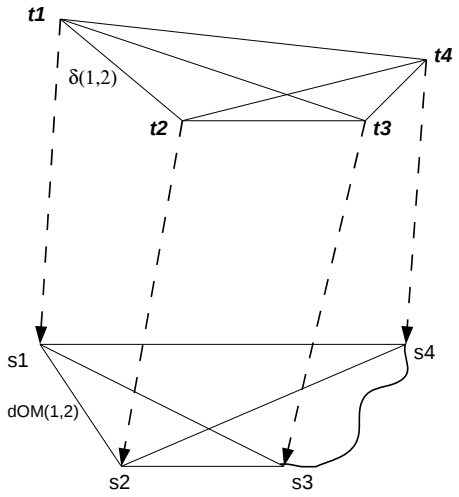


Figure 4

Fit between OM distances (below) and true inter-triple distances (above).

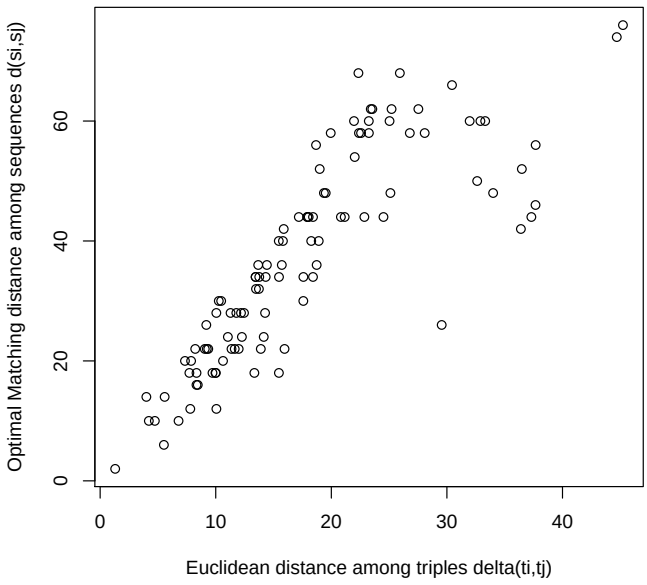


Figure 5

Shepard plot of true distances Vs. OM distances

- calculate the Optimal Matching distances among pairs of sequences (using a value of 2 for all substitution costs and a value of 1 for insertion/deletion costs):

Figure 5, a Shepard diagram (Kruskal & Wish, 1978), represents the (lack of) fit between true distances and OM distances. Each point in a scatterplot will represent a pair $(D(\mathbf{t}_i, \mathbf{t}_j), OM(s_i, s_j))$, or $(D_{i,j}, OM_{i,j})$.

If the OM distances fitted really well the true distances, the plot should show a very tight correlation (either linear or curvilinear). Across many examples of these plots, two kinds of distortions show up:

- a region of points located to the right of the plot, with high true Euclidean distances, but corresponding d^{OM} not high. They are due to cases where one or more t_{ik} components of (t_{i1}, t_{i2}, t_{i3}) were considerably greater than 40; the resulting sequence s_i, however, could not be very distinct from other sequences where the outlying $t_{ik} \approx 40$. So while in the Euclidean space such triple was far apart from others, in the OM pseudo-space[1] the corresponding sequence was not an outlier.

- when, by chance, such outlier triples are generated, and the points are better distributed towards the right side of the graph, it can be seen that the higher the true Euclidean distance, the higher the variance in OM distances. Points are distributed in the shape of a folding fan. The distortion grows with the magnitude of the distance, as it happens when a measure is a sum of other measures with weak or no correlation among them, its total variance being close to the sum of the variances of the components.

The plot of distortions suggests that some transformations of the original data could help in this situation. For instance, what if the squared distance of OM distances was used instead? Figure 6 shows that the graph is a bit more tight. This opens the consideration of appropriate transformations, in coming sections.

A visual exploration of such distortions of representation is not enough. We would need a measure of such distortion – and even more interesting, a measure of how much such distortion affects the power of statistical tests[2]

[1] "Pseudo-space" because, at this point of the presentation, it is not clear if OM define a Euclidean space.

[2] For now, only as a footnote, an approach could be like this: consider that the power of an statistic is a function of the data employed and of other nuisance factors that affect the power:

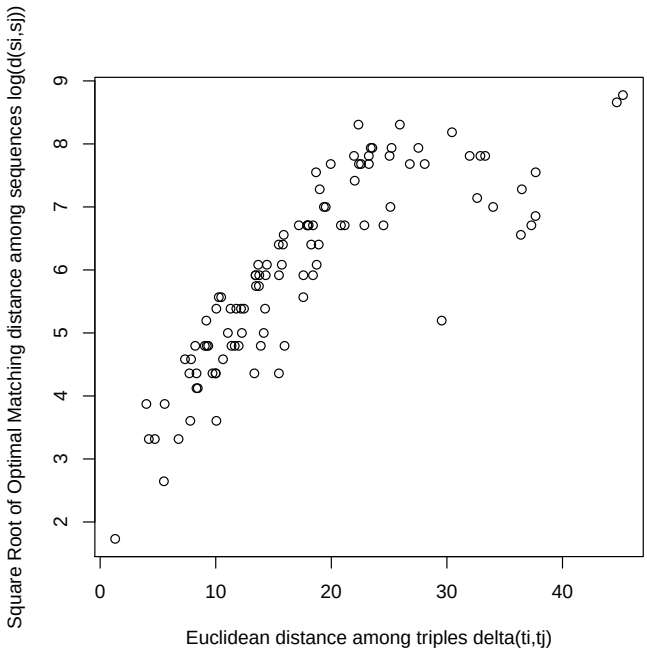


Figure 6

Shepard plot of true distances Vs. Square Root of OM distances

How Much Distortion?

A systematic exploration of the OM distances as a good representation of actual distances could borrow from the idea of *Stress* in the literature of Multidimensional Scaling (MDS) (Kruskal & Wish, 1978; Borg & Groenen, 2005).

The goal of multidimensional fitting was to find a function that operates on the matrix of distances such that a new matrix of coordinates for items (matrix size $n \times p$, where n is the number of items and p is the number of coordinates) is

sample size, how far apart are parameter values set by Null alternative hypotheses, etc. If, instead of using the "true" data (Euclidean distances), we were to use distorted data, the new power curve should be lower. Could such difference be tested for significance? Hopefully a measure of distortion would be associated with a measure of power loss.

obtained, with items that fit in a space of lower dimensions (p less than original $p*$), while still preserving most of the inter-item distances from the original data. Solutions could be evaluated to minimize stress, calculated, for instance, as

$$\text{Stress} = \sqrt{\frac{\sum[f(p_{ij}) - d_{ij}]^2}{\sum d_{ij}^2}},$$

where p_{ij} is the empirical distances among items i and j, $f(p_{ij})$ is the function on such matrix of empirical distances, and d_{ij} is an Euclidean distance among the same items as they get relocated in a space of lower dimensions (more precisely, euclidean distance obtained from the new coordinates of the items).

In our case, however, we just want a measure of representational *distortion* between the "true" (Euclidean) distances between the vector (triple) $\mathbf{t}$ of quantities that define sequences, and OM distances obtained among such sequences.

$$\text{Distortion} = \sqrt{\frac{\sum[OM_{ij} - \delta_{ij}]^2}{\sum OM_{ij}^2}}$$

where OM_{ij} is the calculated OM distance between sequences originated from triples i and j, and δ_{ij} is the true euclidean distance between triples i and j.

Why $\sum OM_{ij}^2$ in denominator instead of $\sum \delta_{ij}^2$? Since choosing different indel/substitution costs changes the values in the OM distance matrix, a measure of Distortion should control for the total magnitude of such changes ($\sum OM_{ij}^2$). $\sum \delta_{ij}^2$ on the other hand is fixed.

A Pearson correlation coefficient could also be used – in fact it receives the special name Cophenetic Correlation coefficient in the classification literature, when the question of interest is how well clustering groupings fit an original matrix of

dissimilarities; following Wikipedia's article on Cophenetic correlation, formula would be

$$c = \frac{\sum_{i<j}(OM_{ij} - \overline{OM})(\delta_{ij} - \bar{\delta})}{\sqrt{[\sum_{i<j}(OM_{ij} - \overline{OM})^2][\sum_{i<j}(\delta_{ij} - \bar{\delta})^2]}}$$

where $\overline{OM}$ is the average OM distance, and $\hat{\delta}$ is the average inter-triple true distance. The formula normalizes for matrices with non-normalized cell values.

In R, the Distortion function and the cophenetic correlation could be written as

```
> Distortion = function(m1,m2) {
+       m1.s = m1
+       m2.s = m2
+       Numerator = sum(as.vector(( m1.s - m2.s)^2))
+       Denominator=sum(as.vector(m1.s^2))
+       return(sqrt(Numerator/Denominator))
+
+ }
> CopheneticCorrel = function(m1,m2) {
+      return(cor(as.vector(m1),as.vector(m2)))
+      }
>
>
```

See the appendix for some checks on the validity of this computation of Distortion.

Distortion Affected by Sequence Length and OM Costs

Length: Now we can examine how much Distortion is there between the inter-triple distance matrix and the inter-sequence OM distance matrix – controlling

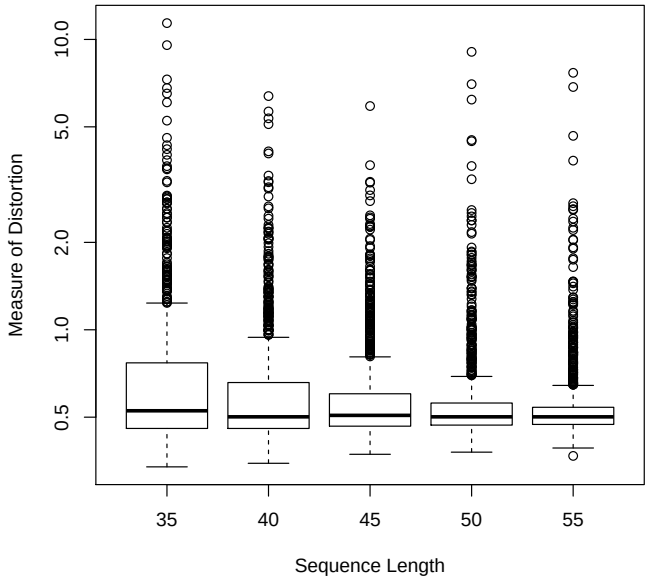


Figure 7

Distortion, by length of sequence

for two important factors, the length of the sequence, and the indel/substitution costs.

It would be plausible to assume that increasing the length of the sequence should have a positive effect on power. As an example, suppose the length of the sequence was 40. If one of the loglogistic values, by chance was, say, 42, the resulting sequence would appear to be identical to a sequence generated if the specific value was 45 instead of 42.

```
> generateSequence(triple=c(15,42,30),seqLength=40)

 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5
[39] 5 5
```

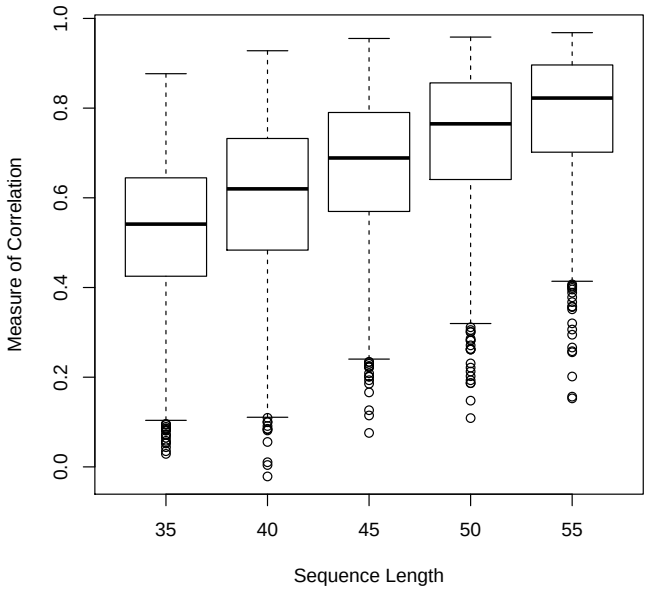


Figure 8

Correlation by length of sequence

```
> generateSequence(triple=c(15,45,30),seqLength=40)

 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5
[39] 5 5
```

However, if the length of the sequence is, say, 50, if triple A contains a value 42, and triple B contains a value 45, the sequence derived from triple A could be observed to be different that one from triple B.

```
> generateSequence(triple=c(15,42,30),seqLength=50)

 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5
[39] 5 5 5 7 7 7 7 7 7 7 7 7

> generateSequence(triple=c(15,45,30),seqLength=50)

 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5
[39] 5 5 5 5 5 5 7 7 7 7 7 7
```

Increasing the length of the sequence, by increasing the ability to differentiate between triples, should increase the accuracy of the representation.

Figure 7 shows that Distortion slightly decreases with length of sequence (higher Distortion means worse fit). Effect is even more clear in Figure 8, that uses the (Cophenetic) correlation measure (lowest correlation means worse fit).

OM Costs: There have been debates in the sociological literature on OM distances for the study of sequences, about how indel/substitution costs are, to a certain extent, arbitrary, and what good rules of judgement to use (Levine, 2000; Wu, 2000; Gauthier et al., 2009; Abbott & Tsay, 2000; Macindoe & Abbott, 2003; Lesnard, 2010).

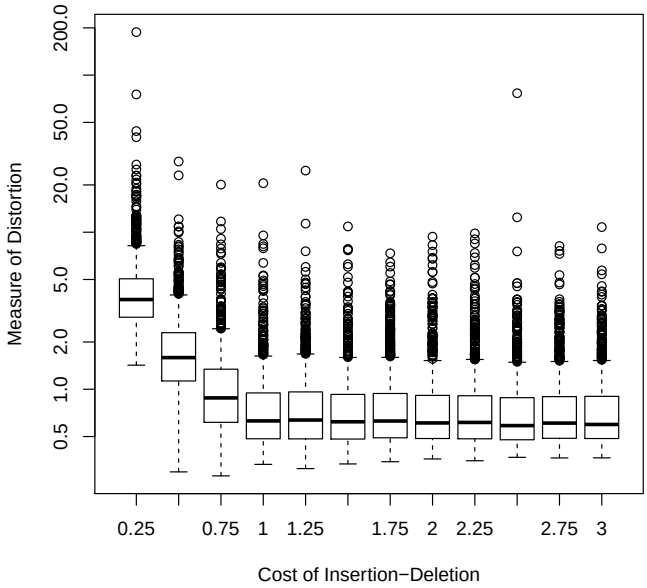


Figure 9

Distortion, changing indel cost, fixed subst=2

Here it is assumed that the matrix of substitutions is a constant 2 for all of them (instead of some other plausible rules, like substitution costs same as transition probabilities), and that the indel costs will range from 0.1 to 2.5.

See Figure 9 for a plot on the effect of varying the Insertion–Deletion cost (and keeping substitution costs constant = 2) on the Distortion measure between true distances (distances between original triplets) and OM distances among derived sequences. Interestingly, indel costs don't seem to affect Distortion – provided that indel is greater than 1/2 of substitution costs.

GEOMETRIC PROPERTIES OF DISTANCES

Practicioners in Ecology (Legendre & Legendre, 1998) have developed many measures of similarity and dissimilarity that are substantively relevant; however the statistical and distributional consequences of so many alternative measures present a challenge for the researcher that need to make inferences. As an example, analytical definition of useful distributions for semimetric distances like the Bray-Curtis is an open question.

