Dependent Comparanda Interlanguage Measure for Historical Linguistics and Dialectology

H.M. Hubey, Professor
Department of Computer Science
Montclair State University, Upper Montclair, NJ 07043

Abstract

There are various "quantitative" methods employed in historical linguistics. These attempt to create family trees, judge time depths of separation, etc. The particular means of these attempts have given rise to various terms such as glottochronology, or lexicostatistics. In the first type, e.g. statistical attempts are those due to Nichols[1991], or Embleton[1986], or correlation-regression analysis as done by Labov, and this paper has nothing to say about any of these. The second category of works attempt to determine "how many" putative cognages (PCs) we can obtain due to chance alone using probability theory. Among these are the types that attempt to use the binomial density as in Ringe[1992], and those that calculate the probability of CVC syllables and the like to seem to be cognates due to chance [Bender,1969], [Cowan,1962], [Swadesh,1954], Greenberg [1960]. A thorough review of some of these with corrections, and a discussion of historical linguistics family tree construction can be found in Embleton[1986]. In some unpublished works presented on web pages, we find calculations (based on the binomial density) that up to 400 false cognates may be obtained due to chance. More sophisticated methods than these are necessary. The third type are those that attempt to argue that if the distribution of cognates resembles that of the binomial density then it must be due to chance [Ringe,1995]. A fourth way to attempt to obtain "ball-park" figures on the number of putative cognates (PCs) due to chance is in Hubey[1994]. This paper introduces a method that can be used as a kind of distance similar to the chi-square tests due to Kessler (2001), but does not require assumptions that the comparanda are independent.

Introduction

Science starts with analysis and there are many methods of analyzing data.

> The field of Statistics is constantly challenged by the problems that science and industry brings to its door....With the advent of computers and the information age, statistical problems have exploded both in size and complexity....Vast amounts of data are being generated in many fields, and the statistician's job is to make sense of it all: to extract important patterns and trends, and understand "what the data says". We call this learning from data. The challenges in learning from data have led to a revolution in the statistical sciences. Since computation plays such a key role, it is not surprising that much of this new development has been done by researchers in other fields such as computer science and engineering [Hastie2001]

It seems that as the rest of the data-analysis community has broken itself free of rigid (and simple) statistical methodology, historical linguistics has barely begun the task of using such methods.

> This methodology measures the probabilistic significance of sound correspondences between short word lists. Many rules of thumb invoked by linguists in order to obviate chance resemblances, such as multilateral comparison and emphasizing grammar over vocabulary, are show to actually decrease the power of quantitative tests. While the procedures presented here are straight-

forward, the author also details the extensive linguistic work needed to produce word lists that will not yield nonsensical results [Kessler2001].

The χ^2 Test of Goodness of Fit

Goodness-of-fit tests determine how well a given function (probability density function, pdf, in this case) fits another function. The χ^2 tests for the "extent to which it is unreasonable to assume that the population has a given distribution [pdf]" or extent to which "a set of data cannot be reasonably assumed to be a random sample from a population having a given distribution [pdf]". The test can also be used to test for independence of the random variables. For two variables, the data can be expressed as 2D array (table) [Strait:491]. See also Embleton [1986].

Table 1:

	B_1	B_2	...	B_n	
A_1	$\mathbf{f_{11}}$	$\mathbf{f_{11}}$		$\mathbf{f_{11}}$	$f_{1,x}$
A_2	$\mathbf{f_{11}}$	$\mathbf{f_{11}}$		$\mathbf{f_{11}}$	$f_{2,x}$
	...	...			
	...	...	$\mathbf{f_{ij}}$		
A_m	$\mathbf{f_{11}}$	$\mathbf{f_{11}}$		$\mathbf{f_{11}}$	$f_{m,x}$
	$f_{x,1}$	$f_{x,1}$		$f_{x,1}$	f

The table (contingency table) above is for categorical variables. Then the entry f_{ij} represents the number of items belonging to A_i and B_j. From these we can calculate the marginal frequencies

1) $$f_{i,x} = \sum_{j=1}^{n} f_{ij}; \quad f_{x,i} = \sum_{i=1}^{m} f_{ij} \text{ and } f = \sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}$$

and put the values besides the table as shown. We then compute

2) $$\chi^2 = \sum_{j=1}^{n}\sum_{i=1}^{m} \frac{(f_{ij} - e_{ij})^2}{e_{ij}} \text{ where } fe_{ij} = \frac{f_{i,x} f_{x,j}}{f^2}$$

It can be shown that if A and B are independent random variables and if each value of f_{ij} is sufficiently large, then χ^2 is a value of a random variable whose distribution is approximately χ^2 with

(m-1)(n-1) DOF (degrees of freedom). and reject the null hypothesis that the random variables are independent if $\chi^2 \geq \chi^2_{\alpha,(m-1)(n-1)}$.

Datamining, Knowledge Discovery and Data Analysis

Datamining is based on what was called pattern recognition. One way of classifying the components of pattern recognition is via (i) classification and (ii) estimation. Typically classification is used to create a set of discrete, finite classes, whereas estimation is taken to be an approximation of some desired numerical value based on an observation. The boundaries are not very crisp since estimation consisting of a large number of integer values may just as easily be thought of as categorization or classification. This is especially true if the measured quantities (input data) does not consist of interval or ratio-scaled values [Hubey1999]. Typically a broad-brush classification of the procedures that consist at least of parts of datamining can be strung along a continuum as shown in Fig (1).

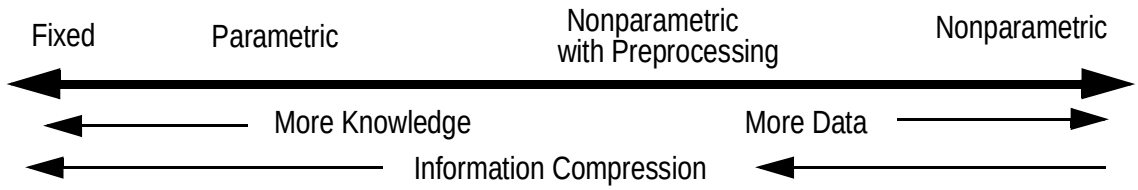

Figure 1: The Modeling Method Continuum:: Fixed models use existing knowledge on a problem (such as in engineering). The nonparametric method relies on a large data set but does not use existing knowledge. The less-well-known aspect of a problem is captured by the nonparametric model.

It may be said that the goal of datamining is to produce domain-knowledge for fields in which there are no models of the type one finds in the ultimate example of a science; physics and its derivatives. Hence the KD (Knowledge Discovery) in KDD (Knowledge Discovery and Datamining). But it is also a "replacement" and improvement of the classical statistical techniques. An informal listing of the classes of data mining procedures would include, Classification/Segmentation/Clustering, Forecasting/Prediction, Association Rule Extraction (knowledge discovery), Sequence Detection. Data Mining Methods may also be classified according to various criteria as: Decision Trees, Rule Induction, Neural Networks, Nearest Neighbor, Genetic Algorithms, Regression Models, Bayesianism, etc. At present datamining sits at the intersection of three broad and converging trends as shown in Fig (2).

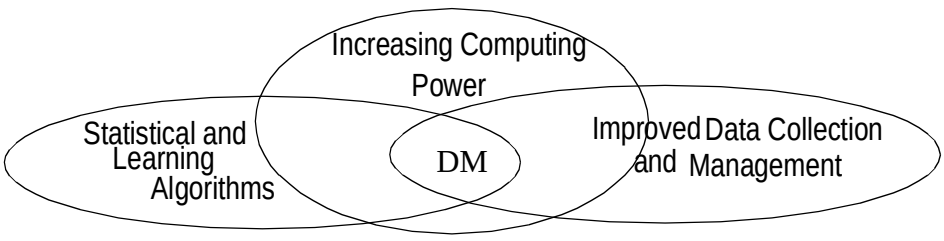

Figure 2: Datamining Algorithm Development

The Problem Space consists of:
Input dimensionality:: the number of components of the input vector
Input space:: the set of allowed input vectors (typically infinite)

Mapping:: the *model*; the function that transforms/maps the inputs to the output, e.g. $\vec{y} = f(\vec{x})$ where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ and $\vec{y} = [y_1, y_2, \ldots, y_m]^T$

Parameter vector::a more accurate model is $\vec{y} = f(\vec{x}|\vec{\Theta})$ where $\vec{\Theta}$ is the parameter vector

Learning algorithm:: generally supervised or nonsupervised learning, which fine-tune the parameters which are a part of the model.

