

BIMM

Microarrays

November 5th, 2009

Bioconductor.

Notes on using Bioconductor are found at the main Bioconductor website: www.bioconductor.org

Bioconductor is a set of add on packages for analysing gene expression data, c-DNA and oligo-arrays are covered as well as visualization of the expressed genes location along the chromosome.

One of the favorite types of data representation for microarray data is the clustered heatmap, many EDA tools are essential to microarrays.

<http://www.bioconductor.org/workshops/Milan/Lectures/GenBio.pdf>

<http://www.bioconductor.org/workshops/Milan/Lectures/OligoArrays.pdf>

c-DNA arrays

<http://www.bio.davidson.edu/courses/genomics/chip/chip.html>

<http://www.bioconductor.org/workshops/2002/WyethCourse101702/Marra>

Design of Gene expression studies

What's important inside the cell is what genes are turned on and what genes are turned off will determine the cell's composition, the difference between liver and brain; healthy and ill; immunological reactions to attack or threat which can be characterized by what genes are on and off.

We will show an example of the study of different patterns in gene expression in T-cells of different types. We can discover through the statistical analysis of the gene expression patterns, not only which genes are have different expression patterns in the different cell types, but also how genes group together according to these patterns. The hope is that we can see their organisation into genetic networks or pathways.

Many microarray experiments are designed to compare two conditions,

such as a tumorous cell and a healthy cell. There might be 6,000-10,000 probes, the actual sequences which we will be studying on the array. The array is a matrix in a sense, and the probes will be formed of subsets of the actual gene, enough to be characteristic of the genes we are interested in.

c-DNA (complementary DNA) microarrays instead of being clean dots, they're messy and not round. They rely on hybridization to stick together to make a sequence with complementary base pairing. To split up the sequences, you heat it up which is called denaturation. The red always goes with the white one and the blue always goes with the green one. Complementary DNA arrays which originated in Pat Brown's lab at Stanford. There are ink jet arrays. You might have glass, nylon membranes, paper and various ways to print or making of the probes. There's a more precise method of printing that involves radioactivity. If we could, we'd like to measure the proteins. What we're measuring

instead is the level of transcription of the mRNA. This transcripton reflects what type of cell, organ, stages of growth, healthy or ill state. Every state will have a different expression profile.

This gives a much higher resolution than just looking at the presence or absence of a mutation.

At the first level of approximation is the ratio of red and green. When you're trying to compare the two conditions, people have been using the ratio of red to green dye. We have to change that a little to get rid of the noise because there's an additive factor, there are dye effects. If you look at just the ratios, the log of the ratios M . The sum of the logs is called A . It ws decided that the effect was multiplicative and one should look at the log of the ratios, but that has changed more recently, and in general the rations tell you that if the log is smaller than zero that tells

you the ration is smaller than 1.

The idea is to look for the different wavelengths. The print tips collect cDNAs from wells in a grid. When there are streaks across the background, so you can use various methods for comparing the actual spot and deciding where the spot is called a segmentation problem. Software called Spot, genepix, ScanAnalyze and QuantArray are specialized in this image analysis and allow for various choices of segmentation separated the actual spot from its surrounding background.

It's not very precise to have a fixed circle segmentation if the spots are not very well printed.

Using the local background, one gets a typical plot of microarray intensities that can account for spatial effects.

If you just made the log of the red against the log of the green you get something along the 45 degree line. You don't have so much resolution.

$$\text{Average} = (\log_2 R + \log_2 G) / 2$$

In the usual cDNA arrays when performing spot normalization the actual grid matters. We need to take into account the spot's coordinates, as they correspond to sets of identical printtips and soemtimes there is quite a strong printtip effect. There's also systematic offset from the red and the green known as the dye effect. This can actaually only be carefully accounted for by performing dye swap experiments.

Making an inventory of all the possible sources of systematic variation are very hard without calibration experiments. There's often systematic offset from the red and the green known as the dye effect. This can actually only be carefully accounted for by performing dye swap

experiments.