In this particular sequence generator, distances among original triples are, of course, embedded in a space of three dimensions.

Two questions are posed: first, do the OM distances between sequences define a metric? Second, are they embedded in an euclidean space?

OM as Metric

An outcome matrix of distances is semimetric or pseudometric when it fulfills three conditions (Legendre & Legendre, 1998):

minimum 0 : if sequence S_i is the same as sequence S_j, then distance $d(S_i, S_j) = 0$;

positiveness : if $S_i \neq S_j$, $d(i, j) > 0$;

symmetry : $d(i, j) = d(j, i)$.

Furthermore, a distance is *metric* when, in addition to the three previous conditions, it fullfills the **triangle inequality** condition
$d(S_i, S_j) + d(S_j, S_k) \geq d(S_i, S_k)$.

The first condition needs no further explanation. If costs are defined for all substitutions, insertions and deletions, there will always be a way to transform a sequence into any other sequence, at a cost, so that cost will be defined. As for the symmetry condition, it will be fulfilled if the matrix of substitution costs is symmetric. In our generators, such matrix is assumed to be symmetric.

The triangle inequality condition can be easily proven true by considering that, in order to transform sequence A to sequence C, it can be done by transforming A to B, and then taking sequence B and transforming it to C. Such transformation, via B, will have a cost $d_{(B)}(A, C) = d(A, B) + d(B, C)$, that defines an upper bound for candidates for the true OM distance $d(A, C)$.

$$d(A, B) + d(B, C) = d_{(B)}(A, C) \geq d(A, C)$$

Metricity of Transformed Distances Notice how, if distances were to be squared, the triangle inequality is in doubt:

$$d^2_{(B)}(A, C) = (d(A, B) + d(B, C))^2 = d^2(A, B) + d^2(B, C) + 2d(A, B)d(B, C)$$

Because of the $2d(A, B)d(B, C)$, we cannot prove that there is a $d^2(A, C)$ such that $d^2(A, C) \leq d^2(A, B) + d^2(B, C)$.

On the other hand, if we take the square root of distances, distances are still metric. To prove that $\sqrt{d(AC)} \leq \sqrt{d(AB)} + \sqrt{d(BC)}$ by contradiction, assume that:

$$\sqrt{d(AC)} > \sqrt{d(AB)} + \sqrt{d(BC)}$$

by squaring each side,

$$d(AC) > d(AB) + d(BC) + 2\sqrt{d(AB)d(BC)}$$

However we established before that true OM distance
$d(AC) \leq d_{(B)}(AC) = d(AB) + d(BC)$, so we have arrived to a contradictory statement.

A square root transformation is just a case of a more general kind of metric preserving transformations. As an example, (Gower & Legendre, 1986, theorem 2, page 7), that says "If D is metric then so are the matrices with elements (i) $d_{ij} + c^2$, (ii) $d_{ij}^{1/r}$ (where $r \geq 1$) (iii) $d_{ij}/(d_{ij} + c^2)$ where c is any real constant".

OM as City Block Distance

Suppose that the sequences of interest, to transform one to the other, are *SOCIOLOGY* to *PSYCHOLOGY*.

Given the alphabet of states {S, O, C, I, L, G, Y, P, S, H, ∅}, (10 letters plus an ∅ as an "empty" state) and assuming that deleting X is just going from a state X to a state ∅, and vice versa for an insertion, a transformation from one sequence to another can be seen as a specific set of elementary exchanges of one state by other state, among the possible $(10 + 1) \times 10/2 = 55$ exchanges possible.

In this case, let us say that the transformation was very simple, exchanged

P$/\emptyset, Y/O, H/I$, like this:

$$\begin{array}{cccccccccc} P & S & Y & C & H & O & L & O & G & Y \\ \emptyset & S & O & C & I & O & L & O & G & Y \end{array}$$

Every one of those 55 possible exchanges define a dimension of distance. Of all those 55 possible exchanges, only 3 were used.

In general, the cost of a transformation from sequences S to S' is a sum of penalties incurred by a set of elementary exchange operations:

$$d(S, S') = \sum_{i=1}^{n+1} \sum_{j=i}^{n+1} w(a_i, a_j)\epsilon(a_i, a_j)$$

Where i and j are counters across the alphabet (of size $n+1$), $w(a_i, a_j)$ is the penalty cost of an exchange between state a_i and state a_j of the alphabet, and $\epsilon(a_i, a_j)$ is the number of times that such exchange happened.

This formulation has the same structure than the city-bloc metric distance (Krause, 1975). How could this be useful? We will see in next subsection that it could be useful to find a better low-dimensional Euclidean approximation via MDS.

Euclideanity of OM and Euclidean Transformations

Raw and Transformed OM as Euclidean There are a number of different questions on Euclidean-ity. A first question is about the intrinsic Euclidean form of the (transformed) OM measure itself.

From a previous subsection, it is evident that the OM distance itself is

city-block (ℓ_1) rather than Euclidean (ℓ_2)[3], since the total cost of transforming a sequence onto another is just a sum of costs of elementary operations.

$$OM_{\ell_1} = \min \sum_{i=1}^{m} w_i \epsilon_i$$

where w_i is the penalty for elementary operation i and ϵ_i is the number of times such operation has been done. Gower & Legendre (1986, page 8) describe the necessary and sufficient Euclidean-ity conditions for **D**:

> **Theorem 4.** **D** is Euclidean iff the matrix (**I - 1s'**)Δ(**I - 1s'**) is positive-semi-definite (p.s.d) where **s'1**=1.

where **I** is a unit matrix, **1** a vector of units and Δ is a matrix with elements $-\frac{1}{2}d_{ij}^2$.

To see if that matrices of OM distances (assume them ℓ_1) are always embeddable in a Euclidean space, we could simulate distance matrices and try to find non-euclidean ones – furthermore, we could estimate what proportion of simulated matrices violate Euclidean-ity (and if factors like relationship between substitution costs and indel costs alter such proportion).

After running simulations, we find that not a single generated OM distance matrix is Euclidean. However, if we take the square root of such matrix values, an interesting relation with the indel cost is observed.

[3]However, a total cost could be defined differently, as a measure in Euclidean space,

$$OM_{\ell_2} = \min \sqrt{\sum_{i=1}^{m} (w_i \epsilon_i)^2}$$

At this point I do not have an algorithm that can compute such distance.

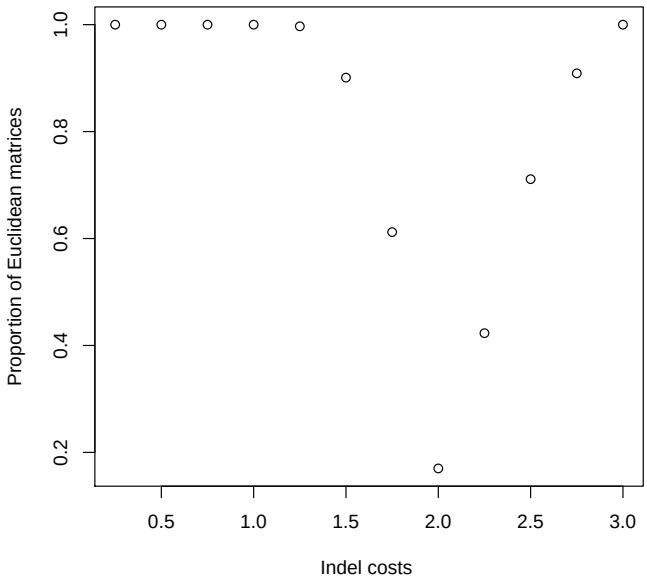


Figure 10

Proportion of Euclidean distance matrices

Figure 10 shows, interestingly, that varying the indel cost has a clear effect: given a matrix of equal substitution costs, when the insertion–deletion cost is between half and double that substitution cost, the proportion of OM matrices that are euclidean drops.

So, an indel of 1 (or, in general, half the substitution cost) seems to offer the goldilocks spot: minimum distortion, but still 100% euclidean if squared root was to be taken from distance.

The literature in MDS has some relevant results: Gower & Legendre (1986, theorem 7), very close to theorem 2, poses the existence of a constant h such that the matrix defined by $(d_{ij}^2 + h)^{1/2}$ is Euclidean; also, there will be a constant k such that $d_{ij} + k$. Constants h and k are functions of the eigenvalue structure of distance matrices. Transformations like these have been used to recast the matrix of distances in a lower dimensional Euclidean space.

Low Dimensional Euclidean via MDS The goal of Multidimensional Scaling is to find low dimensional coordinates that keep as much information as posible, on a raw matrix of dissimilarities.

A low dimensional Euclidean solution would be of great interest, since the coordinates could be used as variables in regression models.

However, since a MDS solution is a new transformation on top of the OM distances (already a transformation and distortion of original triples), we should how much distortion or stress is again introduced. [4]

[4] In finding the MDS solution it is possible to incorporate the knowledge that our distances were city-block distances: in a Minkowski formulation, $d_{ij} = (\sum_{i=1}^{m}(w_i\epsilon_i)^p)^{1/p}$, we should expect our MDS solution to have the least stress when $p = 1$ (city-bloc metric). However, in our simulations, distortion was the same when MDS solution was calculated by `R` (`vegan` package) as `metaMDS(<OMmatrix>,`

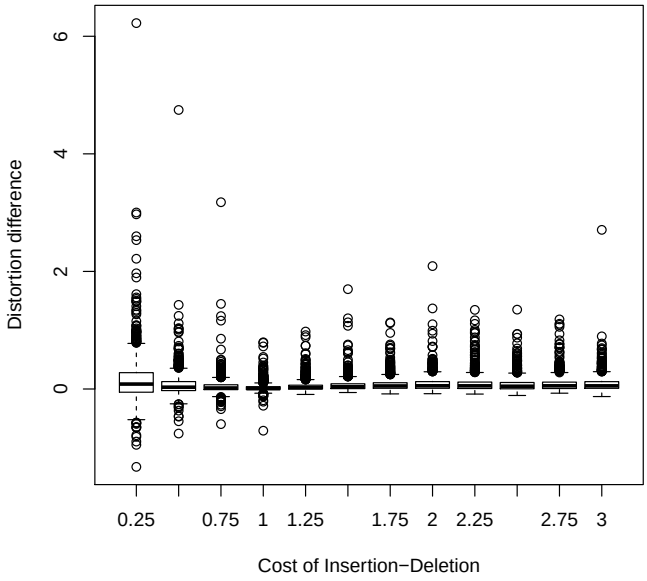


Figure 11

Comparing MDS and OM distortions vs true inter-triple distances

Could it be possible that a MDS procedure helps retrieve deep structure? If the fit between the original inter-triple distances and a MDS transformation of the OM distances is better than the fit between the original inter-triple and the OM distances, then MDS retrieves deep structure.

```
> summary(Distortion.diff)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
-1.324000 -0.000802  0.038370  0.078090  0.108100  6.224000         1
```

`distance='manhattan')` than `metaMDS(<OMmatrix>, distance='euclidean')`

Figure 11 and summary shows that the difference in Distortion is a bit positive – most of the time, the OM representation is closer to the true inter-triple space than the MDS solution is. The few cases where the MDS solution was an improvement happened when the indel cost was very low and the OM solution was already ill fitting (as seen in Figure 9).

The issue of Euclideanity is important because, according to their authors, some tests depend on that assumption, and the robustness of similar tests is an active area of discussion (McArdle & Anderson, 2001; Anderson, 2001), as discussion on how to transform the data (Legendre & Anderson, 1999). Mantel's test does not depend on Euclideanity, Permanova neither; MRPP does according to Mielke & Berry (2001), as did Good (1982).

DISTRIBUTION OF INTER-SEQUENCE DISTANCES

What is the distribution of sequences in a space defined by the Optimal Matching dissimilarity among them? This question can be split in two questions: one is about the *unidimensional* distribution of the inter-sequence OM distance $d(s_i, s_j) = d_{ij}$. The second is about the distribution in a *multidimensional* space of randomly generated sequences, as specified by a matrix of OM distances. For now we can only explore the first question.

Distribution of True Inter-Triple and OM Distances

Figure 12 shows the distribution of true inter-triple distances among 100 sequences ($(100 \times 99)/2 = 4950$ points in total); graph is censored – there are a few distances with very large values. Figure 13 shows the distribution of OM distances.

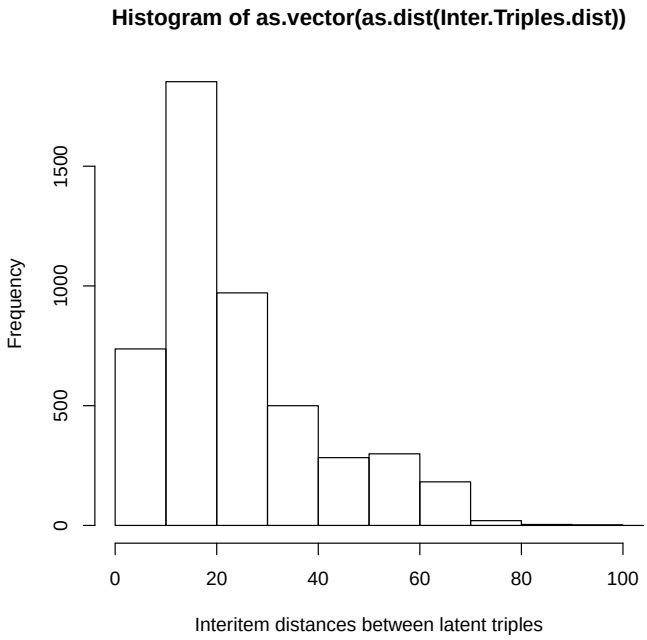


Figure 12

Lengths of true inter-triple distances

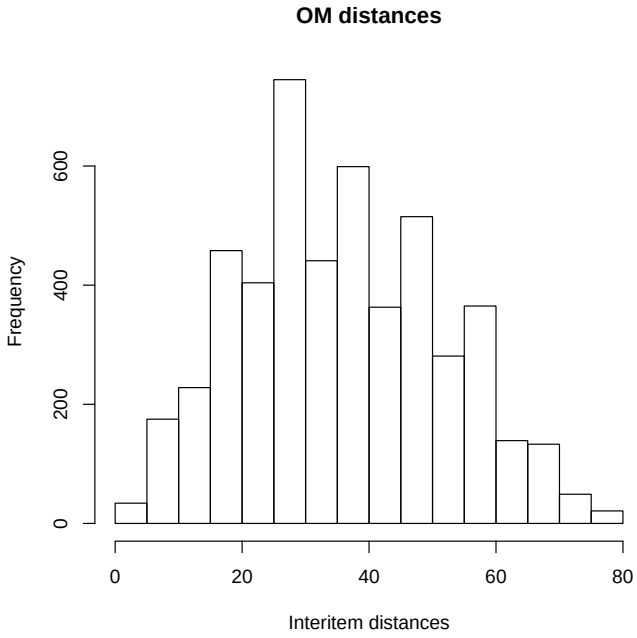


Figure 13

Lengths of OM distances

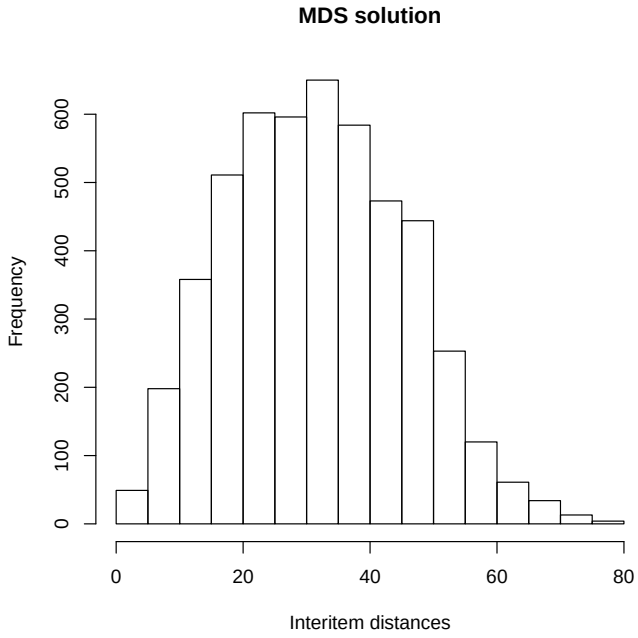


Figure 14

Lengths of MDS solution to OM distances

The difference in shapes is very noticeable.