Typically the basis of all datamining is some kind of a *clustering technique* which may serve as a preprocessing, and data reduction technique which may be followed by other algorithms for rule extraction, so that the data can be interpreted for and comprehended by humans. Prediction and classification may be a goal of the process also. There are a set of related problems in the fields of datamining, knowledge discovery, and pattern recognition. We don't know how many neurons should be in the hidden layer or the output layer. Thus if we attempt to use ANNs for clustering as a preliminary method to finding patterns we must use heuristic methods to determine how many clusters the ANN should recognize (i.e. what is the rank/dimension of the output vector). This is just another view of the problem in datamining of knowing how many patterns there are in the data and how we would go about discerning these patterns. There is a related problem in k-nearest-neighbors clustering in which we need an appropriate data structure to be able to efficiently find the neighbors of a given input vector. Indeed, before the k-neighbors method can be used to classify an input vector we need to be able to cluster the training input vectors and an ANN might have been used for this process. The problem of knowing how many patterns (categories or classes/clusters) there are is an overriding concern in datamining, and in unsupervised artificial neural network training.

Clustering

Clustering is the process of grouping data into classes or clusters so that objects within a cluster have high similarity in comparison with one another, but are very dissimilar to objects in other clusters. *Dissimilarities* are assessed based on the attribute values describing the objects. Often *distance measures* are used. Clustering is an unsupervised activity, or should be. Clustering can be thought of as the preprocessing stage for much of datamining. An automated clustering algorithm may be said to be the goal of datamining since *classification* and *prediction* algorithms can work on the clusters. Similarly *association rules* may be derived from the clusters. Clustering can be used by marketers to discover distinct groups of buyers in their customer bases. It can be used to derive taxonomies in biology or linguistics. It can help categorize genes with similar functionality, classify WWW documents for information discovery. In other words, it is a tool to gain insight into the distribution of data. It has been a branch of statistics for years. In machine learning it is an example of unsupervised learning. In datamining active themes for research focus on scalability of clustering methods, the effectiveness of methods for clustering of complex shapes, and types of data, high-dimensional clustering techniques, and methods for clustering mixed numerical and categorical data in large databases. It can be seen that clustering is the basis of datamining, data analysis, and hierarchical-tree building (e.g. language family tree building). Since clustering is based on a concept of distance/dissimilarity, it is not surprising that there are so many different

means of measuring "dissimilarity". The Kullback-Leibler distance is a special and principled case which is reasonably well-suited for interlanguage-distance measurements based on Swadesh-like lists. It is shown below that the goodness-of-fit tests are also based on distances, e.g. distances between probability density functions (pdfs) or probabilty mass functions (pmfs).

Measures of Association, Distance, [Dis]Similarity

There are many different mathematical formulas for measuring distance or similarity. Among these are probabilistic formulas, which may also be considered to be goodness-of-fit measures from the point of view of statistics. Some of these are given below in Table 2. Others can be found in various books [e.g. Hastie2001].

Table 2:

Kullback Leibler Divergence	$D_{KL}(q, r)$	$= \sum q(y)\log\left(\frac{q(y)}{r(y)}\right)$
Jensen-Shannon Divergence	$D_{JS}(q, r)$	$= \frac{1}{2}[D(q, avg(q, r)) + D(r, avg(q, r))]$
Skew Divergence	$D_{\alpha}(q, r)$	$= D(r, \alpha q + (1-\alpha)r)$

Relative Entropy D(p||q) is the distance between two probability densities or mass functions. It is also called the Kullback Leibler distance and is defined as [Cover & Thomas1991]

3) $$D(p||q) = \sum p_i \log\frac{p_i}{q_i} = \int p(x)\log\frac{p(x)}{q(x)}dx$$

where, the convention (based on continuity arguments) $0\log\frac{0}{q} = 0$ and $p\log\frac{p}{0} = \infty$ has been used. For a two dimensional density this is given by

4) $$D(p||q) = \sum\sum p_{ij}\log\frac{p_{ij}}{q_{ij}} = \int\int p(x, y)\log\frac{p(x, y)}{q(x, y)}dxdy$$

It has been used in the form

5) $$D(P||Q) = \sum\sum p_{ij}\log\frac{p_{ij}}{q_{ij}}$$

where P and Q are Markov transition probabilities in the search for gene clusters [Hayes & Borodovsky1998], in composer identification [Jacobliu], scene analysis [Feixas et al] and in a new pairwise clustering [Dubnov et al]. One dimensional versions have seen even more use including speech recognition [Jelinek1997]. The mutual information I(X;Y) is a special case of the relative entropy or the Kullback Leibler distance/divergence for 2D joint pdf and its product distribution e.g.

$$6) \qquad I(X;Y) = \sum\sum p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$$

Clearly, it is asymmetric. It also does not obey the triangular inequality. However, it can be seen that it is a measure of independence of two pdfs and is thus a good alternative to the chi-square tests since the data are not required to be independent. There are methods to fix up the problem of asymmetry of the Kullback Leibler distance or divergence. One simple method is simply to compute the arithmetic or geometric means [Johnson et al]. Other more sophisticated methods are available such as the Ali-Silvey class of information-theoretic distance measures [Johnson et al], or the Jensen-Shannon divergence [Dubnov et al]. These are all related to the Aikake Information Criterion (AIC) [Aikake1973] which figures prominently in study of complexity and statistics [Risanen1989] and data analysis [Hastie2001]. There are other such information criteria. The KL distances are also not normalized. We do not know how large distances can be. Furthermore, for interlanguage comparison we must use mutual information in the form,

$$7) \qquad S(L_p||L_q) = \sum\sum p_{ij}\log\frac{p_{ij}}{p_i q_j}$$

where p_{ij} is the transition-correspondence matrix, and p_i and q_j are the distributions of the phonemes of languages L_p and L_q, hence the use of $S(L_p||L_q)$ (similarity) instead of distance. However since the distance being measured is from the pmf that would occur by chance, for the case that the languages are not related at all the distance measure obtains a zero which means that equation really is a measure of similarity. However, if the two languages are the same language then it does not yield 1 therefore is difficult to normalize the way it is. In the case that $p_{ij} = p_i q_j$ then the correspondences are identical to that which would have occurred by chance, and the distance should be maximum. Therefore the distance should be something like

$$8) \qquad D(L_p||L_q) = e^{-\sum\sum p_{ij}\log\frac{p_{ij}}{p_i q_j}}$$

so that the closer p_{ij} is to $p_i q_j$, the greater distance between the languages and $max(D(L_p||L_q)) = 1$ if the double sum is zero. Unfortunately there are other problems. If any of the $p_{ij} = 0$ then since $\log(0) = -\infty$ we obtain nonsensical answers unless we use $0\log\frac{0}{q} = 0$. One quick fix of this problem is to use smoothed pmf for the data so that

$\forall i,j)(p_{ij} \neq 0)$. However there are still more problems. *What we must do is change the mutual information so that it measures similarity to a diagonal matrix at one extreme and similarity to the hypothetical/random joint pmf, $p_i q_j$ at the other extreme.* Therefore what we should use is something like

$$9) \qquad D(L_p||L_q) = e^{-\sum\sum p_{ij}\log\frac{p_i q_j + p_{ij}}{p_i q_j}} = e^{-\sum\sum p_{ij}\log\left[1+\frac{p_{ij}}{p_i q_j}\right]}$$

One may also interchange p_{ij} and $p_i q_j$ or average the two distances as alternative metrics (vide supra). One can see that $p_i q_j$ are never zero if we do not include phonemes that do not show up in the sample. Therefore, if $p_{ij} = 0$ then since $\log(1) = 0$ these terms do not contribute to the sum, and thus to the distance. We can now compute the extrema of this function so that we can normalize it.

Case I: The languages are not related and therefore $p_{ij} = p_i q_j$

$$10a) \qquad D(L_p||L_q) = e^{-\sum\sum p_{ij}\log\left[1+\frac{p_{ij}}{p_i q_j}\right]} = e^{-\sum\sum p_{ij}\log[2]} = e^{-\ln(2)} = \frac{1}{2} = \Delta_{max}$$

If the $log_2(\ )$ is used (e.g. base 2 logarithm) then it takes a particularly simple form

$$10b) \qquad \Delta_{max} = D(L_p||L_q) = e^{-\sum\sum p_{ij}\log[2]} = e^{-\sum\sum p_{ij}} = e^{-1}$$

Of course, in practice, we will not obtain the exact equality $p_{ij} = p_i q_j$ but only approximate equivalence. If the errors are more or less symmetrically distributed, e.g. if $p_{ij} - p_i q_j$ is symmetrically distributed about zero, and if none of the p_{ij} are zero, then Δ_{max} should be the largest number one can obtain for the two languages in question, and thus should be used in the normalization.