The effect is due to different labeling efficiencies of the dyes. The dyes don't react in the same way. The different amounts of Cy3 (green) and Cy5 (red).

At molecular level counts, one isn't able to put the same amount of those two. There's no way to balance with exactly the right number of cells in each.

In the hybridization if you see a difference in hybridization in the red versus the green, you hope to be able to assign it to the difference in the conditions of the red and the green, but in the test tubes they may not have been the same amount of each solution of mRNA.

The different print tips have different amounts of probe that they have

put in and they might be used up or broken and there's a spatial effect and cross-hybridizations .

The variation you're after is the difference between the two conditions or the difference between the various experiments. The variations have to be rid of by normalization. We only want to see differences in expression. Different designs or different ways of doing these experiments.

The first thing one wants to do is create diagnostic plots. If you don't renormalize before there'll be a print tip effect. If the probes were all chosen at random across the whole image, you should just have random spots of intensity green and red.

With spatial effects one wants to say the baseline is much stronger and that's where the background come in. We'll need to take out the backgrounds. Boxplot per print tip The median for the different print

tips is slightly different. One wants to renormalize per print tip group. You replace the actual log ratio with the corrected log ratio by corrected log ratio which you corrected the different sectors you're in, the intensity level of your background and the channel dye. When it's global normalization, it hasn't taken out any of the local effects.

Re-normalized- taking out the correction factor L . There's sector and spatial effects usually give the same result. The boxplots are a good way of comparing whether the normalization is working well. You want everything on about the same baseline. The scale normalization is once you've taken out the local normalization you have to divide by something, the x scale. You have to use something robust: high breakdown point. The breakdown point measures robustness. We need something with a high breakdown. N numbers has breakdown point of $\frac{1}{n}$. If you want to move the mean by an arbitrary amount, take one of

the observations and move it by n times that amount. The median has a much higher breakdown. You need to use absolute deviation.

MAD rescaling.

Lowess smoothing for renormalisation

Variance Stabilization and Normalisation

There is another way to rescale is by wolfgang huber (SP) who had a very good idea on how to standardize the scales. Heteroscedascity: a phenomena in which the variance depends on the intensity, or size of the expressin. In any kind of statistical situation you want the variance to be the same across the range of the meausrement of the random variable.

If this is not true, we will only detect differences in expression for genes that are highly expressed.

Web references:

<http://www.bio.davidson.edu/courses/genomics/chip/chip.html>

<http://www.imagecyte.com/array2.html>

<http://www.bioconductor.org/workshops/Milan/Lectures/MachineLearning>

<http://www.bioconductor.org/workshops/Seattle02/MarrayTech.pdf>

Multidimensional Scaling

From a non-technical point of view, the purpose of multidimensional scaling (MDS) is to provide a visual representation of the pattern of proximities (i.e., similarities or distances) among a set of objects. For example, given a matrix of perceived similarities between various brands of air fresheners, MDS plots the brands on a map such that those brands that are perceived to be very similar to each other are placed near each other on the map, and those brands that are perceived to be very different from each other are placed far away from each other on the map.

For instance, given the matrix of distances among cities shown above, MDS produces a new map. In this example, the relationship between

input proximities and distances among points on the map is positive: the smaller the input proximity, the closer (smaller) the distance between points, and vice versa. Had the input data been similarities, the relationship would have been negative: the smaller the input similarity between items, the farther apart in the picture they would be.

From a slightly more technical point of view, what MDS does is find a set of vectors in p -dimensional space such that the matrix of euclidean distances among them corresponds as closely as possible to some function of the input matrix according to a criterion function called **STRESS**. A simplified view of the algorithm is as follows:

1. Assign points to arbitrary coordinates in p -dimensional space.
2. Compute euclidean distances among all pairs of points, to form the

Dhat matrix.

3. Compare the Dhat matrix with the input D matrix by evaluating the stress function. The smaller the value, the greater the correspondance between the two.
4. Adjust coordinates of each point in the direction that best maximally stress.
5. Repeat steps 2 through 4 until stress won't get any lower.