Figure 14 shows the distribution of inter-item distances from an MDS transformation of OM data.

Multinormal Baseline

As a baseline of comparison, if the items are distributed in a space according to a Multivariate Normal (with Variance/Covariance matrix Σ, size $p \times p$, diagonal), a plot of their inter-item distances would look as in Figure 15.

Could an empirical sample of distances be used to assert if the distribution of

```
> hist(d,freq=FALSE,xlab="Interitem distances, items are Multinormal")
```

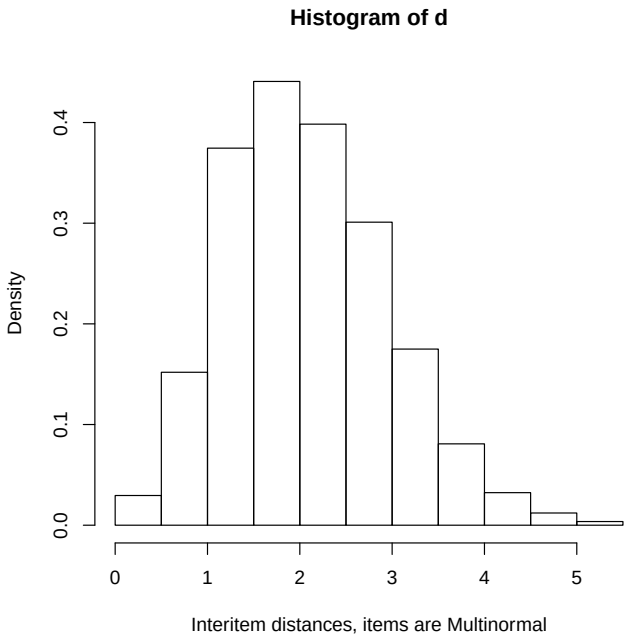


Figure 15

Lengths of inter-item distances, multinormal distribution

inter-sequence OM distances or the MDS solution are roughly equivalent to the one produced by a multinormal process? It is encouraging that the distribution of distances produced by the MDS solution is quite close to the multinormal solution. However we realize that the most important component is not what transformation of the OM matrix would be the least distorted, of if it is the closest to a well known distribution, but what transformation would increase the power of tests.

Chapter III

STATISTICAL TESTS FOR INFERENCE

Suppose that the sample of sequences was not from a homogeneous peopulation, but from a heterogeneous population. For example, some of the educational/job trajectories correspond to women, some to men. If gender does affect their trajectory, could the statistic pick that difference as significant (from the raamdom noise that may be present)? A number of tests will be compared for power; I will run simulations following Studer et al. (2011), evaluating

- type I error (probability that, in many repetitions of the test, a statistic will result in values that appear to be inconsistent with the Null Hypothesis of no group diference, *given that there is no such real difference*; specifically it will be a proportion of p-values that are less than a 0.05 threshold, given that the null hypothesis is true (no difference among subjects by partition).

- statistical power (proportion of times that, in many repetitions of the test, a statistic will result in values that appear inconsistent with the null hypothesis of no group difference, *given that there is a real difference between groups*; implemented as the proportion of p-values that are less than a 0.05 threshold, given that the alternative hypothesis is true).

MANOVA on Principal Coordinates

In a previous section we saw how an MDS transformation could be used to obtain a Euclidean version of the OM distance matrix, minimizing distortion and approaching the distribution of interitem distances by a multinormal distribution. Given a matrix of distances (size $n \times n$), MDS procedures (principal coordinates being another name for metric MDS) can be used to obtain a number of coordinates for every sequence. Such coordinates will be represented in a $n \times k$ matrix, where k is the number of dimensions that was considered to be enough to embody the variability in the sample; such coordinates can be used as outcome variables, and explained by categorical explanatory variables, in a MANOVA setting.

Mantel Test

The Mantel test (Mantel, 1967) was initially developed to test for association between spatial clustering and temporal clustering in cancer cases; in its most basic form, it uses two square matrices of dissimilarities (or similarities) defined on the same items – the two matrices are, of course, the same size (n by n), n being the number of subjects.

In our case, the matrix ($\mathbf{Y}$) can be the matrix of pairwise distances between sequences; the second matrix ($\mathbf{X}$) can be a matrix of differences between subjects – the subjects from which the sequences are derived. For instance, it can be a matrix of absolute age differences, or a matrix of distances generated by some other trajectory of interest in a state space.

The null hypothesis is that the cells in $\mathbf{X}$ are not linearly correlated with the

corresponding cells in $\mathbf{Y}$.

Suppose that they were, say, positively correlated. Then the value of Mantel's statistic

$$z_M = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} y_{ij}$$

(the sum of the Hadamard product on the lower half of the matrices, not including diagonals) will be high.

For testing differences between two groups, $\mathbf{X}$ can be defined in two equivalent ways:

- as $x_{ij} = 0$ if sequences i and j belong to the same group, $x_{ij} = 1$ if they belong to different groups. This coding corresponds to x_{ij} as an intuitive measure of distance – if two objects are in the same group, their distance is 0. As a consequence,

  $$z_M = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} y_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} y_{ij} \epsilon_{i,j}$$

  where indicator function $\epsilon_{i,j} = 0$ if i, j belong to same group, $\epsilon_{i,j} = 1$ if they belong to different groups. z_M then can be defined as the sum of mutual distances between sequences that do not belong to the same group – a between-group distance. The alternative hypothesis of clustering by group would imply that the between-group measure is large. (`vec1` is a vector of covariate values for every sequence).

```
> vec1

 [1] 0 0 0 0 0 1 1 1 1 1

> X = 1 * outer(vec1,vec1,FUN="!=")
> X

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    0    1    1    1    1     1
 [2,]    0    0    0    0    0    1    1    1    1     1
 [3,]    0    0    0    0    0    1    1    1    1     1
 [4,]    0    0    0    0    0    1    1    1    1     1
 [5,]    0    0    0    0    0    1    1    1    1     1
 [6,]    1    1    1    1    1    0    0    0    0     0
 [7,]    1    1    1    1    1    0    0    0    0     0
 [8,]    1    1    1    1    1    0    0    0    0     0
 [9,]    1    1    1    1    1    0    0    0    0     0
[10,]    1    1    1    1    1    0    0    0    0     0



> X * d.example

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    0   64   60   44   38    48
 [2,]    0    0    0    0    0  140   80   96  140   140
 [3,]    0    0    0    0    0   72   60   76   78   116
 [4,]    0    0    0    0    0   46  112  112   70    94
 [5,]    0    0    0    0    0  140   90   96  140   140
 [6,]   64  140   72   46  140    0    0    0    0     0
 [7,]   60   80   60  112   90    0    0    0    0     0
 [8,]   44   96   76  112   96    0    0    0    0     0
 [9,]   38  140   78   70  140    0    0    0    0     0
[10,]   48  140  116   94  140    0    0    0    0     0

> sum(X * d.example)/2 # usually sum over lower triangle

[1] 2292
```

- Conversely, we can define $x_{ij} = \epsilon_{i,j} = 1$ if they belong to the same group, and $\epsilon_{i,j} = 0$ if they belong to different groups. This corresponds to $\mathbf{X}$ as a similarity matrix, and to define z_M as a sum of within-group distances. The

alternative hypothesis of clustering by group would imply that the within-group distances is small.

```
> X = 1*outer(vec1,vec1,FUN="==")
> X

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    1    1    1    1    0    0    0    0     0
 [2,]    1    1    1    1    1    0    0    0    0     0
 [3,]    1    1    1    1    1    0    0    0    0     0
 [4,]    1    1    1    1    1    0    0    0    0     0
 [5,]    1    1    1    1    1    0    0    0    0     0
 [6,]    0    0    0    0    0    1    1    1    1     1
 [7,]    0    0    0    0    0    1    1    1    1     1
 [8,]    0    0    0    0    0    1    1    1    1     1
 [9,]    0    0    0    0    0    1    1    1    1     1
[10,]    0    0    0    0    0    1    1    1    1     1

> X * d.example

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0  140  116  108  140    0    0    0    0     0
 [2,]  140    0   72  140   22    0    0    0    0     0
 [3,]  116   72    0   68   90    0    0    0    0     0
 [4,]  108  140   68    0  140    0    0    0    0     0
 [5,]  140   22   90  140    0    0    0    0    0     0
 [6,]    0    0    0    0    0    0   68   68   26    66
 [7,]    0    0    0    0    0   68    0   16   60    60
 [8,]    0    0    0    0    0   68   16    0   44    48
 [9,]    0    0    0    0    0   26   60   44    0    48
[10,]    0    0    0    0    0   66   60   48   48     0

> sum(X * d.example)/2

[1] 1540
```

Small or large compared to what? Ideally, we should contrast it with a distribution of possible values of the z_M statistic derived from matrices of distances from two sets of sequences generated by the same mechanism (in other words, from the same population). However, since we don't know such mechanism and cannot generate a population of intersequence distances, we have to imagine ways to use

the empirical distance data to imagine a population.

From Legendre & Legendre (1998, p. 554): "According to H_0, the vector of values [distances] observed by any object [sequence] could have been observed by any other object; in other words, the objects are the permutable units. [...] An equivalent result is obtained by permuting at random the rows of matrix [**X**] and the corresponding columns."

```
> vec1

 [1] 0 0 0 0 0 1 1 1 1 1

> permuted.vector = sample(vec1,replace=FALSE)
> permuted.vector

 [1] 1 0 0 0 1 1 1 1 0 0
```

Now the matrix of permuted group affiliations

```
> X = 1*outer(permuted.vector,permuted.vector,FUN="==")
> X

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    0    0    0    1    1    1    1    0     0
 [2,]    0    1    1    1    0    0    0    0    1     1
 [3,]    0    1    1    1    0    0    0    0    1     1
 [4,]    0    1    1    1    0    0    0    0    1     1
 [5,]    1    0    0    0    1    1    1    1    0     0
 [6,]    1    0    0    0    1    1    1    1    0     0
 [7,]    1    0    0    0    1    1    1    1    0     0
 [8,]    1    0    0    0    1    1    1    1    0     0
 [9,]    0    1    1    1    0    0    0    0    1     1
[10,]    0    1    1    1    0    0    0    0    1     1

> X * d.example

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0  140   64   60   44    0     0
 [2,]    0    0   72  140    0    0    0    0  140   140
 [3,]    0   72    0   68    0    0    0    0   78   116
 [4,]    0  140   68    0    0    0    0    0   70    94
 [5,]  140    0    0    0    0  140   90   96    0     0
 [6,]   64    0    0    0  140    0   68   68    0     0
 [7,]   60    0    0    0   90   68    0   16    0     0
 [8,]   44    0    0    0   96   68   16    0    0     0
 [9,]    0  140   78   70    0    0    0    0    0    48
[10,]    0  140  116   94    0    0    0    0   48     0

> sum(X * d.example)/2

[1] 1752
```

This process could be repeated for all possible permuted assignments of individuals to groups ($n! = 3,628,800$), or at least for a large enough sample.

Permutational MANOVA

The following graph from Anderson (2001) describes the logic of Anova testing – if we assume that responses can be in a multidimensional space (and not just in an unidimensional space).

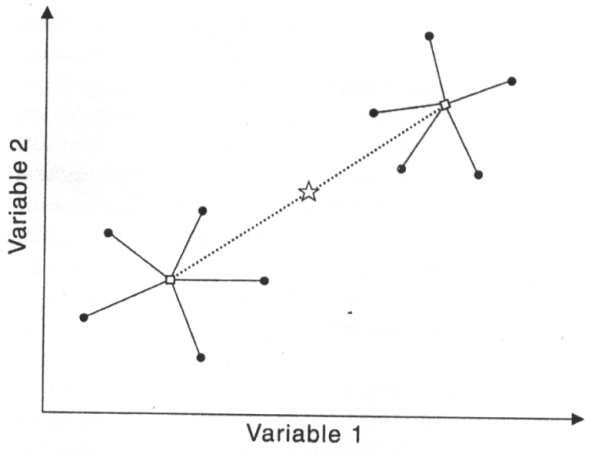


Fig. 1. A geometric representation of MANOVA for two groups in two dimensions where the groups differ in location. The within-group sum of squares is the sum of squared distances from individual replicates to their group centroid. The among-group sum of squares is the sum of squared distances from group centroids to the overall centroid. (——) Distances from points to group centroids; (·····) distances from group centroids to overall centroid; (☆), overall centroid; (□), group centroid; (●), individual observation. (Anderson 2001)

In our case or sequence comparison, however, our raw data consists of interpoint distances. Furthermore, a centroid of sequences ("average") cannot be easily imagined or defined (same as the average of a set of integer numbers may not be an integer itself).

To solve these difficulties, and if one assumes that distances are metric, there is a family of approaches that start from the geometrical consideration that variance and distance are linked by

$$\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}\sum_{i<j}^{n}(x_i - x_j)^2$$

The right hand is "a sum of the $\binom{n}{2}$ pairwise distances among [n points x_i]" (Gower & Krzanowski, 1999).

So it may be possible to reformulate the Anova relationship based on interpoint distances. I quote from Pillar & Orloci (1996):

> "test criterion is the sum of squared dissimilarity between groups: $Q_b = Q_t - Q_w$".
>
> $$Q_t = \frac{1}{n}\sum_{h=1}^{n-1}\sum_{i=h+1}^{n} d_{hi}^2$$
>
> "which is the total sum of squares involving $n(n-1)/2$ pairwise squared distances (d_{hi}^2) between n [objects]. The Q_w term is given by
>
> $$Q_w = \sum_{c=1}^{k} Q_{wc}$$
>
> This is the sum of squares within the k groups. The terms in Q_w accord with
>
> $$Q_{wc} = \frac{1}{n_c}\sum_{h=1}^{n-1}\sum_{i=h+1}^{n} d_{hi}^2\delta(h,i,c)$$
>
> which is the sum of squares within group c with size n_c. The indicator variable $\delta(h,i,c)$ is one if [objects] h and i belong to group c or zero if

otherwise.

How do we evaluate the significance of an F ratio in a multidimensional distance setting? Under the null hypothesis the allocation of objects to groups would not matter, so objects could be randomly allocated or permuted to groups, and a distribution of pseudo-F values computed.