Case II: $L_p = L_q$, therefore $p_{ij} = \delta_{ij} p_i$ where δ_{ij} is the Kronecker delta. Therefore $\frac{\delta_{ij} p_i}{p_i p_i} = \frac{\delta_{ii} p_i}{p_i p_i} = \frac{p_i}{p_i p_i} = \frac{1}{p_i}$ yielding, after splitting the summation into diagonal and nondiagonal terms

$$11) \qquad D(L_p||L_q) = e^{-\left\{\sum_i p_{ii}\log\left[1+\frac{p_{ii}}{p_i p_i}\right] + \sum_i\sum_{j\neq i} p_{ij}\log\left[1+\frac{p_{ij}}{p_i p_j}\right]\right\}} = e^{-\sum_i p_i \log\left[1+\frac{1}{p_i}\right]} = \Delta_{min}$$

Every language when tested against itself will have all nondiagonal entries zero, so that the second sum vanishes yielding the final result. The first sum simplifies to a one-dimensional form as shown. The exponent can be simplified

12) $$\sum_i p_i \log\left[1+\frac{1}{p_i}\right] = \sum_i \log\left[1+\frac{1}{p_i}\right]^{p_i} = \log\left(\prod_i \left[\frac{1+p_i}{p_i}\right]^{p_i}\right)$$

Therefore the final result is

13) $$\Delta_{min} = e^{-\log\left(\prod_i \left[\frac{1+p_i}{p_i}\right]^{p_i}\right)} = \prod_i \left[\frac{1+p_i}{p_i}\right]^{p_i}$$

if we use natural logarithms. The form of the result is suggestive in that probability ratios are weighted by their probability of occurrence by being raised to the exponent. In order to achieve the results in Eqs (11) and (13), the actual phoneme distributions for the languages must come directly from the sample e.g. Swadesh-like lists. If the distribution is taken from the language as a whole, then the sample distribution (e.g. pmf) that comes from the Swadesh-like list will not be identical to it, and the distance metric will have slight errors due to operation of the law of large numbers. Finally, therefore, the normalized distance metric must be of the form

14) $$x_n = \frac{x - x_{min}}{x_{max} - x_{min}}$$

and hence

15) $$D_n(L_p||L_q) = \frac{e^{-\sum\sum p_{ij}\log\left[1+\frac{p_{ij}}{p_i q_j}\right]} - e^{-\sum_i p_i \log\left[1+\frac{1}{p_i}\right]}}{e^{-\sum\sum p_{ij}\log[2]} - e^{-\sum_i p_i \log\left[1+\frac{1}{p_i}\right]}}$$

However the results of the computation for some simple cases (Appendix IV) shows that Eq (15) fails to capture some important aspects of the problem. The metric is not a monotonically increasing measure that hits a maximum at the random case, as it should be. There are other factors must be considered to modify the distance measure. In simple terms, it fails to measure the diagonality of the case when the distance is being measured between identical languages and in which it should yield zero, and it obtains larger numbers than for the random case. The problem is that there are a different number of terms in the sums and these affect the final results. We should find some way to modify these sums depending on the shape of the p_{ij}. Define k as the number of nonzero cells in p_{ij}, n as the number of phonemes being considered (and hence basically a mea-

sure of the size/rank of the matrix), σ, the sum in the exponential term and τ as the trace of the matrix p_{ij}. Then we can see that we have two separate measures of the "degree of diagonality" of the matrix p_{ij}; σ/τ and k/n, or $\sigma-\tau$ and $k-n$. Now, the simplest measure of the "degree of diagonality" of p_{ij} is k, however either k/n or $k-n$ is better since different size p_{ij} will be used in different measurements by different researchers. Of the two k/n is probably better because it is a ratio of the degree of nondiagonality, and it is always positive whereas $k-n$ may need a sign correction. Similarly, the trace, e.g. $\delta_{ij}p_{ij} = p_{11}+p_{22}+\ldots+p_{nn}$ is a measure of how much of the data is along the diagonal. It would be easier if we could find a measure of the degree of "Toeplitzity" of p_{ij}, however lacking such a measure, and realizing that we can always change a Toeplitz matrix into a diagonal one by exchanging columns, we can just as easily operate with a "degree of diagonality" but which is normalized, e.g. σ/τ. If the matrix is diagonal this ratio will be unity. If the ratio is greater than one, it means that the p_{ij} is spread out like a random matrix, and if there is still a tendency toward peakedness about the diagonal, the ratio k/n will catch it. Therefore we need a function $f(k/n, \sigma/\tau)$ or in more general terms $f(k, n, \sigma, \tau)$ that can capture the "diagonality" effects and which can be used to modify the basic entropic distance measure. For now, we will only use a simple co-factor function

15) $$f = \frac{[k/n]^{\alpha}}{[\sigma/\tau]^{\beta}}$$

Thus greater the value of f, the greater the tendency of p_{ij} to be spread out away from diagonality, and hence distance zero. The calculations in the Appendix use $\alpha = \beta = 1$, however there is a principled way in which the values of these constants can be decided. Hence we have the final form of the distance metric

16) $$D_n(L_p||L_q) = \frac{e^{-\frac{[k/n]^{\alpha}}{[\sigma/\tau]^{\beta}}\sum\sum p_{ij}\log\left[1+\frac{p_{ij}}{p_i q_j}\right]} - e^{-\frac{[k/n]^{\alpha}}{[\sigma/\tau]^{\beta}}\sum_i p_i\log\left[1+\frac{1}{p_i}\right]}}{e^{-\frac{[k/n]^{\alpha}}{[\sigma/\tau]^{\beta}}\sum\sum p_{ij}\log[2]} - e^{-\frac{[k/n]^{\alpha}}{[\sigma/\tau]^{\beta}}\sum_i p_i\log\left[1+\frac{1}{p_i}\right]}}$$

There are other caveats, modifications, and simplifications one must consider. Taking a hint from the Ali-Silvey class of measures we can even drop the weighting by the pmf or use some other function instead of the pmf. And because we often wind up with quantities of the sort log(2), we might use logarithms to base 2 which might help with the computations. In any case, if base 2 is used then information is measured in bits, and in nats if the natural logarithm is used. In this case the quantities are normalized so it does not seem to matter which logarithm is used. Furthermore since the sum is sensitive to the number of terms, the same number of terms should be used in all such tests or the sums in the exponents should be normalized for the effects of the number of

terms. The coefficients $\alpha = \beta = 1$ of the cofactor function f() should be calculated, if possible, on a set of languages (real or artificial) such that the triangle inequality of the distance metric is preserved.

Quick experimentation shows that simply using the normalized sum (e.g. divided by the trace) is just as good if not better, therefore

17) $$D_n(L_p||L_q) = \frac{e^{-[k/n]^{\alpha}\left\{\frac{\tau^{\beta}}{\sum\sum p_{ij}\log\left[1+\frac{p_{ij}}{p_i q_j}\right]}\right\}} - e^{-[k/n]^{\alpha}\left\{\frac{\tau^{\beta}}{\sum_i p_i \log\left[1+\frac{1}{p_i}\right]}\right\}}}{e^{-[k/n]^{\alpha}\left\{\frac{\tau^{\beta}}{\sum\sum p_{ij}\log[2]}\right\}} - e^{-[k/n]^{\alpha}\left\{\frac{\tau^{\beta}}{\sum_i p_i \log\left[1+\frac{1}{p_i}\right]}\right\}}}$$

is probably as good a distance measure as Eq (16). Other simpler alternatives may also be found based on the ratios used as exponents in Eq (17).

Language Family Analysis

Language is an extremely complex object of analysis. Even a simple distance metric such as those developed here requires a series of operations in which the researchers have choices. The stages and choices are shown below in Fig (3).

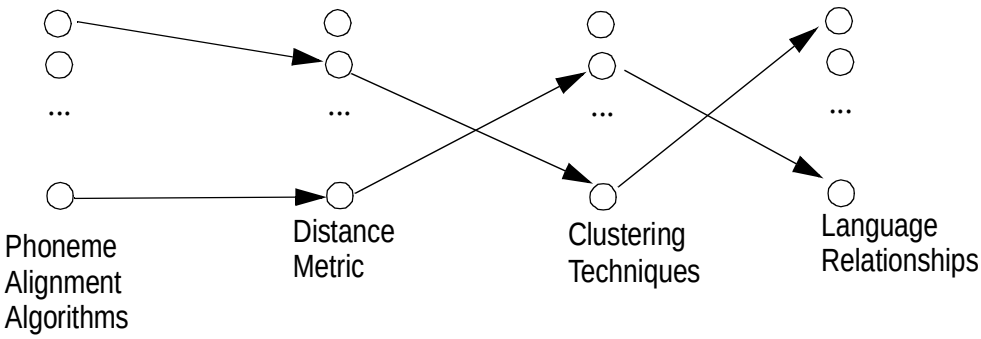

Figure 3: Discovering Language Relationships. The steps that are required to be able to create language relationships such as language family trees, or minimal spanning trees.