Input Data

The input to MDS is a square, symmetric 1-mode matrix indicating relationships among a set of items. By convention, such matrices are categorized as either similarities or dissimilarities, which are opposite poles of the same continuum. A matrix is a similarity matrix if larger numbers indicate *more* similarity between items, rather than less. A matrix is a dissimilarity matrix if larger numbers indicate less similarity. The distinction is somewhat misleading, however, because similarity is not the only relationship among items that can be measured and analyzed using MDS. Hence, many input matrices are neither similarities nor dissimilarities. However, the distinction is still used as a means of indicating whether larger numbers on the map, or far apart. Calling the

data 'similarities' indicates a negative or descending relationship between input values and corresponding map distances, while calling the data "dissimilarities" or "distances" indicates a positive or ascending relationship.

A typical example of an input matrix is the aggregate proximity matrix derived from a pilesort task. Each cell x_{ij} of such a matrix records the number (or proportion) of respondents who placed items i and j into the same pile.

It is assumed that the number of respondents placing two items into the same pile is an indicator of the degree to which they are similar. An MDS map of such data would put items close together which were often sorted into the same piles.

Another typical example of an input matrix is a matrix of correlations

among variables.

Treating these data as similarities (as one normally would), would cause the MDS program to put variables with high positive correlations near each other, and variables with strong negative correlations far apart.

Another type of input matrix is a flow matrix. For example, a dataset might consist of the number of business transactions occurring during a given period between a set of corporations. Running this data through MDS might reveal clusters of corporations that whose members trade more heavily with one another than other than with outsiders. Although technically neither similarities nor dissimilarities, these data should be classified as similarities in order to have companies who trade heavily with each other show up close to each other on the map.

Dimensionality

Normally, MDS is used to provide a visual representation of a complex set of relationships that can be scanned at a glance. Since maps on paper are two-dimensional objects, this translates technically to finding an optimal configuration of points in 2-dimensional space. However, the best possible configuration in two dimensions may be a very poor, highly distorted, representation of your data. If so, this will be reflected in a high stress value. When this happens, you have two choices: you can either abandon MDS as a method of representing your data, or you can increase the number of dimensions. There are two difficulties with increasing the number of dimensions. The first is that even 3 dimensions are difficult to display on paper and are significantly more difficult to comprehend. Four or more dimensions render MDS virtually useless as a method of making complex data more accessible to the human mind. (However, there are other uses of MDS that are not affected by this

problem.) The second problem is that with increasing dimensions, you must estimate an increasing number of parameters to obtain a decreasing improvement in stress. The result is model of the data that is nearly as complex as the data itself.

On the other hand, there are some applications of MDS for which high dimensionality is not a problem. For instance, MDS can be viewed as a mathematical operation that converts an item-by-item matrix into an item-by-variable matrix. Suppose, for example, that you have a person-by-person matrix of similarities in attitudes. You would like to explain the pattern of similarities in terms of simple personal characteristics such as age, sex, income and education. The trouble is, these two kinds of data are not conformable. The person-by-person matrix in particular is not the sort of data you can use in a regression to predict age (or vice-versa). However, if you run the data through MDS

(using very high dimensionality in order to achieve perfect stress), you can create a person-by-dimension matrix which is similar to the person-by-demographics matrix that you are trying to compare it to.

Stress

The degree of correspondence between the distances among points implied by MDS map and the matrix input by the user is measured (inversely) by a *stress* function. The general form of these functions is as follows:

$$\text{STRESS} = \sqrt{\frac{\sum (f(x_{ij}) - d_{ij})^2}{\text{scale}}}$$

In the equation, d_{ij} refers to the euclidean distance, across all dimensions, between points i and j on the map, $f(x_{ij})$ is some function of the input data, and *scale* refers to a constant scaling factor, used to keep stress values between 0 and 1. When the MDS map perfectly reproduces the input data, the stress is zero. Thus, the smaller the stress, the better the representation.