The problem of evaluating the significance of a pseudo$F$ to distinguish among groups is different from the problem of partitioning variability – defined as "attributing additive proportions of the total variability to individual factors in an experimental design" (McArdle & Anderson, 2001) when the measure of distance is not metric. Statistics for the first problem are already well known (Mantel, 1967; Good, 1982; Dietz, 1983; Clarke, 1993) and as we will show, mostly equivalent.

On the other hand, the issue of partitioning, particularly in multifactorial models, is not completely understood. When decomposing a semimetric distance matrix, by means of principal coordinate analysis, into a linear set of Euclidean coordinates, the result may contain imaginary Euclidean coordinates or negative eigenvalues, of unclear interpretation. McArdle & Anderson (2001); Anderson (2001); Reiss et al. (2010) propose to avoid any correction (advocated by Legendre & Anderson (1999)) and work directly with the data.

Permanova's statistic, in the simplest balanced design (same number n of

items per group) is

$$\begin{aligned} F &= \frac{SS_A/(a-1)}{SS_W/(N-a)} \text{ where } SS_A = SS_T - SS_W, \\ SS_T &= \frac{1}{N}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}^2 \\ SS_W &= \frac{1}{n}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}^2 \epsilon_{ij} \end{aligned}$$

where $\epsilon_{ij} = 1$ if item i and j belong to the same group, 0 otherwise.

The permutational strategy to test for significance of the statistic is the same as for Mantel's.

MRPP

In the original notation of Mielke & Berry, MRPP statistic is

$$\delta = \sum_{i=1}^{g} C_i \xi_i$$

where

$$\xi_i = \binom{n_i}{2}^{-1} \sum_{I<J} \Delta_{I,J} \Psi_i(\omega_I) \Psi_i(\omega_J)$$

i is every one of the groups, from 1 to g. n_i is the number of items in group i, $\binom{n_i}{2}$ divides by the number of symmetric pairs in a group of size n_i. $\sum_{I,J}$ represents a sum of terms over a lower triangle. $\Psi_i(\omega_I) = 1$ if object ω_I belongs to group S_i. So $\Psi_i(\omega_I)\Psi_i(\omega_J)$ is an indicator function with a value of 1 if objects ω_I and ω_J belong to same group S_i. C_i is a weight coefficient that can be defined many ways; Mielke (1985) recommends the proportion of items in group i, $C_i = n_i/N$, and criticizes

other choices implicit in the literature, as inefficient.

$\Delta_{I,J}$ is an dissimilarity defined as

$$d_{I,J} = [\sum_{h=1}^{r}(x_{hI} - x_{hJ})^2]^{v/2}$$

where the value v is left to the theoretical choice of the researcher. If x_{hI}, x_{hJ} are coordinates, and $v = 1$, $\Delta_{I,J}$ is an Euclidean distance. Note how with $v = 2$, $\Delta_{I,J}$ may violate the triangle inequality (as an example in one-dimensional space, if $x_1 = 4, x_2 = 6, x_3 = 7$, then $\Delta_{12} = 4, \Delta_{13} = 9, \Delta_{23} = 1$ and $\Delta_{12} + \Delta_{23} < \Delta_{13}$, violating triangle inequality). Mielke & Berry call for congruence between the data space (Euclidean in this case) and the statistical method's analysis space. Authors note to the existence of counterintuitive results caused by the use of $v = 2$. Mielke & Berry (2001) explored power of tests to detect location shifts, as affected by different values of v, across many unidimensional distributions. In cases where the underlying distribution is not known, authors defend the choice of $v = 1$.

A rewrite of MRPP's statistic, closer to the formulas for Mantel's and Permanova, would be

$$\delta = \sum_{k=1}^{g} \frac{n_k}{N} \frac{2}{n_k(n_k - 1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}\epsilon_{ij}$$

(Mielke & Berry would make group weight $C_k = n_k/K$, where K is the total number of items classified in groups – there may be items not classified, $N - K \geq 0$; however, for simplification, we assume that all items are classified and that $K = N$.)

For MRPP, permutational testing is done in an similar way than Mantel's –

with a caveat: possible allocations of N objects to the $g+1$ groups (the $[g+1]$th group being the excess group or residual category), would be

$$M = \frac{N!}{\prod_{k=1}^{g+1} n_k!}$$

(This is more precise than the $N!$ from Mantel' s description).

Chapter IV

A COMPARISON OF POWER

GENERAL STRATEGY

Simulations will be used to investigate the power of some statistical tests. In each simulation run, two sets of sequences will be generated: the baseline set and the alternate set.

To investigate Type I error, in every simulation run, these two sets are generated from the same mechanism with the same parameter values. In other words, they come from the same population – null hypothesis H_0 is true. Out of many simulations, how often will the test be mistaken? (by the statistic value being in the rejection region, leading to the mistaken interpretation that they would come from different populations.)

To investigate power, in every simulation run, the alternate set and the baseline set are generated from different parameters – each set comes from different populations. Out of many simulations, how often will the test be able to detect that they, in fact, come from two different populations – how often will the value of the statistic be in the rejection region?

For each simulation run, a total of 30 sequences will be generated; 15 sequences in the baseline set, and 15 in the alternate set.

At every one of the 2000 simulation runs, the tests (non-parametric like

Mantel and Permutational MANOVA, parametric like tests of means and medians) will be applied on the same pair of sets (baseline and alternate).

The process will be repeated, across a range of progressively more distant values of alternate parameters (from baseline parameters).

Making Use of Parametric Information

The focus of this research is on a number of non-parametric tests that do not assume previous knowledge about how sequences are generated. However, it may be interesting to compare the performance of such non-parametric tests with the performance of tests that made use of our knowledge about the stochastic generator of the sequences.

Suppose that we knew much more about the generating mechanism. In this case, going back to Chapter II, notice that the parameters that generate the first and third preliminary sequences are fixed, and we only change the value of parameters for the mechanism acting on the second sequence.

If we only observed resulting data, but knew that the key piece of information was the change in t_2, we could identify the footprint of that change, by realizing that when a state switched from (Z,0,Z) to (Z,1,Z), the inflection point has happened.

We know that the switch has happened in the final sequence, when one of these listed sub-sequences shows up. In parenthesis we see the triple representation of the digits. Notice how the central 0 switches to 1.

- sub-sequence ...02... (state 000 became 010),
- ...03... (000 $\rightarrow$ 011),

- ...06... (000 → 110),
- ...07... (000 → 111),
- ...13... (001 → 011),
- ...17... (001 → 111),
- ...46... (100 → 110),
- ...47... (100 → 111),
- ...57... (101 → 111).

For each sequence we can get a number that indicates where any such pair shows up in the sequence. Suppose the sequence was 00000022, the value of interest will be 7, since 7th is the place where the transition shows up (the place of the "2" after "0"). For each sequence, let us call this value its observed transition value.

A set of n sequences, be the baseline or the alternate, would provide a vector (size n) of observed transition values. If the sub-sequence never happens in a sequence, because the randomly generated value for t_2 was larger than the sequence length, we can consider that the transition value is missing.

The vector of the transition values from baseline set can be compared with the vector of the transition values from the alternate set; two straightforward tests that come to mind are a 2-sample t tests, and a permutational test of difference of medians.

It seems reasonable to expect that these two tests will be more powerful than the Mantel, Permanova or MRPP tests, since they incorporate more knowledge about how the sequences were generated and what footprint to look for.

SIMULATION

Optimal matching needs a matrix of penalty costs for the transition between state i to j. (Note that for our particular generator, some values in the transition matrix will be of no use, since some transitions cannot occur, like 011 to 010: once a member of a triple becomes 1, it stays that way, it cannot go back to a 0 value. However, since we ignore that fact, all substitution costs are assumed to be constant.)

```
> cost.matrix

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0    2    2    2    2    2    2    2
[2,]    2    0    2    2    2    2    2    2
[3,]    2    2    0    2    2    2    2    2
[4,]    2    2    2    0    2    2    2    2
[5,]    2    2    2    2    0    2    2    2
[6,]    2    2    2    2    2    0    2    2
[7,]    2    2    2    2    2    2    0    2
[8,]    2    2    2    2    2    2    2    0
```

R packages were used: `ape` (Paradis, Claude, & Strimmer, 2004), `ecodist` (Goslee & Urban, 2007), `vegan` (Oksanen et al., 2013).

Changing Start Parameter, Fixing Rate

In this mechanism defined by a multiple log-logistic model, we'll be comparing reference sequences generated by

$$t_1 = 0 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$
$$t_2 = 10 + LL(\text{Rate} = 0.126, \text{Shape} = 2.364)$$
$$t_3 = 20 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$

with sequences generated by

$$t_1 = 0 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$
$$t_2 = a_2 + LL(\text{Rate} = 0.126, \text{Shape} = 2.364)$$
$$t_3 = 20 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$

where the Start parameter a_2 increases from 10 (Null Hypothesis is true) to 15.

Figure 16 shows the power function of various statistics. Power function is defined as the probability of the statistic being in the rejection region, given data generated from a mechanism with such parameter value (Start in this case).

Baseline set is generated with Start=10, alternative sets are generated from increasing values of Start. At first, the two sets are both generated from Start=10 – they come from the same population. Probability of rejecting the null hypothesis, given that Start=10, is the Type I error. We see how the lines are concentrated a bit over 0, perhaps around 0.05 – this is in agreement with a tolerable type I error.

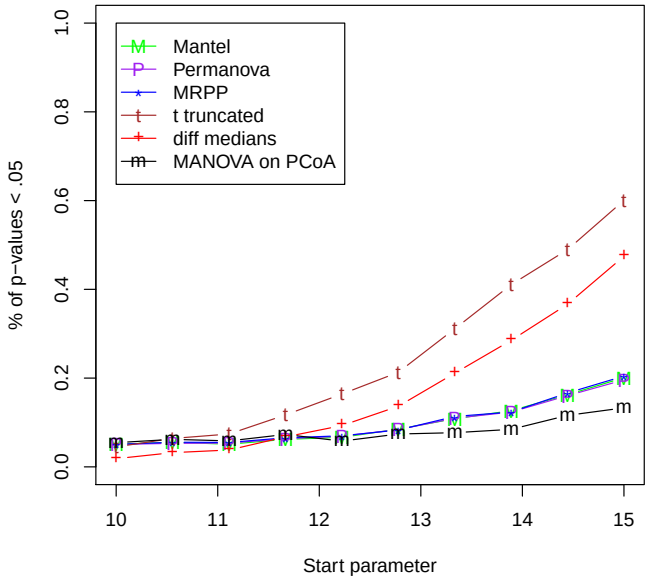


Figure 16

Power of tests, by values of Start parameter (Rate fixed)

As the alternate sets are generated by more divergent Start values, the power to detect that they are not from the same baseline population increases – as expeted. However we see how the non-parametric tests are less powerful than the ones that make use of parametric information.

Figure 16 shows that the parametric tests (t truncated, and difference of means, based on inside knowledge about the parameter of interest and footprint to follow to identify differing sequences) have the most power. The non-parametric permutational tests have less power. The MANOVA test done on a euclidean transformation of the OM solution has the least power of the compared tests.

Changing Rate (Keeping Start Fixed)

Now we are comparing reference sequences generated by

$$t_1 = 0 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$
$$t_2 = 10 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$
$$t_3 = 20 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$

with sequences generated by

$$t_1 = 0 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$
$$t_2 = 10 + LL(\text{Rate} = \lambda_2, \text{Shape} = 2.364)$$
$$t_3 = 20 + LL(\text{Rate} = 0.078, \text{Shape} = 2.364)$$

where $0.078 \leq \lambda_2 \leq 0.205$.

The inverse of the Rate (aka Intensity) parameter in a log-logistic distribution is called the Scale parameter; such Scale parameter happens to be the median of the distribution. So a natural estimator for Rate is the inverse of the median of a sample.

However, if we are not really dealing with a continuous log-logistic distribution, but with a discretized log-logistic (since we do not observe the actual random quantity T, but the integer X that is equal or greater than T) the distribution of medians of discretized random quantities may not have enough granularity.

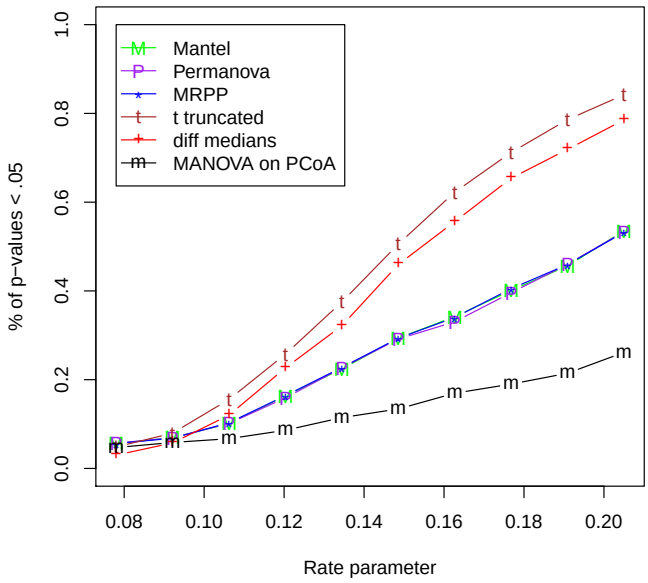


Figure 17

Power of tests, by values of Rate parameter (Start fixed)

Same with previous comparison, Figure 17 shows that the parametric tests (t truncated, and difference of means, based on inside knowledge) have the most power. The non-parametric permutational tests have less power. The MANOVA test done on a euclidean transformation of the OM solution has the least power of the compared tests.

The Effect of Length of Sequence in Power of Tests

We saw in subsection (page 63) that distorion decreases as sequence length increases, as expected.

Could we expect that power increases with sequence length? In the simulated sequences, half come from a baseline population where Start=10, half come from a popuation where Start=15; Rate parameter is fixed.

Figure 18 shows that, surprisingly, power seems to go down, the longer the sequences are. To better understand this result, we should review some findings and conclusions:

- Should length of strings (sequences) affect raw edit distance? All other things being equal, yes, since a tranformation from one string to another would need more insertions or substitutions.

- Should length of string affect a Mantel-type statistic? no: while it would affect the computed value of the specific statistic, it should not affect the evaluation of significance – the distribution of permuted statistics should still be equivalent. If distance matrix is $[d_{ij}]$, a statistic calculated on a transformed $[d^*_{ij}]$ should not change in power.

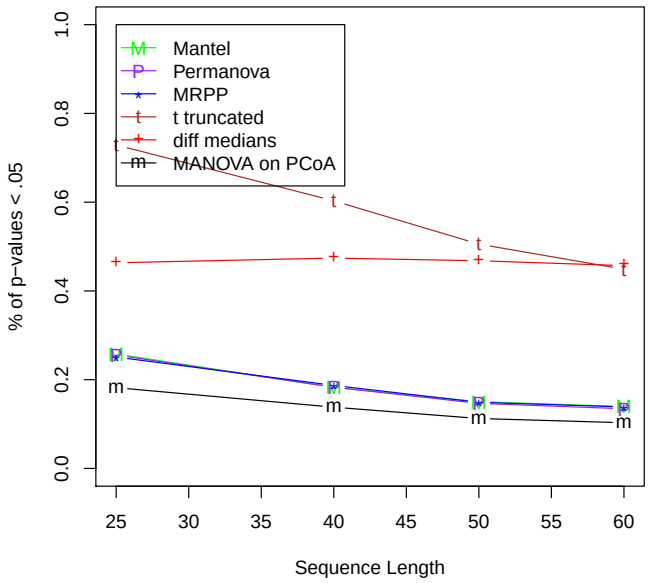


Figure 18

Power of tests, by length of sequences (Start and Rate fixed)

- should length of sequences affect the distortion between $[d_{ij}]$ (the matrix of Levenshtein-like distances among sequences) and $[\delta_{ij}$ (the matrix of true euclidean distances between original triples)? As seen before, Distortion decrease with increased sequence length. With shorter sequence lengths (for example, with length 40) some triples that are truly different (say triples 10,15,42 and 10,15,50) would still produce the same final sequence. If length was 50, the triples would produce different sequences. Differentiation of previous outliers is a way in which increased length lessens distortion.