Conclusions

It is clear that the concept of distance is paramount in data analysis, clustering, and datamining. Distance allows the use of powerful mathematical techniques. The distance measures and metrics

given above can be used with Swadesh-like lists to derive normalized distances between languages. These distances then can be used with a variety of clustering algorithms to produce a variety relationships amongst languages. More experimentation is needed to more fully understand the properties of the various distances. Only the simplest start has been given here. The distance given here does not require heroic assumptions such as the independence of the comparanda, and is based on information theory. The values of the coefficients can be determined via

References

Ayres, R., *Information, Entropy, and Progress:A New Evolutionary Paradigm*, AIP Press, New York, 1994.

Baldi, P, and Brunak, S. (2001) *Bioinformatics*, MIT Press.

Bender, M. (1969) *Chance CVC correspondences in unrelated languages*, Language, No. 45,1969, pp.519-531.

Benedetto, D. E. Caglioti, and V. Loreto, *Language Trees and Zipping*, arXiv:con-mat/0108530 v2, 19 December 2001.

Brooks, D. and E. Wiley, *Evolution as Entropy: Toward a Unified Theory of Biology*, University of Chicago Press, Chicago, 1988.

Cover, T. and J. Thomas, *Elements of Information Theory*, John Wiley and Sons, Inc, 1991.

Cowan, H. (1962) *Statistical determination of linguistic relationships*, Studia Linguistica 16, pp.57-96.

Dubnov, S., El-Yaniv, R., and Y. Gdalyahu, *A New Nonparametric Pairwise Clustering Algorithm*, http://www.informatik.uni-bonn.de/III/lehre/AG/Mustererkennung/SS00/papcolor.ps

Elrott, K, C. Yang, F., Sladek, and T. Jiang, *Identifying transcription factor binding sites through Markov chain optimization*, *Bioinformatics*, Vol 18, Suppl. 2, 2002, pp. 100-109.

Embleton, S. *Statistics in Historical Linguistics*, Studienverlag Dr. N. Brockmeyer, Bochum, 1986.

Feixas, M, E. del Acebo, P. Bekaert, and M. Sbert, *An information theory framework for the analysis of scene complexity*, Eurographics, Volume 18 (1999) Number 3.

Gershenfeld, N., *The Nature of Mathematical Modeling*, Cambridge University Press, Cambridge, 1999.

Graham, N., *Automatic Detection of Authorship Changes Within Single Documents*, MS Thesis, Computer Science, University of Toronto, 2000.

Greenberg, J. (1960) *A qualitative approach to morphological typology of language*, IJAL, vol 26, No. 3, July 1960, pp.179-194.

Greenberg, J. (1993) *Observations concerning Ringe's "On calculating the factor of chance in language comparison"*. Proceedings of the American Philosophical Society, 137(1): 79-90.

Gorodkin, J. O. Lund, C. Andersen, and S. Brunak, *Using Sequence Motifs for Enhanced Neural Network Prediction of Protein Distance Constraints*, ISMB Proceedings, 1999, 95-105.

Grosse, I., P. Bernaola-Galvan, P. Carpena, R. Roman-Roldan, J. Oliver, H. Stanley, *Analysis of symbolic sequences using the Jensen-Shannon divergence*, Physical Review E, vol 65, 04 1905.

Hastie, T, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning:datamining, inference and prediction*, Springer, New York, 2001.

Hayes S, and M. Borodovsky, *How to Interpret an Anonymous Bacterial Genome: Machine Learning Approach to Gene Identification*, Genome Research, 8:1154-1171, 1998.

Hertz, G, and G. Stormo, *Identifying DNA and protein patterns with statistically significant alignments of multiple sequences*, Bioinformatics, Vol 15, nos 7/8 1999, pp 563-577.

Hock, H.H., (1991) *Principles of Historical Linguistics*, Mouton De Gruyter.

Holmes, I., Studies in Probabilistic Sequence Alignment and Evolution, PhD Thesis, University of Cambridge, Cambridge, 1998.

Hubey, H.M. (1999) *Vector Phonological Spaces via Dimensional Analysis*, Journal of the International Quantitative Linguistics Association, 1999

Hubey, H.M. (1998) *Quantitative Methods in Linguistics with an Application to *PIE,* History of Language, September 1998.

Hubey, H.M., *Mathematical Foundations of Linguistics*, Lincom Europa, Muenchen, Germany, 1999.

Hubey, H.M. *Computational and Mathematical Linguistic*s, Mir Domu Tvoemu, Moscow, Russia, 1994. (Also published by Lincom Europa, 1999).

Hunstberger, D., P. Billingsley, *Elements of Statistical Inference,* Allyn & Bacon, Boston, MA, 1977.

Jazwinski, A. *Stochastic Processes and Filtering Theory,* Academic Press, New York, 1970.

Jelinek, F. *Statistical Methods for Speech Recognition,* MIT Press, Cambridge, MA, 1997.

Johnson

Juola, P. Cross Entropy and Linguistic Typology, In D.M.W. Powers (ed.) *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning,* ACL, pp 141-149.

Kessler, B. *The Significance of Word Lists*, CSLI Publications, Stanford, CA, 2001.

Kondrak, G. Algorithms for Language Reconstruction, PhD Thesis, University of Toronto, Canda, kondrak@cs.toronto.edu.

Kondrak, G. *Identifying Cognates by Phonetic and Semantic Similarity,* kondrak@cs.toronto.edu.

Kondrak, G. *Determining Recurrent Sound Correspondences by Inducing Translation Models*, kondrak@cs.toronto.edu.

Kontoyiannis, I., P. Algoet, Yu. Suhov, and A. Wyner, *Nonparametric Entropy Estimation for Stationary Processes and Random Fields, with Applications to English Text*, IEEE Transactions on Information Theory, Vol. 44, No. 3, May 1998, 1319-1327.

Lamb, S.M. and E.D. Mitchell (1991) *Sprung from Some Common Source: Investigations Into The Prehistory of Languages*, Stanford University Press, 1991.

Lee, L. *On the Effectiveness of the Skew Divergences for Statistical Language Analysis*, Artificial Intelligence and Statistics, 2001, pp. 65-72.

Leff, H, and A.Rex, *Maxwell's Demon: Entropy, Information, and Computing*, Princeton University Press, Princeton, NJ, 1990.

Li, M, and P. Vitanyi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 1993.

Liu, Jacob (Yi-Wen), *Modeling music as Markov chains - composer identification*, http://ccrma-www.stanford.edu/~jacobliu/254report/ Composer_identification_com.html

Manning, C. and H. Schutze, *Foundations of Statistical Natural Language Processing,* MIT Press, Cambridge, MA, 1999.

Needleman, S. B., Wunsch, C. D., J. Mol. Biol. (1970) 48:443-453

Nerbonne, J. and P. Kleiweg, *A Dialectical Yardstick,* Quantitative Linguistics Conference 2003, University of Georgia, May 29, 2003.

Nichols, J. (1992) *Linguistic Diversity in Space and Time*, University of Chicago Press, 1992.

Nikias, C. and A. Petropulu, *Higher Order Spectra Analysis: A Nonlinear Signal Processing Framework*, Prentice-Hall, Englewood Cliffs, 1993.

Nowak, M, Komarova, N, and P. Niyogi, *Computational and Evolutionary Aspects of Languag*e, Nature, Vol, 4, 6 June 2002, pp. 611-617

Papadimitrou, C., *Computational Complexity,* Addison-Wesley, Reading, MA, 1994.

Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 1984.

Pickrell, J. *Searching for the Tree of Babel*, Science, 25 May 2002.

Ringe, D. (1992) *On Calculating the Factor of Chance in Language Comparison*, The American-Philosophical Society. Transactions of the American Philosophical Society, vol. 82, Phil.

Ringe, D. (1995) *Nostratic and the Factor of Chance*, Diachronica XII:1.55-74.

Rissanen, J. *Stochastic Complexity in Statistical Inquiry*, World Scientific, Singapore, 1989.

Salmons, J., and B. Joseph (eds), (1998) *Nostratic: Sifting the Evidence*, John Benjamins.

Segal, E. and D. Koller, *Probabilistic Hierarchical Clustering for Biological Data*, RECOMB'02, April 18-21, 2002, Washington, DC, ISBN 1-581 12-498-3-02/04.

Sjolander, K. *Automatic Alignment of phonetic segments*, Lund University, Dept. of Linguistics, Working Papers 49 (2001), 140-143.

Smith, T. F., Waterman, M. S., J. Mol. Biol. (1981) 147:195-197.

Soong, T.T. *Random Differential Equations in Science and Engineering*, Academic Press, New York, 1973.

Strait, P. *Probability and Statistics with Applications*, Harcourt Brace Jovanovich, NYC, 1983.

Vihola, M., *Dissimilarity Measures for Hidden Markov Models and Their Application in Multilingual Speech Recognition*, MS Thesis, Tampere University of Technology, Finland.

Williams, G (1978) *Chaos Theory Tamed*, Joseph Henry Press, Washington, D.C

Zhen, W. *Entropic Approach for Reduction of Amino Acid Alphabets*, arXive.physics/0106074 v1 22 June 2001.