The stress function used in ANTHROPAC is variously called 'Kruskal Stress'.

The transformation of the input values used depends on whether metric or non-metric scaling. In metric scaling, $f(x_{ij}) = x_{ij}$. In other words, the raw input data is compared directly to the map distances (at least in the case of dissimilarities: see the section of metric scaling for information on similarities). In non-metric scaling, $f(x_{ij})$ is a weakly monotonic transformation of the input data that minimizes the stress function. The monotonic transformation is computed via 'monotonic regression', also known as 'isotonic regression'.

From a mathematical standpoint, non-zero stress values occur for only one reason: insufficient dimensionality. That is, for any given dataset, it may be impossible to perfectly represent the input data in two or other

small number of dimensions. On the other hand, any dataset can be perfectly represented using $n-1$ dimensions, where n is the number of items scaled. As the number of dimensions used goes up, the stress must either come down or stay the same. It can never go up.

Of course, it is not necessary that an MDS map have zero stress in order to be useful.

A certain amount of distortion is tolerable. Different people have different standards regarding the amount of stress to tolerate. The rule of thumb we use is that anything under 0.1 is excellent and anything over 0.15 is unacceptable. Care must be exercised in interpreting any map that has non-zero stress since, by definition, non-zero stress means that some or all of the distances in the map are, to some degree, distortions of the input data. The distortions may be spread out over all

pairwise relationships, or concentrated in just a few egregious pairs. In general, however, longer distances tend to be more accurate than shorter distances, so larger patterns are still visible even when stress is high. See the section on Shepard Diagrams and Interpretation for further information on this issue.

From a substantive standpoint, stress may be caused either by insufficient dimensionality, or by random measurement error. For example, a dataset consisting of distances between buildings in New York City, measured from the center of the roof, is clearly 3-dimensional. Hence we expect a 3-dimensional MDS configuration to have zero stress. In practice, however, there is measurement error such that a 3-dimensional solution does not have zero stress. In fact, it may be necessary to use 8 or 9 dimensions to bring stress down to zero. In this case, the fact that the 'true' number of dimensions is known to be three

allows us to use the stress of the 3-dimensional solution as a direct measure of measurement error. Unfortunately, in most datasets, it is not known in advance how many dimensions there 'really' are.

In such cases we hope (with little foundation) that the true dimensionality of the data will be revealed to us by the rate of decline of stress as dimensionality increases. For example, in the distances between buildings example, we would expect significant reductions in stress as we move from a one to two to three dimensions, but then we expect

the rate of change to slow as we continue to four, five and higher dimensions. This is because we believe that all further variation in the data beyond that accounted for by three dimensions is non-systematic noise which must be captured by a host of 'specialized' dimensions each accounting for a tiny reduction in stress. Thus, if we plot stress by

dimension, we expect the following sort of curve:

Thus, we can theoretically use the 'elbow' in the curve as a guide to the dimensionality of the data. In practice, however, such elbows are rarely obvious, and other, theoretical, criteria must be used to determine dimensionality.

Shepard Diagrams

The Shepard diagram is a scatterplot of input proximities (both x_{ij} and $f(x_{ij})$) against output distances for every pair of items scaled.

Normally, the X-axis corresponds to the input proximities and the Y-axis corresponds to both the MDS distances d_{ij} and the transformed ('fitted') input proximities $f(x_{ij})$. An example is given in Figure 3. In the plot, asterisks mark values of d_{ij} and dashes mark values of $f(x_{ij})$. Stress measures the vertical discrepancy between x_{ij} (the map distances) and $f(x_{ij})$ (the transformed data points). When the stress is zero, the asterisks and dashes lie on top of each other. In metric scaling, the asterisks form a straight line. In nonmetric scaling, the asterisks form a weakly monotonic function, the shape of which can sometimes be revealing (e.g., when map-distances are an exponential function of input proximities).