However, if we take outlier differentiation out of the table, would length still increase power? To verify that, we can write the simulation by generating sequences, but discarding those triples where one or more of the three components is greater than the sequence length. So there would be no outliers in the triples.

Figure 19 shows that power does go down even more with length.

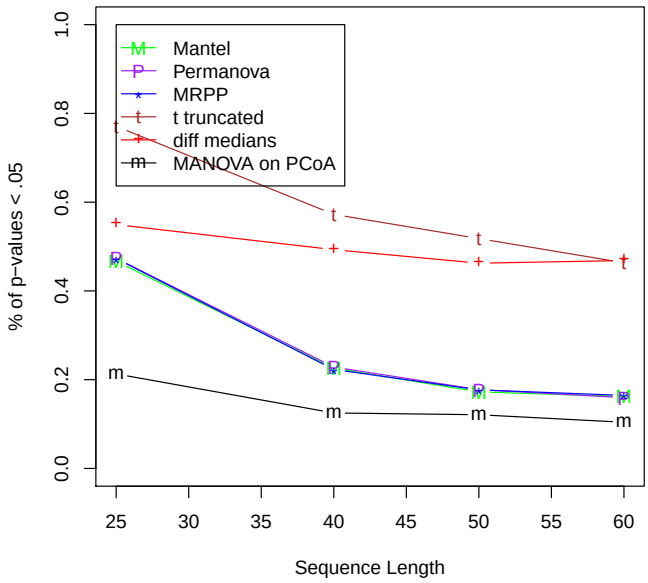


Figure 19

Power by Seq Length, keeping triples truncated

Chapter V

ANALYTICAL IDENTITY BETWEEN MANTEL AND PERMANOVA

Why should Mantel, Permanova and MRPP produce identical results?

Remember how Mantel's statistic in our case of two group comparison was

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}\epsilon_{i,j}$$

where $\epsilon_{i,j}$ is an indicator variable with value 1 if item i and j belong to the same original group, 0 otherwise.

MRPP's statistic was

$$\delta = \sum_{k=1}^{g} C_k \frac{2}{n_k(n_k-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \Delta_{ij}\epsilon_{i,j}$$

where $\epsilon_{i,j}$ is defined as before (1 if both $i, j \in k$ group), and $\Delta_{i,j} = d_{i,j}^{v}$ and usually $v = 1$.

As n_k is a constant in this case, Mantel and MRPP, at least in this very simple case, when $v = 1$, are analytically equivalent. Furthermore, raising d_{ij}^{v} to a $v = 2$, for instance, would make it a monotonous transformation of d_{ij}, with unchanged distributional properties.

For Permanova, statistic of interest was (Anderson, 2001):

$$F = \frac{SS_A/(a-1)}{SS_W/(N-a)} = \frac{SS_A}{SS_W}\frac{N-a}{a-1}$$

The term $\frac{N-a}{a-1}$ (where a represents number of groups and N total number of items) is invariant.

$$\frac{SS_A}{SS_W} = \frac{SS_T - SS_W}{SS_W} = \frac{SS_T}{SS_W} - 1$$

SS_T is invariant over all possible permutations, so statistic is only a function of SS_W.

As

$$SS_W = \frac{1}{n}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}^2 \epsilon_{ij}$$

We have shown that Permanova' s statistic, under these simple conditions, and given the same data, has the same distributional properties as Mantel's and MRPP.

Chapter VI

CONCLUSIONS

Optimal Matching distances have been used to explore categorical sequences; in rcent times a more careful consideration of statistical inference has been attempted Studer et al. (2011).

Given a known distribution of random objects (triples), and given an algorithm that generates sequences from such triples, this study looked at how distorted was the OM representtaion vis a vis the original. We find that OM costs affect the distortion, and that an indel cost valued at half of substitution costs produce the least amount of distortion. We also found that the longer the sequences, the less distorted the OM distances were from the original. We found that the distances were metric, but not Euclidean. Given the OM distances, an Euclidean aproximation could be found by computing the square root of the values, or even better, by finding an MDS solution. MDS solutions also had the nice property of producing a distribution of distances that was quite close to the euclidean distances from items that came from a Multivariate Normal distribution of items.

Not surprisingly, permutational tests like MRPP, Mantel and Permanova were less powerful than tests based on insight about the parametric generation of the sequences. Surprisingly, a MANOVA test on the MDS solution had very weak power. Surprisingly also, power was decreased by an increase in the sequence length.

At his point we do not have a good explanatory mechanism for this result.

Since Permanova (Anderson, 2001) explicitly addresses the issue of multiway analysis of discrepancy, practical researchers are encouraged to follow Dr. Anderson's work and tools. However, researchers need to be aware that the power of these tests is low.

FUTURE WORK

It will be necessary to study if results are robust across sequences derived from other generators. Second, there are two other tests that I discovered very late in the process: *DISCO* (Rizzo & Székely, 2010) and *crossmatch* (Rosenbaum, 2005), based on work by Friedman & Rafsky (1983). This last one is of particular interest because it is based on minimal spanning trees, an area of work very close to Joe Kruskal. [1] These may or may not be as powerful or extendable as Anderson's, so more research is needed to compare their power.

We can expect more theoretical work on the issue of non-euclideanity and how it affects inference, going beyond Gower & Legendre (1986); Legendre & Anderson (1999); McArdle & Anderson (2001).

[1] It is a mistery how Kruskal, who was involved in both MDS and Optimal Matching, never seemed to have combined the two.

REFERENCES

REFERENCES

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

Abbott, A., & Forrest, J. (1986, winter). Optimal matching methods for historical sequences. *Journal of Interdisciplinary History*, *16*(3), 471–494.

Abbott, A., & Hrycak, A. (1990). Measuring resemblence in sequence data: An optimal matching analysis of musicians careers. *American Journal of Sociology*, *96*(1), 144–185.

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

Anderson, M. J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology*, *26*, 32–46.

Blossfeld, H.-P., & Rohwer, G. (2002). *Techniques of event history modeling. New approaches to causal analysis* (2nd ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

Borg, I., & Groenen, P. J. F. (2005). *Modern multidimensional scaling: Theory and applications* (2nd ed.). New York: Springer.

Clarke, K. R. (1993). Non-parametric multivariate analyses of changes in community structure. *Australian journal of Ecology*, *18*, 117-143.

Dietz, E. J. (1983, March). Permutation tests for association between two distance matrices. *Systematic Zoology*, *32*(1), 21–26.

Diggle, P. J., Liang, K.-Y., & Zeger, S. L. (1994). *Analysis of longitudinal data*. Oxford (England): Oxford University Press.

Dutang, C., Goulet, V., & Pigeon, M. (2008). actuar: An R package for actuarial science. *Journal of Statistical Software*, *25*(7), 38. Retrieved from `http://www.jstatsoft.org/v25/i07`

Elzinga, C. H. (2010). Complexity of categorical time series. *Sociological Methods & Research*, *38*(3), 463–481.

Friedman, J. H., & Rafsky, L. C. (1983). Graph-theoretic measures of multivariate association and prediction. *The Annals of Statistics*, *11*(2), 377–391.

Gabadinho, A., Ritschard, G., Muller, N. S., & Studer, M. (2011, 4 7). Analyzing and visualizing state sequences in R with TraMineR. *Journal of Statistical Software*, *40*(4), 1–37. Retrieved from `http://www.jstatsoft.org/v40/i04`

Gauthier, J.-A., Widmer, E. D., Bucher, P., & Notredame, C. (2009). How much does it cost? optimization of costs in sequence analysis of social science data. *Sociological Methods and Research*, *38*(1), 197–231.

Good, I. J. (1982). An index of separateness of clusters and a permutation test for its significance. *Journal of Statistical Computation and Simulation*, *15*, 81–84.

Goslee, S. C., & Urban, D. L. (2007). The ecodist package for dissimilarity-based analysis of ecological data. *Journal of Statistical Software*, *22*, 1-19.

Gower, J. C., & Krzanowski, W. J. (1999). Analyses of distance for structured multivariate data and extensions to multivariate analysis of variance. *Applied Statistics*, *48*, 505–519.

Gower, J. C., & Legendre, P. (1986). Metric and euclidean properties of dissimilarity coefficients. *Journal of classification*, *3*, 5–48.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge (England): Cambridge University Press.

Hubert, L. J., & Schultz, J. (1976). Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, *29*, 190–241.

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding groups in data : an introduction to cluster analysis*. New York: Wiley.

Klein, J. P., & Moeschberger, M. L. (2007). *Survival analysis. techniques for censored and truncated data* (2nd ed.). New York: Springer.

Krause, E. F. (1975). *Taxicab geometry: an adventure in non-euclidean geometry*. New York: Dover Publications.

Kruskal, J. B., & Wish, M. (1978). *Multidimensional scaling*. Beverly Hills and London: Sage Publications.

Legendre, P., & Anderson, M. J. (1999). Distance-based redundance analysis: testing multi-species responses in multi-factorial ecological experiments. *Ecological Monographs*, *69*, 1–24.

Legendre, P., & Legendre, L. (1998). *Numerical ecology* (Second English ed.). Amsterdam: Elsevier.

Lesnard, L. (2010). Setting cost in optimal matching to uncover contemporaneous socio-temporal patterns. *Sociological Methods and Research*, *38*(3), 389–419.

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.

Macindoe, H., & Abbott, A. (2003). Handbook of data analysis. In M. Hardy & A. Bryman (Eds.), (chap. Sequence Analysis and Optimal Matching Techniques for Social Science Data). London: SAGE.

Manly, B. F. J. (1997). *Randomization, bootstrap and monte carlo methods in biology* (Second ed.). London: Chapman & Hall.

Mantel, N. (1967). The detection of disease clustering and a generalized regression approach. *Cancer Research*, *27*(2), 209–220.

McArdle, B. H., & Anderson, M. J. (2001). Fitting multivariate models to community data: a comment on distance-based redundancy analysis. *Ecology*, *82*(1), 290–297.

McVicar, D., & Anyadike-Danes, M. (2002). Predicting successful and unsuccessful transitions from school to work by using sequence methods. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, *165*(2), 317–334.

Mielke, P. W., Jr. (1985). Multiresponse permutation procedures. In S. Kotz & N. L. Johnson (Eds.), *Encyclopedia of statistical sciences vol. 5* (pp. 724–727). John Wiley and Sons.

Mielke, P. W., Jr, & Berry, K. J. (2001). *Permutation methods. A distance based approach* (First ed.). New York: Springer.

Oksanen, J., Blanchet, F. G., Kindt, R., Legendre, P., Minchin, P. R., O'Hara, R. B., ... Wagner, H. (2013). vegan: Community ecology package [Computer software manual]. Retrieved from `http://CRAN.R-project.org/package=vegan` (R package version 2.0-8)

Paradis, E., Claude, J., & Strimmer, K. (2004). APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, *20*, 289-290.

Pillar, V. D. P., & Orloci, L. (1996, Aug). On randomization testing in vegetation science: Multifactor comparisons of releve groups. *Journal of Vegetation Science*, *7*(4), 585–592.

R Core Team. (2012). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org/` (ISBN 3-900051-07-0)

Reiss, P. T., Henry, M., Stevens, H., Shehzad, Z., Petkova, E., & Milham, M. P. (2010). On distance-based permutation tests for between-group comparisons. *Biometrics*, *66*, 636–643.

Rizzo, M. L., & Székely, G. J. (2010). DISCO analysis: a nonparametric extension of analysis of variance. *The Annals of Applied Statistics*, *4*(2), 1034–1055.

Rosenbaum, P. R. (2005). An exact distribution-free test comparing two multivariate distributions based on adjacency. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(4), 515-530.

Sankoff, D., & Kruskal, J. B. (Eds.). (1983). *Time warps, string edits and macromolecules*. Reading, MA: Addison-Wesley.

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

Ware, J. H., & Lipsitz, S. (1988). Issues in the analysis of repeated categorical outcomes. *Statistics in Medicine*, *7*, 95–107.

Wilson, C. (2006). Reliability of sequence-alignment analysis of social processes: Monte Carlo tests of ClustalG software. *Environment and Planning A*, *38*, 187–204.

Wu, L. L. (2000, August). Some comments on "Sequence Analysis and Optimal Matching Methods in Sociology: Review and Prospect". *Sociological Methods and Research*, *29*(1), 41–64.

Yamaguchi, K. (1991). *Event history analysis*. Newbury Park, Calif.: Sage.

APPENDICES

APPENDIX A

Parallelizing the code

Using multiple CPU cores may speed up the code. In this case the packages `multicore` and `doMC` will be used.

To parallelize a computation means to divide the computation in parts and send each part to be run by a separate CPU core. Each time a CPU core finishes a job, communicates to the main process, and starts a new job, it spends precious time. If we were to send very many small jobs to a limited number of cores, too much time would be spent. For this reason, given few CPU cores (8 in our case) it is better to divide the job by partial jobs, and send each partial job to each core.

In this examination of the power of tests, we are calculating the proportion of tests that reject a null hypothesis, for a given parameter value. As an example, when analyzing the effect of varying the start parameter on power, the parameter can take values of

```
 [1] 10.00000 10.55556 11.11111 11.66667 12.22222 12.77778 13.33333 13.88889
 [9] 14.44444 15.00000
```

For each of these values the code will generate 2000 tests, each test done on sets of 30 each. So, a good way to divide the total computation is by using the 10 values of the start parameter, by creating 10 subjobs and allocating each to every CPU core.

This `R` code describes the process:

```
range.a2 <- seq(from=10,to=15,length.out=NumberOfSteps)
values <- foreach (fa.second = range.a2, .combine=rbind) %dopar%
                  {SimulateSequences(..., a2=fa.second) }
```

As said, `range.a2` defines the vector of possible values for the start parameter. `fa.second` is that same vector – `foreach` needs it like that to be digested. The pair `foreach %dopar%` allocates the job (`SimulateSequences()`) to each available core, by each value of `fa.second` (called `a2` when passed to `SimulateSequences()`). The function `SimulateSequences()` generates sequences and runs the tests, using some parameter values, among them, the passed value for `a2`.

APPENDIX B

Checking for correctness of Distortion measure

The distortion or stress measure should be valid.