Zuraw, Kie, (2003) *Probability in Language Change, in Probabilistic Linguistics*, (ed) Bod, Hay and Jannedy. MIT Press.

Appendix 0: Phoneme Alignment

The phonemes of words in Swadesh-like lists have to be aligned before the correspondence-matrix can be used. Genetics algorithms such as the Needleman-Wunsch[1970], and Smith-Waterman[1981] algorithms can be used. The distance between phonemes can then be computed using various distances e.g. Levenstein "edit distance" [see for example Nerbonne , Heeringa]. These use dynamic programming techniques to calculate the alignment score S(i,j) e.g. you only need to enumerate and score all ways in which one aligned pair can be added to a shorter alignment to produce an alignment of the first i residues of sequence-1 and the first j residues of sequence-2. All possible pairs are represented by a two-dimensional array, and all possible comparisons are represented by pathways through this array. The three main steps are:

1. Assign similarity scores: A numerical value (score) is assigned to every cell in the array depending on similarity/dissimilarity. Similarity scores may be simple, or related to chemical similarities or frequency of observed substitutions

2. Score pathways through array: For each cell we calculate the maximum possible score for an alignment ending at that point. The cumulative score is calculated by adding in a path through the matrix. Subrows and subcolumns are searched for the highest score. The gap is penalty independent of the length of the gap. The best match is the pathway with the highest score. The maximum match is largest number of amino acids of one protein that can be matched with those of another protein while allowing for all possible deletions

3. Construct the alignment

There is an online version of the Needleman-Wunsch algorithm however it seems to only work with alphanumeric characters:

http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/adp_NeedlemanWunsch

The Smith-Waterman algorithm (1981) is based on the NW algorithm but instead of looking at each sequence in its entirety, this compares segments of all possible lengths and chooses whichever optimise the similarity measure (local alignments). Whereas the NW algorithm finds the best global alignment, the SW algorithm it works by comparing segments of all possible lengths (LOCAL alignments) and chooses whichever maximise the similarity measure.

The similarity scoring might have +1 for a match and 0 or -1 for a mismatch. Gap penalties are also introduced. Using very high gap penalties, the words in the Swadesh-like lists can be concatenated into one big sequence, and the long sequences can be compared using both the NW and SW algorithms. More detail on these algorithms especially as applied to linguistics can be found in Kondrak, Nerbonne, Heerenga.

Appendix I Markov Chains and Entropy of English

Let $\{x_1, x_2, \ldots, x_n\}$ be a sequence of random variables taking on values in some finite alphabet. If nothing more is known about this process, then the Bayes' relationship holds

I.1) $$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1})$$

However, if

I.2) $$p(x_i | x_1, x_2, \ldots, x_{i-1}) = p(x_i | x_{i-1})$$

then the variables are said to form a Markov chain, which is to say that the process that produced this sequence or time series was a Markov or Markovian process. The relationship can be iterated to give

I.3) $$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_{i-1})$$

Markovian processes have the simplest memory. For the purposes of using the concept of RSC (recurrent sound change) in Swadesh-like lists we would have to use phonemes instead of the English alphabet as done in Cover & Thomas. The frequency of phonemes in every language has to have a distribution. The frequency of phoneme-pairs also has a joint distribution, and a conditional distribution (which is necessary for a Markov model). Ditto for higher order conditional densities. To build a third-order Markov model (using letters, not phonemes) we must estimate the values of $p(x_i | x_{i-1} x_{i-2} x_{i-3})$. This table will have $(N+1)^4$ entries, where N is the number of phonemes and to make a reasonably accurate estimate of these probabilities we would need to process many millions of words written in a phonemic alphabet. There are examples of Markov approximations to English (using letters) from Shannon's original paper [Cover & Thomas]

1. Zeroth-order approximation: independent and equiprobable symbols
XFOML RXKHRJFFJUJ ZLPWCFWKCYJ
2. First-order approximation: independent symbols with frequency matching English letters
OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI
3. Second-order approximation: the frequency of pairs of letters matches English text
ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY
4.Third-order approximation: the frequency of triplets matches English text
IN NO IST LAT WHEY CRATICT FROURE BERS GROCID
5. Fourth-order approximation: frequency of quadruplets matches English text
THE GENERATED JOB PROVIDUAL BETTER TRAND THE DISPLAYED CODE, ABOVERY UPONDULTS WELL THE CODERST IN THESTICAL IT DO HOCK BOTHE MERG.

Now, when we construct a transition matrix to match up CVC syllables of language A to language B, we not only have to list the phonemes in the first place but also in the second, and third. Finally we also have to list the pairs and triplets, and quadruplets to be able to account for the conditioned sound changes that may have occurred.

Correspondence-Transition Matrix for Distance Metric Calculations

.....

Appendix II Tensor, pmf, and Notation

Arrays and scalars are very useful in linguistics. Two-dimensional arrays are treated as arrays of arrays and so on. Arrays in other settings are called vectors. Two dimensional arrays, similarly, are called matrices. There are many different types of notations for mathematics in which vectors, scalars and matrices are used. Similarly, typically an uppercase Latin letter may be used for matrices, and Greek letters for scalars in linear algebra courses. Various other notations are used in the literature for vector. It may be represented as a bold lowercase Latin letter, such as **x**, or it may be represented as a letter with a little arrow on top $\vec{x}$, $\vec{x}$ or an underscribble $\underline{x}$.

There are several problems with the *bold notation* for vectors. It is difficult in handwriting to use bold letters. The arrow or underscribble notation is reasonably good for handwriting however it causes confusion when functions, matrices, scalars are used together. For example a scalar function of a vector would be written as f(**x**) whereas a vector function of a vector would be **g(x)**. There is a very simple and fruitful way to combine all of these together. These structures are all tensors. A *scalar* is a tensor of rank zero. A *vector* is a tensor of rank one. A *matrix* is a [mixed] tensor of rank two. A tensor of rank *n* is represented simply as a indexed [sub/superscripted] variable with *n* indices. A scalar, therefore, has no indexing. A vector $\boldsymbol{x}$ is represented simply as x_k. There is a subtle difference of the meaning of the notation different settings. The notation x_k is used for both the whole vector or a particular component of it i.e. the kth component of the tensor (vector) x_k. Which is meant is usually clear from the context. It should be noted that the index in this particular usage is a *free index* and therefore it is a *dummy index* (i.e. it can be anything). However when tensor products are being formed careful attention must be paid to the selection of indices since they are significant in some cases.

Normally, to show inner (dot), outer, vector, and cross products, and also to conserve space one must resort to writing vectors by using transposes. Similarly, to show the dot product using indices one uses the summation sign whereas in reality it is simply redundant. The suppressed summation notation or convention of tensors is traced back to Einstein [Butkov,1968:679]. In linear algebra books where vectors are basically considered to be special classes of matrices ($1 \times n$ or $n \times 1$ matrices), and where matrix products do not commute and where transposes of matrices are commonplace it becomes important to distinguish between row vectors and column vectors. In many physics and engineering texts where is usually no need to discriminate between column vectors and row vectors, the notation above is sufficient. A tensor is invariant to rotation of the underlying Cartesian space. However, here we only basically use the notation for simplification of the concepts. An inner (or dot) product of two vectors is written as

II.1) $$\mathbf{q}= \mathbf{x} \bullet \mathbf{y}= \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$$

Here whether the vectors are column vectors or row vectors is immaterial. However in many books, especially those in linear algebra, the distinction must be made. The dot (inner) product would then be written as $\boldsymbol{p}=\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{y}$ or in longer form as

II.2) $$\boldsymbol{x^T y} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$$

In the suppressed summation notation, the convention requires summation over a repeated index which is used in ways similar to the dummy integration variable. The tensor notation can be used for stylistic purposes even if the quantities do not transform like covariant or contravariant tensors. Some of the quantities here may not transform like tensors, so only the suppressed summation convention of tensors and the subscript notation is used for stylistic purposes. In tensor notation (using the suppressed summation convention) the dot product of the two tensors of rank one x_i, and y_j is simply $x_j y_j$ or $x_i y_i$ or even $x_k y_k$. Thus a vector product such as $\boldsymbol{y x^T}$ is

II.3) $$\boldsymbol{y x^T} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} = \begin{bmatrix} y_1 x_1 & y_1 x_2 & \dots & y_1 x_n \\ y_2 x_1 & y_2 x_2 & \dots & y_2 x_n \\ \dots & \dots & \dots & \dots \\ y_n x_1 & y_n x_1 & \dots & y_n x_n \end{bmatrix}$$

in tensor notation is simply $x_j y_k$, or equivalently $y_k x_j$, (or even $y_m x_k$) clearly a two dimensional tensor in both cases. Here it is required that the two vectors have different subscripts because there is something like a scoping rule that necessitates the use of matching of subscripts when tensors are used in equations. The order of the tensors in the product is immaterial since both products expand to

II.4) $$\begin{bmatrix} x_1 y_1 & x_1 y_2 & \dots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \dots & x_2 y_n \\ \dots & \dots & \dots & \dots \\ x_n y_1 & x_n y_2 & \dots & x_n y_n \end{bmatrix} \text{ or } \begin{bmatrix} y_1 x_1 & y_1 x_2 & \dots & y_1 x_n \\ y_2 x_1 & y_2 x_2 & \dots & y_2 x_n \\ \dots & \dots & \dots & \dots \\ y_n x_1 & y_n x_1 & \dots & y_n x_n \end{bmatrix}$$

respectively. The order of the factors in scalar multiplication is obviously not significant. The easiest way to relate vector product (outer product) of vectors is to think of column and row vectors as being parts of matrices as in

II.5) $$\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \begin{bmatrix} y_1, y_2, \dots, y_m \end{bmatrix} = \begin{bmatrix} x_1 & 0 & \dots & 0 \\ x_2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ x_n & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \dots & y_m \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \dots & x_1 y_m \\ x_2 y_1 & x_2 y_2 & \dots & x_2 y_m \\ \dots & \dots & x_j y_i & \dots \\ x_n y_1 & x_n y_2 & \dots & x_n y_m \end{bmatrix}$$

The representation of a product of a matrix by a vector (i.e. **Ax**) is $v_j=a_{jk}x_k$ showing that there is a summation over the repeated index (the suppressed summation convention). The result is a tensor of rank 1 (vector) with the free index j. An expansion of this product for a specific index is

II.6) $$v_j = \sum_{k=1}^{n} a_{jk}x_k = a_{j1}x_1 + a_{j2}y_2 + \ldots + a_{jn}y_n$$

We can expand this for every *j* (the index of the vector) so the result is clearly a vector. We can combine summation implied over a repeated index, with the Kronecker delta defined as

II.7) $$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Therefore the condition of orthonormality of two vectors is simply $u_i \cdot u_j = \delta_{ij}$. Thus the trace of a matrix b_{ij} is simply $\delta_{ij}b_{ij}$. The trace can simply be written as the contraction b_{ii} or b_{jj} or any dummy index. We can use the Kronecker delta to sum over an index, such as

II.8) $$\delta_{11}x_1 + \delta_{22}x_2 + \ldots + \delta_{nn}x_n = x_1 + x_2 + \ldots + x_n = \sum_{j=1}^{n} x_j = \delta_{ij}x_i$$

Therefore the Kronecker delta can be used to even allow inline formatting of summation. Now, it is just as easy to treat pmf's as tensors or arrays, so that p_iq_j is simply a tensor of rank 2 e.g.

II.9) $$p_iq_j = \begin{bmatrix} p_1 \\ p_2 \\ \ldots \\ p_n \end{bmatrix} \begin{bmatrix} q_1, q_2, \ldots, q_m \end{bmatrix} = \begin{bmatrix} p_1q_1 & p_1q_2 & \cdots & p_1q_m \\ p_2q_1 & p_2q_2 & \cdots & p_2q_m \\ \cdots & \cdots & p_jy_i & \cdots \\ p_nq_1 & p_nq_2 & \cdots & p_nq_m \end{bmatrix}$$

Appendix III: Linear Time Series, Cross-Correlation and the Oswalt Shift Test

It just so happens that the definition of auto/cross-correlation in continuous math is defined as

III.1a) $$R_{xy}(\tau) = \int x(s-\tau)y(s)ds$$

With the substitution s=z+t (and ds=dz) this becomes

III.1b) $$R_{xy}(\tau) = \int x(z)y(z+\tau)dz$$

So the higher order correlations of some function x(t) are then computed as

III.2) $$\iint x(t)x(t-s)x(t-u)dsdu$$

In general signals are first discretized/digitized before processing so that we need to consider systems that are discrete. The most general linear system produces an output y that is a linear function of external inputs to x and its previous outputs

III.3) $$y_t = a_t + \sum_{m=1}^{M} b_m y_{t-m} + \sum_{n=0}^{N} c_n x_{t-n}$$

Typically a_t term is nonzero which imposes an *initial conditions* (IC) on the system. The two parts of the rhs are called by different names in different fields. In statistics, the first summation is called an *autoregressive* (AR) whereas in engineering, signal processing, it is called an *infinite impulse response* (IIR) meaning that the output can continue after the input stops. The second term is called a *moving average* (MA) in statistics and *finite impulse response* (FIR) in engineering. Together, the system is an *autoregressive moving average system of order (M,N)* (i.e ARMA(M,N)). The autocorrelation function is defined to be

III.4) $$\kappa_\tau \equiv \frac{\langle (y_t-\mu_y)(y_{t-\tau}-\mu_y)\rangle}{\langle (y_t-\mu_y)(y_t-\mu_y)\rangle} = \frac{\langle y_t y_{t-\tau}\rangle - \mu_y\langle y_{t-\tau}\rangle - \mu_y\langle y_t\rangle + \mu_y\mu_y}{\langle y_t y_t\rangle - \mu_y\langle y_t\rangle - \mu_y\langle y_t\rangle + \mu_y\mu_y} = \frac{\langle y_t y_{t-\tau}\rangle - \mu_y^2}{\langle y_t y_t\rangle - \mu_y^2}$$

where μ_y is the mean, and σ_y^2 is the variance. The last line derives from the fact that for stationary processes the time averages are independent of the time origin. In some books a distinction is made between *autocorrelation* function and *autocovariance* function. The autocovariance function is the centralized or normalized version of the simple autocorrelation function.

The values of the autocorrelation function is $-1 \le \kappa \le 1$. For specific cases the value of κ_τ can be computed in simpler ways. For example, for zero-mean MA processes (a=b=0), the autocorrelation function is

III.5) $$\kappa_\tau = \frac{(\sum c_n x_{t-n})(\sum c_p x_{t-\tau-p})}{(\sum c_n x_{t-n})(\sum c_r x_{t-r})}$$

For the AR model, we can multiply both sides by $y_{t-\tau}$ and average to obtain

III.6) $$\langle y_t y_{t-\tau} \rangle = \sum_{m=1}^{M} b_m \langle y_{t-m} y_{t-\tau} \rangle$$

which after normalizing by the variance gives

III.7) $$\kappa_\tau = \sum_{m=1}^{M} b_m \kappa_{\tau-m}$$

This linear set of equations is called the *Yule-Walker* equations. It need not be zero after M steps, unlike the MA case. The equations can be inverted to related the AR coefficients to the autocorrelation function. An efficient method to accomplish this is to use the *Levinson-Durbin recursion*. There is no unique method to find the best ARMA model to describe a data set (unlike the simpler AR and MA models). A popular procedure is the *Box-Jenkins recursive solution*. There is some leeway in trading off M vs N in selecting the order of the ARMA model.

One can now combine the cross-correlation function idea, and the alignment algorithm ideas from genetics to show what the Oswalt test attempts to do in some way.

.....

Appendix IV: Entropy and Information

Correlation functions, which are essential for measuring dependencies in a linear system, are nearly useless for nonlinear systems because signals from even simple nonlinear systems can have broadband power spectra and hence a featureless correlation structure. Information-theoretic quantities provide an elegant alternative that captures the essential features of a correlation function, and more. *Information* of a system x is defined as

IV.1) $$I_w = -\sum_{i=1}^{N_s} P_i \ln(P_i)$$

where N_s is the number of nonzero probabilities, and P_i is the probability of the ith bin. The [self] entropy for system x is defined as

IV.2) $$H_x = -\sum_{i=1}^{N_s} P(x_i)\ln[P(x_i)]$$

In other words it is the average of the logarithm of the pdf. Then the cross-entropy is given as

IV.3) $$H_{xy} = -\sum_{i=1}^{N_s}\sum_{j=1}^{N_s} P(x_i y_j)\ln[P(x_i y_j)]$$

If the systems are independent then $P(x_i y_j) = P(x_i)P(y_j)$ therefore

IV.4)
$$H_{xy} = -\sum_{i=1}^{N_s}\sum_{j=1}^{N_s} P(x_i)P(y_j)\{\ln[P(x_i)] + \ln[P(y_j)]\} =$$
$$= -\sum_{i=1}^{N_s} P(x_i)\ln[P(x_i)]\sum_{j=1}^{N_s} P(y_j) - \sum_{j=1}^{N_s} P(y_j)\ln[P(y_j)]\sum_{i=1}^{N_s} P(x_i) = H_x + H_y$$

But we need to do this with conditional probabilities. We can write the probability equations as

IV.5) $$P(x_i y_j) = P(x_i)P(y_j|x_i)$$

Therefore

IV.6) $$H_{xy} = -\sum_{i=1}^{N_s}\sum_{j=1}^{N_s} P(x_i y_j|x_i)\ln[P(x_i y_j|x_i)]$$