For example, suppose that some triples are generated with our mechanism.

```
> numSequences=10
> Simulated=generateManySequences(mySequenceLength=30,
+                  myNumSeqs=numSequences,
+                  myStart=10,myIntensity=0.126)
> Triples= Simulated[[2]]
> Triples

             t1       t2       t3
 [1,]  7.155893 23.45470 35.20091
 [2,] 13.054163 21.09227 41.14212
 [3,] 14.060573 14.71772 36.94382
 [4,]  9.961945 18.95285 36.70726
 [5,] 12.852642 11.56148 23.91011
 [6,] 18.236457 64.29956 52.06315
 [7,] 10.416711 23.31193 27.66013
 [8,]  9.194970 13.02532 49.50518
 [9,] 32.386538 16.00019 26.55220
[10,] 22.307891 18.15877 33.03406
```

From that rectangular matrix of triples, $n \times 3$, an $n \times n$ square matrix of interitem distances is calculated. This will be our original distance matrix.

```
> Euc.dist = as.matrix(dist(Triples,method="euclidean"))
> round(Euc.dist,2)

       1     2     3     4     5     6     7     8     9    10
1   0.00  8.70 11.27  5.51 17.36 45.56  8.22 17.82 27.69 16.20
2   8.70  0.00  7.70  5.81 19.69 44.87 13.92 12.24 24.75 12.65
3  11.27  7.70  0.00  5.90 13.46 52.00 13.17 13.58 21.11  9.75
4   5.51  5.81  5.90  0.00 15.06 48.59 10.05 14.12 24.79 12.91
5  17.36 19.69 13.46 15.06  0.00 60.02 12.57 25.90 20.21 14.70
6  45.56 44.87 52.00 48.59 60.02  0.00 48.34 52.13 56.43 50.08
7   8.22 13.92 13.17 10.05 12.57 48.34  0.00 24.18 23.18 14.03
8  17.82 12.24 13.58 14.12 25.90 52.13 24.18  0.00 32.76 21.67
9  27.69 24.75 21.11 24.79 20.21 56.43 23.18 32.76  0.00 12.18
10 16.20 12.65  9.75 12.91 14.70 50.08 14.03 21.67 12.18  0.00
```

Now, let us derive an MDS solution from this original, of the same dimensionality as the original coordinates:

```
> mds.coord <- cmdscale(Euc.dist,k=3)
> mds.coord

          [,1]       [,2]        [,3]
1   -0.4159077  -5.047368 -6.11882511
2   -0.7092143  -4.703934  2.56822031
3    6.8112724  -3.077368  2.30342393
4    3.0675944  -4.966621 -1.84467164
5   14.9854840   2.953120 -6.53478455
6  -44.6768806   3.849201 -0.01689442
7    2.7429013   1.888164 -9.19079141
8    3.3096144 -14.695999  8.39207944
9    9.7019283  17.388089  6.57900800
10   5.1832077   6.412718  3.86323544
```

With this coordinates, let us derivate an $n \times n$ square matrix of distances.

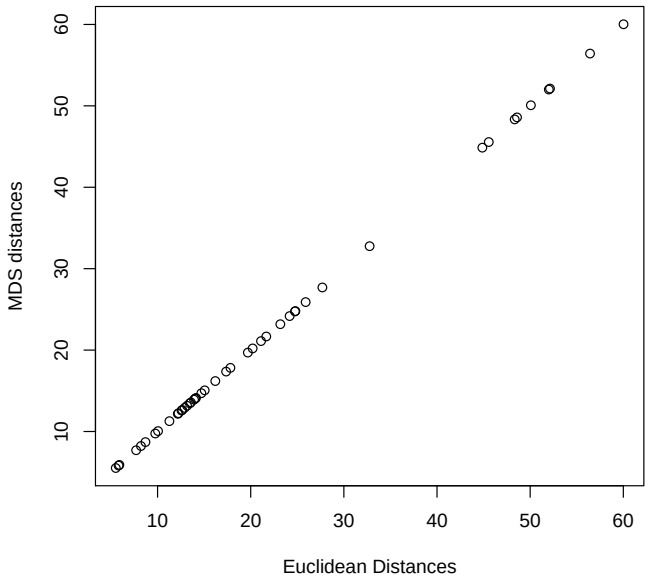
```
> MDS.dist <- as.matrix(dist(mds.coord))
> round(MDS.dist,2)

       1     2     3     4     5     6     7     8     9    10
1   0.00  8.70 11.27  5.51 17.36 45.56  8.22 17.82 27.69 16.20
2   8.70  0.00  7.70  5.81 19.69 44.87 13.92 12.24 24.75 12.65
3  11.27  7.70  0.00  5.90 13.46 52.00 13.17 13.58 21.11  9.75
4   5.51  5.81  5.90  0.00 15.06 48.59 10.05 14.12 24.79 12.91
5  17.36 19.69 13.46 15.06  0.00 60.02 12.57 25.90 20.21 14.70
6  45.56 44.87 52.00 48.59 60.02  0.00 48.34 52.13 56.43 50.08
7   8.22 13.92 13.17 10.05 12.57 48.34  0.00 24.18 23.18 14.03
8  17.82 12.24 13.58 14.12 25.90 52.13 24.18  0.00 32.76 21.67
9  27.69 24.75 21.11 24.79 20.21 56.43 23.18 32.76  0.00 12.18
10 16.20 12.65  9.75 12.91 14.70 50.08 14.03 21.67 12.18  0.00
```

Now, if our understanding is correct, Euc.dist and MDS.dist are equivalent, so there should be non distortion among them.

```
> Distortion(Euc.dist, MDS.dist)
```

```
[1] 7.579766e-16
```



Distortion measure is 0, and Shepard plot show a diagonal line, meaning there was no distortion, as expected.

As a further check, let us add a random quantity to the distances, to convince ourselves that the stress calculation holds for progressively more distorted matrices. To distort them, we add matrices of random values to it.

```
> distorted.a.bit = matrix(runif(n=100,min=0,max=5),nrow=10,ncol=10)
> distorted.a.lot = matrix(runif(n=100,min=0,max=20),nrow=10,ncol=10)
> distorted.a.wholelot =  matrix(runif(n=100,min=0,max=40),nrow=10,ncol=10)
> Distortion(Euc.dist,(MDS.dist+distorted.a.bit))

[1] 0.1120655

> Distortion(Euc.dist,(MDS.dist+distorted.a.lot))

[1] 0.4395485
```

```
> Distortion(Euc.dist,(MDS.dist+distorted.a.wholelot))
[1] 0.8698769
```

As expected, the computed value of Distortion increases with increasingly distorted matrices.

APPENDIX C

R source code

This section contains the code that was run.

```
### R code from vignette source 'IntroToGeneralStrat.Rnw'

###################################################
### code chunk number 1: IntroToGeneralStrat.Rnw:17-19
###################################################
library(TraMineR)
#set.seed(123)


###################################################
### code chunk number 2: try
###################################################
data(mvad)
mvad.lab = c("Employed", "Further Education", "Higher Education", "
    Jobless", "In School", "Training")
mvad.scode = c("EM", "FE", "HE", "JL", "SC", "TR")
mvad.seq = seqdef(mvad,17:86, states=mvad.scode,labels=mvad.lab,xtstep
    =6)
seqiplot(mvad.seq, border=NA)


###################################################
### code chunk number 3: transi
###################################################
#scost = seqsubm(mvad.seq,method="TRATE")
#round(scost,3)
scost=matrix(c(
0,2,2,2,2,2,
2,0,2,2,2,2,
2,2,0,2,2,2,
2,2,2,0,2,2,
2,2,2,2,0,2,
2,2,2,2,2,0),nrow=6,ncol=6,byrow=TRUE)



###################################################
### code chunk number 4: dist10
###################################################
mvad.om = seqdist(mvad.seq,method="OM",indel=1,sm=scost)
round(mvad.om[1:10,1:10],1)


###################################################
```

```
### code chunk number 5: IntroToGeneralStrat.Rnw:146-154
#########################################################
#NonMetric MDS , adapted from http://www.statmethods.net/advstats/mds.
    html
library(MASS)
d.example <- mvad.om[1:10,1:10] # euclidean distances between the rows
fit <- isoMDS(d.example, k=2) # k is the number of dim
# plot solution
x <- fit$points[,1]
y <- fit$points[,2]
rm(fit)


#########################################################
### code chunk number 6: IntroToGeneralStrat.Rnw:159-164
#########################################################
plot(x, y, xlab='Coordinate 1', ylab='Coordinate 2',
  main='Nonmetric MDS on 10 sequences',
     #type='n'
     )
text(x, y, labels = 1:10, cex=.7)


#########################################################
### code chunk number 7: example3loglogs
#########################################################
library(actuar)# provides the loglogistic distribution rllogis.
t1 <- 0 +  rllogis(1,shape=2.364,rate=0.078)
t2 <- 10 + rllogis(1,shape=2.364,rate=0.126)
t3 <- 20 + rllogis(1,shape=2.364,rate=0.078)
c(t1,t2,t3)


#########################################################
### code chunk number 8: IntroToGeneralStrat.Rnw:278-298
#########################################################

i = 1:40
S1 <- ifelse(i >= t1, 1, 0)
S2 <- ifelse(i >= t2, 1, 0)
S3 <- ifelse(i >= t3, 1, 0)
s.final = S1 * 4 + S2 * 2 + S3*1

s.alph <- rep("",length(s))
s.alph[s==0]="0"
s.alph[s==1]="i"
s.alph[s==2]="ii"
```

```
89  s.alph[s==3]="iii"
90  s.alph[s==4]="iv"
91  s.alph[s==5]="v"
92  s.alph[s==6]="vi"
93  s.alph[s==7]="vii"
94
95  mygrid = data.frame(S1,S2,S3,s.final)
96  mygrid
97
98
99
100 ######################################################
101 ### code chunk number 9: IntroToGeneralStrat.Rnw:311-316
102 ######################################################
103
104 library(TraMineR) # provides Anderson's Permanova
105 library(ecodist) # provides Mantel
106 library(vegan) # provides MRPP
107 library(ape)   # provides pcoa
108
109
110 ######################################################
111 ### code chunk number 10: DefineSequenceGenerationFunctions
112 ######################################################
113
114
115 generateSequence <- function(triple,seqLength) {
116   i = 1:seqLength
117   S1 <- ifelse(i >= triple[1],1,0)
118   S2 <- ifelse(i >= triple[2],1,0)
119   S3 <- ifelse(i >= triple[3],1,0)
120   s = S1 * 4 + S2*2 + S3 *1
121   return(s)
122 }
123
124 generateManySequences <- function(mySequenceLength,
125                                   myNumSeqs,
126                                   myStart,
127                                   myIntensity) {
128
129   seq.matrix=NULL
130   seq.matrix=matrix(nrow=myNumSeqs, ncol=mySequenceLength)
131
132   ######################################################################
133   # First generate three vectors of Loglogist random quantities,
134   # each with length SetsSequencesReferences.
135   t1 <- 0 +  rllogis(myNumSeqs,rate=0.078,shape=2.364)
```

```
136    t2 <- myStart + rllogis(myNumSeqs,
137                                    rate=myIntensity,
138                                    shape=2.364)
139    t3 <- 20 + rllogis(myNumSeqs, rate=0.078, shape=2.364)
140    # Catenate them; each row, defined by a triple, will serve to
141    # define the sequence.
142    ThreeLogLogs.r <- cbind(t1, t2, t3)
143
144    Set = t(apply(ThreeLogLogs.r, 1, generateSequence, seqLength=
           mySequenceLength))
145
146    seq.matrix[1:myNumSeqs,] = Set
147    # Make it character
148    seq.matrix <- matrix(as.character(seq.matrix),
149                                 nrow=myNumSeqs,
150                                 ncol=mySequenceLength)
151
152    return(list(Set, ThreeLogLogs.r, seq.matrix))
153 }
154
155
156
157 ####################################################
158 ### code chunk number 11: ejemplo5
159 ####################################################
160 misRefSequences = generateManySequences(mySequenceLength=30,
161                                         myNumSeqs=5,
162                                         myStart=10,
163                                         myIntensity=0.126)
164
165 misCompSequences= generateManySequences(mySequenceLength=30,
166                                         myNumSeqs=5,
167                                         myStart=20,
168                                         myIntensity=0.126)
169
170 myexample = rbind(misRefSequences[[3]], misCompSequences[[3]])
171 myexample.seq = seqdef(myexample)
172
173
174 ####################################################
175 ### code chunk number 12: IntroToGeneralStrat.Rnw:379-380
176 ####################################################
177 myexample.seq[10:1,]
178
179
180 ####################################################
181 ### code chunk number 13: IntroToGeneralStrat.Rnw:385-388
```

```
182 ###################################################
183 #pdf(file="ejemplo.pdf")
184 seqiplot(myexample.seq, border=NA)
185 #dev.off()
186
187
188 ###################################################
189 ### code chunk number 14: IntroToGeneralStrat.Rnw:436-441
190 ###################################################
191 n=15
192 Simulated=generateManySequences(mySequenceLength=40,
193                                 myNumSeqs=n,
194                                 myStart=10,
195                                 myIntensity=0.126)
196
197
198 ###################################################
199 ### code chunk number 15: IntroToGeneralStrat.Rnw:446-448
200 ###################################################
201 Tr= Simulated[[2]]
202 Tr
203
204
205 ###################################################
206 ### code chunk number 16: IntroToGeneralStrat.Rnw:455-456
207 ###################################################
208 D.euc = as.matrix(dist(Tr,method="euclidean"))
209
210
211 ###################################################
212 ### code chunk number 17: IntroToGeneralStrat.Rnw:462-464
213 ###################################################
214  myalphabet = as.character(c(0,1,2,3,4,5,6,7))
215  Simulated.seq = seqdef(Simulated[[3]],alphabet=myalphabet)
216
217
218 ###################################################
219 ### code chunk number 18: IntroToGeneralStrat.Rnw:469-479
220 ###################################################
221
222 cost.matrix = matrix(c(
223 0,2,2,2,2,2,2,2,
224 2,0,2,2,2,2,2,2,
225 2,2,0,2,2,2,2,2,
226 2,2,2,0,2,2,2,2,
227 2,2,2,2,0,2,2,2,
228 2,2,2,2,2,0,2,2,
```

```
2,2,2,2,2,2,0,2,
2,2,2,2,2,2,2,0), nrow=8,ncol=8,byrow=TRUE)


#######################################################
### code chunk number 19: IntroToGeneralStrat.Rnw:482-484
#######################################################
OM.dist <- seqdist(Simulated.seq, method="OM", indel=1,sm=cost.matrix)



#######################################################
### code chunk number 20: ShepardPlot
#######################################################
ShepardPlot = function (A, B, myxlab, myylab) {
    # A, B need to be matrix objects.
    a = as.vector(as.dist(A))
    b = as.vector(as.dist(B))
    plot(a,b, xlab=myxlab, ylab=myylab)

  }


#######################################################
### code chunk number 21: IntroToGeneralStrat.Rnw:503-506
#######################################################
ShepardPlot(D.euc,OM.dist,
myxlab="Euclidean distance among triples delta(ti,tj)",
myylab="Optimal Matching distance among sequences d(si,sj)")


#######################################################
### code chunk number 22: IntroToGeneralStrat.Rnw:523-526
#######################################################
ShepardPlot(D.euc, sqrt(OM.dist+1),
myxlab="Euclidean distance among triples delta(ti,tj)",
myylab="Square Root of Optimal Matching distance among sequences log(d(
    si,sj))")


#######################################################
### code chunk number 23: DefineStandardizedDistortion
#######################################################

Distortion = function(m1,m2) {
       m1.s = m1
       m2.s = m2
```

```
275         Numerator = sum(as.vector(( m1.s - m2.s)^2))
276         Denominator=sum(as.vector(m1.s^2))
277         return(sqrt(Numerator/Denominator))
278
279 }
280
281 CopheneticCorrel = function(m1,m2) {
282       return(cor(as.vector(m1),as.vector(m2)))
283       }
284
285
286
287
288 ######################################################
289 ### code chunk number 24: GenerateOMandEucDistances1
290 ######################################################
291 numSequences=30
292 runs = 1000
293 allDistort = NULL
294
295 possibleSeqLengths = seq(from=35, to=55, by=5)
296
297 totaln = runs*length(possibleSeqLengths)
298 Inter.Triples.dist = array(NA,c(totaln,
299                                 numSequences,numSequences))
300 Inter.Seq.OM.dist  = array(NA,c(totaln,
301                                 numSequences,numSequences))
302
303 LongitudSeq =             rep(NA,totaln)
304 counter=0
305
306 # Let is see what effect has the proportion between
307 # replacement costs and indel costs.
308
309 for (thisSeqLength in possibleSeqLengths) {
310
311
312   for (myrun in 1:runs) {
313
314    counter=counter+1
315    LongitudSeq[counter]=thisSeqLength
316
317    # create triples:
318
319    Simulated=generateManySequences(
320                  mySequenceLength=thisSeqLength,
321                  myNumSeqs=numSequences,
```

```
322                     myStart=10,myIntensity=0.126)
323     Triples= Simulated[[2]]
324     Sequences = Simulated[[3]]
325
326     # Calculate Euclidean distances among triples
327     Inter.Triples.dist[counter,,] = as.matrix(dist(Triples,method="
            euclidean"))
328
329     # Create a "set-of-sequences" object
330     Simulated.seq = seqdef(Sequences,alphabet=myalphabet)
331
332     # Calculate matrix of inter sequence OM distances for set
333     # of sequences
334     Inter.Seq.OM.dist[counter,,] <- seqdist(Simulated.seq,method="OM",
335                                             indel=1,sm=cost.matrix)
336 }}
337
338
339
340 ######################################################
341 ### code chunk number 25: CalculateDistortionTriplesOM1
342 ######################################################
343 Distort.Trip.OM =  rep(NA,totaln)
344 Corr.Trip.OM = rep(NA,totaln)
345 counter=0
346 for (thisSeqLength in possibleSeqLengths) {
347   for (myrun in 1:runs) {
348    counter=counter+1
349
350    # Calculate Distortion between OM distances and Triples distances.
351    Distort.Trip.OM[counter] = Distortion(Inter.Seq.OM.dist[counter,,],
352                                          Inter.Triples.dist[counter,,])
353    Corr.Trip.OM[counter]    = CopheneticCorrel(
354                                  Inter.Seq.OM.dist[counter,,],
355                                  Inter.Triples.dist[counter,,])
356
357
358  } }
359
360
361 ######################################################
362 ### code chunk number 26: ShepPlotTripOM1
363 ######################################################
364 boxplot(Distort.Trip.OM ~ LongitudSeq,
365   log="y",
366   xlab="Sequence Length",
367   ylab="Measure of Distortion")
```

```
368
369
370 ####################################################
371 ### code chunk number 27: ShepPlotTripOM2
372 ####################################################
373 boxplot(Corr.Trip.OM ~ LongitudSeq,
374   xlab="Sequence Length",
375   ylab="Measure of Correlation")
376
377
378 ####################################################
379 ### code chunk number 28: Generate4245WithLength40
380 ####################################################
381 generateSequence(triple=c(15,42,30),seqLength=40)
382 generateSequence(triple=c(15,45,30),seqLength=40)
383
384
385 ####################################################
386 ### code chunk number 29: Generate4245WithLength48
387 ####################################################
388 generateSequence(triple=c(15,42,30),seqLength=50)
389 generateSequence(triple=c(15,45,30),seqLength=50)
390
391
392 ####################################################
393 ### code chunk number 30: DefineCostMatrix
394 ####################################################
395
396 cost.matrix = matrix(c(
397   0,2,2,2,2,2,2,2,
398   2,0,2,2,2,2,2,2,
399   2,2,0,2,2,2,2,2,
400   2,2,2,0,2,2,2,2,
401   2,2,2,2,0,2,2,2,
402   2,2,2,2,2,0,2,2,
403   2,2,2,2,2,2,0,2,
404   2,2,2,2,2,2,2,0), nrow=8,ncol=8,byrow=TRUE)
405
406
407 ####################################################
408 ### code chunk number 31: GenerateOMandEucDistances
409 ####################################################
410 numSequences=30
411 runs = 1000
412 allDistort = NULL
413
414 possiblecosts = seq(from=0.25, to=3, length.out=12)
```

```
415
416 totaln = runs*length(possiblecosts)
417
418 IndelCost =            rep(NA, totaln)
419
420 Inter.Triples.dist = array(NA, c(totaln,
421                                  numSequences, numSequences))
422 Inter.Seq.OM.dist  = array(NA, c(totaln,
423                                  numSequences, numSequences))
424
425 counter=0
426
427 # Let is see what effect has the proportion between
428 # replacement costs and indel costs.
429
430 for (indelcost in possiblecosts) {
431
432   for (myrun in 1:runs) {
433
434    counter=counter+1
435    IndelCost[counter]=indelcost
436
437    # create triples:
438
439    Simulated=generateManySequences(
440                  mySequenceLength=30,
441                  myNumSeqs=numSequences,
442                  myStart=10, myIntensity=0.126)
443    Triples= Simulated[[2]]
444    Sequences = Simulated[[3]]
445
446    # Calculate Euclidean distances among triples
447    Inter.Triples.dist[counter,,] = as.matrix(dist(Triples, method="
       euclidean"))
448
449    # Create a "set-of-sequences" object
450    Simulated.seq = seqdef(Sequences, alphabet=myalphabet)
451
452    # Calculate matrix of inter sequence OM distances for set
453    # of sequences
454    Inter.Seq.OM.dist[counter,,] <- seqdist(Simulated.seq, method="OM",
455                                            indel=indelcost, sm=cost.matrix)
456 }}
457
458
459
460 ###################################################
```

```
### code chunk number 32: CalculateDistortionTriplesOM
#########################################################
Distort.Trip.OM =  rep(NA, totaln)
counter=0
for (indelcost in possiblecosts) {
  for (myrun in 1:runs) {
   counter=counter+1

   # Calculate distortion between OM distances and Triples distances.
   Distort.Trip.OM[counter] = Distortion(Inter.Seq.OM.dist[counter,,],
                                         Inter.Triples.dist[counter,,])

 } }


#########################################################
### code chunk number 33: ShepPlotTripOM
#########################################################
boxplot(Distort.Trip.OM ~ IndelCost,
  log="y",
  xlab="Cost of Insertion-Deletion",
  ylab="Measure of Distortion")


#########################################################
### code chunk number 34: FindNonEuclidean
#########################################################
# ade4 package provides is.euclid
library(ade4)

IsEuclidean = rep(NA,runs*length(possiblecosts))
counter=0

for (indelcost in possiblecosts) {

  for (j in 1:runs) {

   counter=counter+1

   # Determine if matrix is Euclidean or not.
   # is.euclid expects a dist object, so it should be
   # transformed first (take square root, and make
   # distance object).
   IsEuclidean[counter] = is.euclid(
                            as.dist(
                             sqrt(Inter.Seq.OM.dist[counter,,])))
 }
```

```
508 }
509 EucByIndel =  aggregate(IsEuclidean, by=list(IndelCost), mean)
510
511
512
513 ######################################################
514 ### code chunk number 35: PlotEuc
515 ######################################################
516 plot(EucByIndel[[1]],EucByIndel[[2]],
517       xlab="Indel costs ",
518       ylab=paste("Proportion of Euclidean matrices"))
519
520
521 ######################################################
522 ### code chunk number 36: MDSDeepPrep
523 ######################################################
524 Distort.diff =  rep(NA, totaln)
525 #dim(Inter.Seq.OM.dist)
526 #totaln
527 #Inter.Seq.OM.dist[1,,]
528 #counter
529
530
531 ######################################################
532 ### code chunk number 37: MDSDeep
533 ######################################################
534 library(vegan)
535 counter=0
536 for (indelcost in possiblecosts) {
537
538  for (j in 1:runs) {
539
540   counter=counter+1
541
542   # Get a MDS representation from OM matrices.
543    MDS.coord = metaMDS(Inter.Seq.OM.dist[counter,,],
544                        distance="euclidean",k=3)$points
545    MDS.dist = as.matrix(dist(MDS.coord))
546   # Calculate distortion between matrix of true inter-triple
547   # distances and distance matrix from MDS solution.
548    Distortion.True.MDS = Distortion(MDS.dist,
549                                     Inter.Triples.dist[counter,,])
550     #Correl.True.MDS = CopheneticCorrel(MDS.dist,
551     #                                   Inter.Triples.dist[counter,,])
552   # We already calculated distortion between
553   # true inter-triple distances and OM; it is Distort.Trip.OM
554   # Is this distortion
```

```
555
556   Distortion.diff[counter] = Distortion.True.MDS - Distort.Trip.OM[
          counter]
557   }
558 }
559 #summary(Distortion.diff)
560
561 # Calculate an MDS solution to OM distances and get a
562     # new matrix of distances from it.
563     #mds.coord <- cmdscale(Inter.Seq.OM.dist,k=3)
564     #mds.coord <- (isoMDS(as.dist(Inter.Seq.OM.dist), k=3, p=2))$points
565     #Inter.MDS.OM.dist <- as.matrix(dist(mds.coord))
566
567     # Calculate distortion between matrix of inter-triple distances
568     # and matrix of MDS solution to OM distances.
569     #Distort.Trip.MDS.om[counter] = Distortion(Inter.MDS.OM.dist,
570     #                                          Inter.Triples.dist)
571
572     #Distort.OM.MDS[counter] = Distortion(Inter.MDS.OM.dist, Inter.Seq.OM
          .dist)
573
574
575
576 ###################################################
577 ### code chunk number 38: IntroToGeneralStrat.Rnw:1181-1182
578 ###################################################
579 Distortion.diff[Distortion.diff > 10] <- NA
580
581
582 ###################################################
583 ### code chunk number 39: IntroToGeneralStrat.Rnw:1185-1186
584 ###################################################
585 summary(Distortion.diff)
586
587
588 ###################################################
589 ### code chunk number 40: Distortdiff
590 ###################################################
591 boxplot(Distortion.diff ~ IndelCost,
592   xlab="Cost of Insertion-Deletion",
593   ylab="Distortion difference")
594
595
596 ###################################################
597 ### code chunk number 41: GenerOM
598 ###################################################
599 n=100
```

```
600 Simulated=generateManySequences(myNumSeqs=n,
601                                 mySequenceLength=40,
602                                  myStart=10,
603                                  myIntensity=0.126)
604  ######################################
605  # True triples
606  Triples= Simulated[[2]]
607  # Calculate true Euclidean distances among triples
608  Inter.Triples.dist = as.matrix(dist(Triples,method="euclidean"))
609
610  ######################################
611  # calculation of OM distances
612  myalphabet = as.character(c(0,1,2,3,4,5,6,7))
613  Simulated.seq = seqdef(Simulated[[3]],alphabet=myalphabet)
614
615 cost.matrix = matrix(c(
616 0,2,2,2,2,2,2,2,
617 2,0,2,2,2,2,2,2,
618 2,2,0,2,2,2,2,2,
619 2,2,2,0,2,2,2,2,
620 2,2,2,2,0,2,2,2,
621 2,2,2,2,2,0,2,2,
622 2,2,2,2,2,2,0,2,
623 2,2,2,2,2,2,2,0), nrow=8,ncol=8,byrow=TRUE)
624 Simulated.dist <- seqdist(Simulated.seq,method="OM",indel=1,sm=cost.
        matrix)
625
626
627
628
629 ######################################################
630 ### code chunk number 42: IntroToGeneralStrat.Rnw:1268-1270
631 ######################################################
632 hist(as.vector(as.dist(Inter.Triples.dist)), xlim=c(0,100),
633       xlab="Interitem distances between latent triples")
634
635
636 ######################################################
637 ### code chunk number 43: IntroToGeneralStrat.Rnw:1282-1283
638 ######################################################
639 hist(as.dist(Simulated.dist),xlab="Interitem distances",main="OM
        distances")
640
641
642 ######################################################
643 ### code chunk number 44: getmds
644 ######################################################
```

```
645 MDS.coord = metaMDS(Simulated.dist,
646                         distance="euclidean",k=3)$points
647 MDS.dist = as.matrix(dist(MDS.coord))
648
649
650 ########################################################
651 ### code chunk number 45: IntroToGeneralStrat.Rnw:1302-1303
652 ########################################################
653 hist(as.dist(MDS.dist), main="MDS solution",xlab="Interitem distances")
654
655
656 ########################################################
657 ### code chunk number 46: IntroToGeneralStrat.Rnw:1334-1347
658 ########################################################
659 library(MASS) # provides mvrnorm()
660
661 # Diagonal Variance/Covariance matrix
662
663 p=3
664 mySigma=matrix(c
665           (1,0,0,
666            0,1,0,
667            0,0,1),p,p)
668
669 X = mvrnorm(100,mu=rep(0,p), Sigma=mySigma)
670 d = dist(X)
671 d2 = dist(X)^2  # Get the square of such distances
672
673
674 ########################################################
675 ### code chunk number 47: IntroToGeneralStrat.Rnw:1353-1354
676 ########################################################
677 hist(d, freq=FALSE, xlab="Interitem distances, items are Multinormal")
678
679
680 ########################################################
681 ### code chunk number 48: IntroToGeneralStrat.Rnw:1526-1530
682 ########################################################
683 vec1 = c(rep(0,5),  rep(1,5))
684 differences.matrix = 1 * outer(vec1,vec1,FUN= "==")
685 differences.matrix.asdist = as.dist(differences.matrix)
686
687
688
689 ########################################################
690 ### code chunk number 49: IntroToGeneralStrat.Rnw:1544-1547
691 ########################################################
```

```
692 vec1
693 X = 1 * outer(vec1, vec1, FUN="!=")
694 X
695
696
697 ###################################################
698 ### code chunk number 50: IntroToGeneralStrat.Rnw:1550-1552
699 ###################################################
700 X * d.example
701 sum(X * d.example)/2 # usually sum over lower triangle
702
703
704 ###################################################
705 ### code chunk number 51: IntroToGeneralStrat.Rnw:1559-1563
706 ###################################################
707 X = 1*outer(vec1, vec1, FUN="==")
708 X
709 X * d.example
710 sum(X * d.example)/2
711
712
713 ###################################################
714 ### code chunk number 52: IntroToGeneralStrat.Rnw:1576-1579
715 ###################################################
716 vec1
717 permuted.vector = sample(vec1, replace=FALSE)
718 permuted.vector
719
720
721 ###################################################
722 ### code chunk number 53: IntroToGeneralStrat.Rnw:1582-1586
723 ###################################################
724 X = 1*outer(permuted.vector, permuted.vector, FUN="==")
725 X
726 X * d.example
727 sum(X * d.example)/2
728
729
730 ###################################################
731 ### code chunk number 54: FreeMemByDeletingHugeMatrices
732 ###################################################
733 rm(Inter.Triples.dist)
734 rm(Inter.Seq.OM.dist)
```