From this, as before we obtain

$$H_{xy} = -\sum_{i=1}^{N_s}\sum_{j=1}^{N_s} P(x_i)P(y_j|x_i)\{\ln[P(x_i)] + \ln[P(y_j|x_i)]\} =$$

IV.7)

$$= -\sum_{i=1}^{N_s} P(x_i)\ln[P(x_i)]\sum_{j=1}^{N_s} P(y_j|x_i) - \sum_{j=1}^{N_s} P(y_j|x_i)\ln[P(y_j|x_i)]\sum_{i=1}^{N_s} P(x_i)$$

Therefore

IV.8) $$H_{xy} = -\sum_{i=1}^{N_s} P(x_i)\ln[P(x_i)] - \sum_{j=1}^{N_s} P(y_j|x_i)\ln[P(y_j|x_i)] = H_x + H_{y|x}$$

The quantity $H_{y|x}$ is the *conditional entropy.* Because of symmetry we can also derive the relationship

IV.9) $$H_{xy} = H_y + H_{x|y}$$

From these two we can derive the equality $H_y + H_{x|y} = H_x + H_{y|x}$ or $H_y - H_{y|x} = H_x - H_{x|y}$. This new quantity is called *mutual information*

IV.10) $$I_{xy} = H_y - H_{y|x} = H_x - H_{x|y}$$

The conditional entropy $H_{y|x}$ represents an amount of information about Y. The basic uncertainty H_y is lessened by an amount equal to $H_{y|x}$. Therefore we can write

IV.11) $$I_{xy} = H_y - [H_{xy} - H_x] = H_x + H_y - H_{xy}$$

which is similar to the union operation of sets. It may also be written as

IV.12) $$H_{xy} = H_x + H_y - I_{x,y}$$

We can show also that

IV.13) $$I_{xy} = \sum_{i}^{N_s}\sum_{j}^{N_s} P(x_i y_j)\ln\left\{\frac{P(x_i y_j)}{P(x_i)P(y_i)}\right\}$$

Thus this relationship is something that does a job similar to that of correlation functions or association functions.

Appendix V Initial Exploratory Metric Computations Using Excel

P(i)Q(j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1000	0.0048	0.0095	0.0143	0.0190	0.0238	0.0286
0.1500	0.0071	0.0143	0.0214	0.0286	0.0357	0.0429
0.2000	0.0095	0.0190	0.0286	0.0381	0.0476	0.0571
0.2500	0.0119	0.0238	0.0357	0.0476	0.0595	0.0714
0.3000	0.0143	0.0286	0.0429	0.0571	0.0714	0.0857

Case I ***Simulation--almost diagonal***

P(i,j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1000	0.1000	**0.2000**				
0.1500			**0.1000**		0.1000	
0.2000				**0.2000**		
0.2500			0.1000		**0.1000**	
0.3000						**0.1000**

ln(1+p(i,j)/p(i)q(j)

3.0910	3.0910	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	1.7346	0.0000	1.3350	0.0000
0.0000	0.0000	0.0000	1.8326	0.0000	0.0000
0.0000	0.0000	1.3350	0.0000	0.9858	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.7732

p(i,j)*ln(1+p(i,j)/p(i)q(j)

0.0147	0.0294	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0372	0.0000	0.0477	0.0000
0.0000	0.0000	0.0000	0.0698	0.0000	0.0000
0.0000	0.0000	0.0477	0.0000	0.0587	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0663

distance= **0.6897**

P(i)Q(j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1000	0.0048	0.0095	0.0143	0.0190	0.0238	0.0286
0.1500	0.0071	0.0143	0.0214	0.0286	0.0357	0.0429
0.2000	0.0095	0.0190	0.0286	0.0381	0.0476	0.0571
0.2500	0.0119	0.0238	0.0357	0.0476	0.0595	0.0714
0.3000	0.0143	0.0286	0.0429	0.0571	0.0714	0.0857

Case II P(i,j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1000	**0.2000**	0.0000	0.0000	0.0000	0.0000	0.0000
0.1500	0.0000	**0.3000**	0.0000	0.0000	0.0000	0.0000
0.2000	0.0000	0.0000	**0.3000**	0.0000	0.0000	0.0000
0.2500	0.0000	0.0000	0.0000	**0.1000**	0.0000	0.0000
0.3000	0.0000	0.0000	0.0000	0.0000	**0.1000**	0.0000

ln(1+p(i,j)/p(i)q(j)

9.0849	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	**7.2937**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**5.9094**	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	**3.8089**	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	**3.0253**	0.0000

0.0433	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	**0.1042**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**0.1688**	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	**0.1814**	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	**0.2161**	0.0000

distance= **0.4898**

P(i)P(j)

	0.0476	**0.0952**	**0.1429**	**0.1905**	**0.2381**	**0.2857**
0.0476	0.0023	0.0045	0.0068	0.0091	0.0113	0.0136
0.0952	0.0045	0.0091	0.0136	0.0181	0.0227	0.0272
0.1429	0.0068	0.0136	0.0204	0.0272	0.0340	0.0408
0.1905	0.0091	0.0181	0.0272	0.0363	0.0454	0.0544
0.2381	0.0113	0.0227	0.0340	0.0454	0.0567	0.0680
0.2857	0.0136	0.0272	0.0408	0.0544	0.0680	0.0816

Simulated P(i,j)

	0.0476	**0.0952**	**0.1429**	**0.1905**	**0.2381**	**0.2857**
0.0476	**0.0476**					
0.0952		**0.0952**				
0.1429			**0.1429**			
0.1905				**0.1905**		
0.2381					**0.2381**	
0.2857						**0.2857**

3.0910	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	**2.4423**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**2.0794**	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	**1.8326**	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	**1.6487**	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	**1.5041**

0.1472	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	**0.2326**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**0.2971**	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	**0.3491**	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	**0.3925**	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	**0.4297**

distance= **0.1575**

ln(1+1/j3)--:	3.0910					
		2.4423				
j32*j3-->	0.1472		2.0794			
		0.2326		1.8326		
			0.2971		1.6487	
				0.3491		1.5041
					0.3925	
	Note the result-->		**0.1575**			0.4297

Simulated q(i)q(i)

	0.1000	***0.1500***	***0.2000***	***0.2500***	***0.3000***
0.1000	0.0100	0.0150	0.0200	0.0250	0.0300
0.1500	0.0150	0.0225	0.0300	0.0375	0.0450
0.2000	0.0200	0.0300	0.0400	0.0500	0.0600
0.2500	0.0250	0.0375	0.0500	0.0625	0.0750
0.3000	0.0300	0.0450	0.0600	0.0750	0.0900

Simulated P(i,j)

	0.1000	***0.1500***	***0.2000***	***0.2500***	***0.3000***
0.1000	**0.1000**				
0.1500		**0.1500**			
0.2000			**0.2000**		
0.2500				**0.2500**	
0.3000					**0.3000**

2.3979	0.0000	0.0000	0.0000	0.0000
0.0000	**2.0369**	0.0000	0.0000	0.0000
0.0000	0.0000	**1.7918**	0.0000	0.0000
0.0000	0.0000	0.0000	**1.6094**	0.0000
0.0000	0.0000	0.0000	0.0000	**1.4663**

0.2398	0.0000	0.0000	0.0000	0.0000
0.0000	**0.3055**	0.0000	0.0000	0.0000
0.0000	0.0000	**0.3584**	0.0000	0.0000
0.0000	0.0000	0.0000	**0.4024**	0.0000
0.0000	0.0000	0.0000	0.0000	**0.4399**

distance= **0.1745**

Theoretical **P(i)Q(j)**

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***	sums
0.1	0.0048	0.0095	0.0143	0.0190	0.0238	0.0286	***0.1***
0.15	0.0071	0.0143	0.0214	0.0286	0.0357	0.0429	***0.15***
0.2	0.0095	0.0190	0.0286	0.0381	0.0476	0.0571	***0.2***
0.25	0.0119	0.0238	0.0357	0.0476	0.0595	0.0714	***0.25***
0.3	0.0143	0.0286	0.0429	0.0571	0.0714	0.0857	***0.3***
sums	***0.0476***	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***	

Simulation of P vs Q **P(i,j)** (random)
Simply perturb the P(i)Q(j) above

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1	0.0048	0.0095	0.0143	0.0190	0.0238	0.0286
0.15	0.0071	0.0143	0.0214	0.0286	0.0357	0.0429
0.2	0.0095	0.0190	0.0286	0.0381	0.0476	0.0571
0.25	0.0119	0.0238	0.0357	0.0476	0.0595	0.0714
0.3	0.0143	0.0286	0.0429	0.0571	0.0714	0.0857
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.0033	0.0066	0.0099	0.0132	0.0165	0.0198
	0.0050	0.0099	0.0149	0.0198	0.0248	0.0297
	0.0066	0.0132	0.0198	0.0264	0.0330	0.0396
	0.0083	0.0165	0.0248	0.0330	0.0413	0.0495
	0.0099	0.0198	0.0297	0.0396	0.0495	0.0594