./IntroToGeneralStrat.R

```
1 ### R code from vignette source 'simulation.Rnw'
```

```
2
3  ###################################################
4  ### code chunk number 1: somevars
5  ###################################################
6  TotalReps=2000
7  theta=0.5
8  SequencesPerSet = 30
9  SetsSequencesReference=    SequencesPerSet*theta
10 SetsSequencesSecondGroup= SequencesPerSet*(1-theta)
11 NumberOfSteps=10
12
13
14 ###################################################
15 ### code chunk number 2: init
16 ###################################################
17
18 # For parallel coding
19 library(doMC)
20 registerDoMC()
21 #getDoParWorkers()
22
23 #i = 1:SequenceLength
24
25
26 ###################################################
27 ### code chunk number 3: simulation.Rnw:103-107
28 ###################################################
29 vec1 = c(rep(0,SetsSequencesReference),
30           rep(1,SetsSequencesSecondGroup))
31 differences.matrix = 1 * outer(vec1,vec1,FUN= "==")
32 differences.matrix.asdist = as.dist(differences.matrix)
33
34
35 ###################################################
36 ### code chunk number 4: simulation.Rnw:111-127
37 ###################################################
38 Tot0 = rep(0,TotalReps)
39 p.values.Mantel <- Tot0
40 p.values.AndersonF <- Tot0
41 p.values.ANOSIM <- Tot0
42 p.values.mrpp <- Tot0
43 p.values.ttest <- Tot0
44 p.values.perm.median <- Tot0
45
46 p.values.MANOVA.pcoa <- Tot0
47 p.values.Mantel.pcoa <- Tot0
48 p.values.Anderson.pcoa <- NULL
```

```
49 p.values.ANOSIM.pcoa <- NULL
50 p.values.mrpp.pcoa <- NULL
51
52
53
54
55
56 ######################################################
57 ### code chunk number 5: subsMatrix
58 ######################################################
59 cost.matrix = matrix(c(
60 0,2,2,2,2,2,2,2,
61 2,0,2,2,2,2,2,2,
62 2,2,0,2,2,2,2,2,
63 2,2,2,0,2,2,2,2,
64 2,2,2,2,0,2,2,2,
65 2,2,2,2,2,0,2,2,
66 2,2,2,2,2,2,0,2,
67 2,2,2,2,2,2,2,0), nrow=8,ncol=8,byrow=TRUE)
68
69
70 ######################################################
71 ### code chunk number 6: showCosts
72 ######################################################
73 cost.matrix
74
75
76 ######################################################
77 ### code chunk number 7: subsMatrix.NonMetric
78 ######################################################
79 cost.matrix.nonMetric = matrix(c(
80 0,1,1,1,1,1,1,1,
81 1,0,1,1,1,1,1,1,
82 1,1,0,1,1,1,1,1,
83 1,1,1,0,1,1,1,1,
84 1,1,1,1,0,1,1,1,
85 1,1,1,1,1,0,1,1,
86 1,1,1,1,1,1,0,1,
87 1,1,1,1,1,1,1,0), nrow=8,ncol=8,byrow=TRUE)
88
89
90 ######################################################
91 ### code chunk number 8: coreOfSim
92 ######################################################
93 library(vegan)
94 library(ape)
95 library(ecodist)
```

```
96  library(actuar)
97  library(TraMineR)
98
99  SimulateSequences <- function(SequenceLength,
100                                ReferenceIntensity,
101                                Start2,
102                                mylambda2) {
103
104   seq.matrix=NULL
105   seq.matrix=matrix(nrow=SequencesPerSet,ncol=SequenceLength)
106
107   for (kounter in 1:TotalReps) {
108
109   ReferenceSet = generateManySequences(
110                                  mySequenceLength=SequenceLength,
111                                  myNumSeqs=SetsSequencesReference,
112                                  myStart=10,
113                                  myIntensity=ReferenceIntensity)
114
115   t2.baseline = ReferenceSet[[2]][,2]
116
117   ComparativeSet = generateManySequences(
118                                  mySequenceLength=SequenceLength,
119                                  myNumSeqs=SetsSequencesSecondGroup,
120                                  myStart=Start2,
121                                  myIntensity=mylambda2)
122
123   t2.comp = ComparativeSet[[2]][,2]
124
125    seq.matrix = rbind(ReferenceSet[[3]],ComparativeSet[[3]])
126
127       ##########################################################################
128       #  For our more parametric exploration, let us derive a discretized
129       #  transformation of the random quantity of interest:
130
131       #t2.baseline = object[[3]]
132       #t2.comp = object[[4]]
133
134       x.baseline = ceiling(t2.baseline)
135       x.baseline.trunc = subset(x.baseline,(x.baseline <= SequenceLength)
              )
136       x.tocompare = ceiling(t2.comp)
137       x.tocompare.trunc = subset(x.tocompare,(x.tocompare <=
              SequenceLength))
138       # Truncated versions correspond to those cases where
139       # t was greater than 41.
140
```

```
141
142
143       # Now that we have a set of sequences stored in seq.matrix,
144       # produce a distance matrix from sequences using an OM distance
145       # and store it in matdist.
146       myalphabet = as.character(c(0,1,2,3,4,5,6,7))
147       matseq <- seqdef(seq.matrix, alphabet=myalphabet)
148       matdist <- seqdist(matseq, method="OM", indel=1,sm=cost.matrix)
149
150       #matdist.nonmetric <-seqdist(matseq, method="OM", indel=1,sm=cost.
              matrix.nonMetric)
151
152       # This is the part where, given two matrices,
153       # the tests are run.
154       ############################################################
155       # Mantel correlation between matdist (Y) and the matrix of
156       # origins (first half of people are fixed-parameter people
157       my.mantel = ecodist::mantel(differences.matrix.asdist
158                                   ~ as.dist(matdist),
159                                   nperm=1000)
160       # this vector of size TotalReps stores the p-values for each
161       # repetition.
162       # one-tailed p-value (H0: r >= 0)
163       p.values.Mantel[kounter] = my.mantel[3]
164       ##########################################################
165       #  dissassoc McArdle and Anderson
166       my.dissassoc = dissassoc(matdist,group=vec1,R=1000)
167       p.values.AndersonF[kounter] = my.dissassoc$stat[1,2]
168       ##########################################################
169       # MRPP ?
170       my.mrpp = mrpp(dat=as.dist(matdist),
171                      grouping=vec1,
172                      permutations=1000)
173       p.values.mrpp[kounter] = my.mrpp$Pvalue
174       #p.values.mrpp[kounter] = NA
175       ##############################################
176       #   force random matrix to a PCoA form,
177       matdist.pcoa <- (pcoa(matdist))$vectors
178
179       # now, with a matrix of coordinates I can do many things.
180       # One is to assume that such matrix is multinormally
181       # distributed, and run a test. matdist.pcoa.m
182       manovafit <-  summary(manova(matdist.pcoa ~ vec1))
183       p.values.MANOVA.pcoa[kounter] = manovafit$stats["vec1","Pr(>F)"]
184
185
186       ################################################
```

```
187       #  This one instead takes the pcoa, converts it back
188       # to a matrix of distances (meaning it "cleans" the non
189       # euclidian weirdness, and reruns the Mantel tests on it.
190       matdist.pcoa.m <- dist(matdist.pcoa)
191
192       my.mantel.pcoa = ecodist::mantel(differences.matrix.asdist
193                                        ~ matdist.pcoa.m, nperm=1000)
194       p.values.Mantel.pcoa[kounter] = my.mantel.pcoa[3]
195
196       ################################################
197       #
198       #   Take squared root of matrix
199
200       matdist.raised <- matdist^(0.5)
201
202
203       ##############################################################
204       #    More parametric tests:
205       # Plain comparison of two sample means:
206
207       p.values.ttest[kounter] = t.test(x.baseline.trunc,x.tocompare.trunc
              )$p.value
208       ##############################################################
209       #  permutational test of differences between medians.
210       #
211        diff.medians = abs(median(x.baseline) - median(x.tocompare))
212        diff.vector = rep(0,1000)
213        x.all = c(x.baseline,x.tocompare)
214        indicator = c(rep(0,SetsSequencesReference),rep(1,
               SetsSequencesSecondGroup))
215        for (j in 1:1000) {
216          perm.indicator = sample(indicator,replace=FALSE)
217          median1 = median(x.all[perm.indicator==1])
218          median0 = median(x.all[perm.indicator==0])
219          diff.vector[j]=abs(median1-median0)
220          }
221        p.values.perm.median[kounter] = sum(diff.vector >= diff.medians)/
               1000
222       ################################################################
223   }
224   # Function returns the proportion of test p-values
225   # whose values indicate a rejection of Null Hypothesis.
226  myvector=c( Start2,
227              mylambda2,
228              sum(p.values.Mantel < 0.05)/TotalReps,
229              sum(p.values.AndersonF < 0.05)/TotalReps,
230              sum(p.values.mrpp < 0.05)/TotalReps,
```

```
231                 sum(p.values.ttest < 0.05)/TotalReps,
232                 sum(p.values.perm.median < 0.05)/TotalReps,
233                 sum(p.values.MANOVA.pcoa < 0.05)/TotalReps,
234                 sum(p.values.Mantel.pcoa < 0.05)/TotalReps,
235                 SequenceLength
236               )
237  #grid.results  = rbind(grid.results,myvector)
238
239 #return(grid.results)
240 return(myvector)
241 }
242
243
244 ######################################################
245 ### code chunk number 9: simul
246 ######################################################
247 data.sim.mll = NULL
248 PassSequenceLength=40
249 range.a2 = seq(from=10,to=15,length.out=10)
250
251 values <- foreach (fa.second = range.a2,
252                    .combine=rbind) %dopar% {SimulateSequences(
253                                        SequenceLength=PassSequenceLength,
254                                        ReferenceIntensity=0.126,
255                                            Start2=fa.second,
256                                            mylambda2=0.126)}
257
258 data.sim.mll = values
259 #values = NULL
260
261
262 ######################################################
263 ### code chunk number 10: graf
264 ######################################################
265 powerplot = function(mydata,xlegend,cual) {
266
267 if (cual==1) {minx=10}
268 if (cual==2) {minx=0.078}
269 if (cual==10) {minx=25}
270
271 x = mydata[,cual]
272
273 pMantel=mydata[,3]
274 pPermanova=mydata[,4]
275 pMRPP=mydata[,5]
276 pt.trunc=mydata[,6]
277 pdiffmedians=mydata[,7]
```

```
278 pmanova.pcoa=mydata[,8]
279 pmantel.pcoa=mydata[,9]
280
281
282
283 plot(x,pMantel,pch="M",type="b",col="green",
284       xlab=xlegend,ylab="% of p-values < .05",ylim=c(0:1))
285 points(x,pPermanova,pch="P",type="b",col="purple")
286 points(x,pMRPP,pch="*",type="b",col="blue")
287 points(x,pt.trunc,pch="t",type="b",col="brown")
288 points(x,pdiffmedians,pch="+",type="b",col="red")
289 points(x,pmanova.pcoa,pch="m",type="b",col="black")
290 #points(mydata[,cual],mydata$mantel.pcoa,pch="x",type="b",col="blue")
291 #points(mydata[,cual],mydata[,10],pch="z",type="b",col="dark blue")
292
293 legend(minx,1, c("Mantel","Permanova", "MRPP","t truncated","diff
        medians",
294                  "MANOVA on PCoA"),
295        pch=c("M","P","*","t","+","m"),
296        col=c("green","purple","blue","brown","red","black"),
297        lty=1)
298 }
299
300
301 ######################################################
302 ### code chunk number 11: simulation.Rnw:412-414
303 ######################################################
304 #data.sim.mll
305 powerplot(data.sim.mll,"Start parameter",1)
306
307
308 ######################################################
309 ### code chunk number 12: simulIntens
310 ######################################################
311 data.sim.mll = NULL
312 PassSequenceLength=40
313 range.lambda2 = seq(from=0.078, to=0.205, length.out=10)
314
315 values <- foreach (lambda2 = range.lambda2,
316                    .combine=rbind) %dopar% {
317                     SimulateSequences(
318                                       SequenceLength=PassSequenceLength,
319                                       ReferenceIntensity=0.078,
320                                       Start2=10,
321                                       mylambda2=lambda2)
322                   }
323
```

```
values

#for (my.lambda.second in range.lambda2) {
#   values = SimulateSequences(a.second=10,lambda.second=my.lambda.second
    )

#data.sim.mll = rbind(data.sim.mll, values)
#  }


##################################################
### code chunk number 13: simulation.Rnw:497-498
##################################################
#data.sim.mll


##################################################
### code chunk number 14: graf3
##################################################
powerplot(values,"Rate parameter",2)
```

./simulation.R

```
### R code from vignette source 'EffectOfLength.Rnw'

##################################################
### code chunk number 1: EffectLength
##################################################
#values = NULL
#data.sim.mll = NULL
range.seqlength = c(25, 40, 50, 60)

values <- foreach (thisPassSequenceLength = range.seqlength,
                   .combine=rbind) %dopar% {
                                SimulateSequences(
                                  SequenceLength=thisPassSequenceLength
                                    ,
                                  ReferenceIntensity=0.126,
                                     Start2=15,
                                     mylambda2=0.126)
                              }


##################################################
### code chunk number 2: EffectOfLength.Rnw:31-33
##################################################
```

```
powerplot(values,"Sequence Length",10)
values=NULL


#####################################################
### code chunk number 3: newsim
#####################################################
generateManySequences <- function(mySequenceLength,
                                  myNumSeqs,
                                  myStart,
                                  myIntensity) {

  seq.matrix=NULL
  seq.matrix=matrix(nrow=myNumSeqs, ncol=mySequenceLength)

  ######################################################################
  # First generate three vectors of Loglogist random quantities,
  # each with length SetsSequencesReferences.
  t1 <- 0 +  rtrunc(n=myNumSeqs,spec="llogis",
                    a=0,b=mySequenceLength,
                    rate=0.078,shape=2.364)

  t2 <- myStart + rtrunc(n=myNumSeqs,spec="llogis",
                         a=0, b=mySequenceLength,
                         rate=myIntensity, shape=2.364)

  t3 <- 20 +       rtrunc(n=myNumSeqs,spec="llogis",
                         a=0,b=mySequenceLength,
                         rate=0.078,shape=2.364)

  # Catenate them; each row, defined by a triple, will serve to
  # define the sequence.
  ThreeLogLogs.r <- cbind(t1,t2,t3)

  Set = t(apply(ThreeLogLogs.r,1,generateSequence,seqLength=
      mySequenceLength))

  seq.matrix[1:myNumSeqs,] = Set
  # Make it character
  seq.matrix <- matrix(as.character(seq.matrix),
                               nrow=myNumSeqs,
                               ncol=mySequenceLength)

  return(list(Set,ThreeLogLogs.r,seq.matrix))
}


```

```
70
71 ##################################################
72 ### code chunk number 4: EffectLength2
73 ##################################################
74 library(truncdist)
75 values = NULL
76 #data.sim.mll = NULL
77 range.seqlength = c(25, 40, 50, 60)
78
79 values <- foreach (thisPassSequenceLength = range.seqlength,
80                    .combine=rbind) %dopar% {
81                                  SimulateSequences(
82                                    SequenceLength=thisPassSequenceLength
                                        ,
83                                    ReferenceIntensity=0.126,
84                                      Start2=15,
85                                      mylambda2=0.126)
86                                }
87
88
89
90 ##################################################
91 ### code chunk number 5: EffectOfLength.Rnw:114-115
92 ##################################################
93 powerplot(values,"Sequence Length",10)
```

./EffectOfLength.R