Distance= **0.5**

Appendix VI Final Distance Metric Computations

P(i)Q(j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1000	0.0048	0.0095	0.0143	0.0190	0.0238	0.0286
0.1500	0.0071	0.0143	0.0214	0.0286	0.0357	0.0429
0.2000	0.0095	0.0190	0.0286	0.0381	0.0476	0.0571
0.2500	0.0119	0.0238	0.0357	0.0476	0.0595	0.0714
0.3000	0.0143	0.0286	0.0429	0.0571	0.0714	0.0857

Case I Simulation--almost diagonal

P(i,j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1000	**0.1000**	**0.2000**				
0.1500			**0.1000**		**0.1000**	
0.2000				**0.2000**		
0.2500			**0.1000**		**0.1000**	
0.3000						**0.1000**

total==> 1.0000

ln(1+p(i,j)/p(i)q(j)

3.0910	**3.0910**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**1.7346**	0.0000	**1.3350**	0.0000
0.0000	0.0000	0.0000	**1.8326**	0.0000	0.0000
0.0000	0.0000	**1.3350**	0.0000	**0.9858**	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	**0.7732**

p(i,j)*ln(1+p(i,j)/p(i)q(j)

0.0147	**0.0294**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**0.0372**	0.0000	**0.0477**	0.0000
0.0000	0.0000	0.0000	**0.0698**	0.0000	0.0000
0.0000	0.0000	**0.0477**	0.0000	**0.0587**	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	**0.0663**

sum=> 0.3715 s/r-ratio 1.36621628

*pseudo trace==> 0.2719 k/n-ratio 3.75

count==> 8

old distance=> **0.689733** **new** **0.360754**

0.064261

0.064261

* the matrix is not aligned correctly, therefore the largest numbers in each row was used, since this is what the trace would have been if the alignment had been correct.

P(i)Q(j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***
0.1000	0.0048	0.0095	0.0143	0.0190	0.0238
0.1500	0.0071	0.0143	0.0214	0.0286	0.0357
0.2000	0.0095	0.0190	0.0286	0.0381	0.0476
0.2500	0.0119	0.0238	0.0357	0.0476	0.0595
0.3000	0.0143	0.0286	0.0429	0.0571	0.0714

Case II P(i,j)

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***
0.1000	**0.2000**	0.0000	0.0000	0.0000	0.0000
0.1500	0.0000	**0.3000**	0.0000	0.0000	0.0000
0.2000	0.0000	0.0000	**0.3000**	0.0000	0.0000
0.2500	0.0000	0.0000	0.0000	**0.1000**	0.0000
0.3000	0.0000	0.0000	0.0000	0.0000	**0.1000**

ln(1+p(i,j)/p(i)q(j)

3.7612	0.0000	0.0000	0.0000	0.0000
0.0000	**3.0910**	0.0000	0.0000	0.0000
0.0000	0.0000	**2.4423**	0.0000	0.0000
0.0000	0.0000	0.0000	**1.1314**	0.0000
0.0000	0.0000	0.0000	0.0000	**0.8755**

0.0179	0.0000	0.0000	0.0000	0.0000
0.0000	**0.0442**	0.0000	0.0000	0.0000
0.0000	0.0000	**0.0698**	0.0000	0.0000
0.0000	0.0000	0.0000	**0.0539**	0.0000
0.0000	0.0000	0.0000	0.0000	**0.0625**

	sum=>	0.2483	s/r-ratio	1
pseudo	trace=>	0.2483	k/n-ratio	6
	count=>	5		
	distance	**0.225472731**	is good for a diagonal matrix	
		0.002478752	is even better	

P(i)P(j)

	0.0476	**0.0952**	**0.1429**	**0.1905**	**0.2381**	**0.2857**
0.0476	0.0023	0.0045	0.0068	0.0091	0.0113	0.0136
0.0952	0.0045	0.0091	0.0136	0.0181	0.0227	0.0272
0.1429	0.0068	0.0136	0.0204	0.0272	0.0340	0.0408
0.1905	0.0091	0.0181	0.0272	0.0363	0.0454	0.0544
0.2381	0.0113	0.0227	0.0340	0.0454	0.0567	0.0680
0.2857	0.0136	0.0272	0.0408	0.0544	0.0680	0.0816

Simulated P(i,j)

	0.0476	**0.0952**	**0.1429**	**0.1905**	**0.2381**	**0.2857**
0.0476	**0.0476**					
0.0952		**0.0952**				
0.1429			**0.1429**			
0.1905				**0.1905**		
0.2381					**0.2381**	
0.2857						**0.2857**

3.0910	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	**2.4423**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**2.0794**	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	**1.8326**	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	**1.6487**	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	**1.5041**

0.1472	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	**0.2326**	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	**0.2971**	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	**0.3491**	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	**0.3925**	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	**0.4297**

old distance= **0.1575**
sum 1.8482
trace 1.8482
count 6
distance

s/r-ratio 1.0000
k/n-ratio 5.0000
distance= **0.0001** diagonal e.g small
0.0067
0.0067

Simulated q(i)q(i)

	0.1000	***0.1500***	***0.2000***	***0.2500***	***0.3000***
0.1000	0.0100	0.0150	0.0200	0.0250	0.0300
0.1500	0.0150	0.0225	0.0300	0.0375	0.0450
0.2000	0.0200	0.0300	0.0400	0.0500	0.0600
0.2500	0.0250	0.0375	0.0500	0.0625	0.0750
0.3000	0.0300	0.0450	0.0600	0.0750	0.0900

Simulated P(i,j)

	0.1000	***0.1500***	***0.2000***	***0.2500***	***0.3000***
0.1000	**0.1000**				
0.1500		**0.1500**			
0.2000			**0.2000**		
0.2500				**0.2500**	
0.3000					**0.3000**

2.3979	0.0000	0.0000	0.0000	0.0000
0.0000	**2.0369**	0.0000	0.0000	0.0000
0.0000	0.0000	**1.7918**	0.0000	0.0000
0.0000	0.0000	0.0000	**1.6094**	0.0000
0.0000	0.0000	0.0000	0.0000	**1.4663**

0.2398	0.0000	0.0000	0.0000	0.0000
0.0000	**0.3055**	0.0000	0.0000	0.0000
0.0000	0.0000	**0.3584**	0.0000	0.0000
0.0000	0.0000	0.0000	**0.4024**	0.0000
0.0000	0.0000	0.0000	0.0000	**0.4399**

old distance= **0.1745**

sum	1.7459		s/r-ratio	1.0000
trace	1.7459		k/n-ratio	5.0000
count	5.0000			
distance	**0.0002**	diagonal	hence should be small	
	0.0067			
	0.0067			

Theoretical **P(i)Q(j)**

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***	sums
0.1	0.0048	0.0095	0.0143	0.0190	0.0238	0.0286	***0.1***
0.15	0.0071	0.0143	0.0214	0.0286	0.0357	0.0429	***0.15***
0.2	0.0095	0.0190	0.0286	0.0381	0.0476	0.0571	***0.2***
0.25	0.0119	0.0238	0.0357	0.0476	0.0595	0.0714	***0.25***
0.3	0.0143	0.0286	0.0429	0.0571	0.0714	0.0857	***0.3***
sums	***0.0476***	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***	

Simulation of P vs Q **P(i,j)** (random)
Simply perturb the P(i)Q(j) above

	0.0476	***0.0952***	***0.1429***	***0.1905***	***0.2381***	***0.2857***
0.1	0.0048	0.0095	0.0143	0.0190	0.0238	0.0286
0.15	0.0071	0.0143	0.0214	0.0286	0.0357	0.0429
0.2	0.0095	0.0190	0.0286	0.0381	0.0476	0.0571
0.25	0.0119	0.0238	0.0357	0.0476	0.0595	0.0714
0.3	0.0143	0.0286	0.0429	0.0571	0.0714	0.0857
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.6931	0.6931	0.6931	0.6931	0.6931	0.6931
	0.0033	0.0066	0.0099	0.0132	0.0165	0.0198
	0.0050	0.0099	0.0149	0.0198	0.0248	0.0297
	0.0066	0.0132	0.0198	0.0264	0.0330	0.0396
	0.0083	0.0165	0.0248	0.0330	0.0413	0.0495
	0.0099	0.0198	0.0297	0.0396	0.0495	0.0594

sum==> 0.6931 s/r-ratio 3.5000
pseudo trace=> 0.1980 k/n-ratio 1
count==> 30
Distance==> **0.82033536** <------ should be high
0.75147729
0.75147729