If the input proximities are similarities, the points should form a loose line from top left to bottom right, as shown. If the proximities are dissimilarities, then the data should form a line from bottom left to top right. In the case of non-metric scaling, $f(x_{ij})$ is also plotted.

Interpretation

There are two important things to realize about an MDS map. The first is that the axes are, in themselves, meaningless and the second is that the orientation of the picture is arbitrary. Thus an MDS representation of distances between US cities need not be oriented such that north is up and east is right. In fact, north might be diagonally down to the left and east diagonally up to the left. All that matters in an MDS map is which point is close to which others.

When looking at a map that has non-zero stress, you must keep in mind that the distances among items are imperfect, distorted, representations of the relationships given by your data. The greater the stress, the greater the distortion. In general, however, you can rely on the larger

distances as being accurate. This is because the stress function accentuates discrepancies in the larger distances, and the MDS program therefore tries harder to get these right.

There are two things to look for in interpreting an MDS picture: clusters and dimensions. Clusters are groups of items that are closer to each other than to other items. For example, in an MDS map of perceived similarities among animals, it is typical to find (among north americans) that the barnyard animals such as chicken, cow, horse, and pig are all very near each other, forming a cluster. Similarly, the zoo animals like lion, tiger, antelope, monkey, elephant and giraffe form a cluster. When really tight, highly separated clusters occur in perceptual data, it may suggest that each cluster is a domain or subdomain which should be analyzed individually. It is especially important to realize that any relationships observed within such a cluster, such as item *a* being slightly

closer to item b than to c should not be trusted because the exact placement of items within a tight cluster has little effect on overall stress (In some cases, however, you will want to re-run the data collection instead.)

Dimensions are item attributes that seem to order the items in the map along a continuum. For example, an MDS of perceived similarities among breeds of dogs may show a distinct ordering of dogs by size. The ordering might go from right to left, top to bottom, or move diagonally at any angle across the map. At the same time, an independent ordering of dogs according to viciousness might be observed. This ordering might be perpendicular to the size dimension, or it might cut a sharper angle.

The underlying dimensions are thought to 'explain' the perceived similarity between items. For example, in the case of similarities among

dogs we expect that the reason why two dogs are seen as similar is because they have locations or scores on the identified dimensions. Hence, the observed similarity between a doberman and a german shepherd is explained by the fact that they are seen as nearly equally vicious and about the same size. Thus, the implicit model of how similarity judgments are produced by the brain is that items have attributes (such as size, viciousness, intelligence, furriness, etc) in varying degrees, and the similarity between items is a function of their similarity in scores across all attributes. This function is often conceived of as a weighted sum of the similarity across each attribute, where the weights reflect the importance or saliency of the attribute.

It is important to realize that these substantive dimensions or attributes need not correspond in number or direction to the mathematical dimensions (axes) that define the vector space (MDS map). For

example, the number of dimensions used by respondents to generate similarities may be much larger than the number of mathematical dimensions needed to reproduce the observed pattern. This is because the mathematical dimensions are necessarily orthogonal (perpendicular), and therefore maximally efficient. In contrast, the human dimensions, while cognitively distinct, may be highly intercorrelated and therefore contain some redundant information.

One thing to keep in mind in looking for dimensions is that your respondents may not have the same views that you do. For one thing, they may be reacting to attributes you have not thought of. For another, even when you are both using the same set of attributes, they may assign different scores on each attribute than you do. For example, one of the attributes might be 'attractiveness'. Your view of what an attractive dog, person, fruit or other item may be very different from

your respondents'. If the input data dissimilarities, the function is never decreasing. If the input data are similarities, the function is never increasing.

In some cases, however, it is better to rerun the data collection on the subset of items. This is because the presence of the other items can evoke additional dimensions/attributes of comparison that could affect the way items in the subset are viewed.

Programs:

By far the best programs for interactively exploring multidimensional data are the Xgobi/xgvis/ggobi suite of programs, these are free software that can be installed on most computers, they allow for browsing through many dimensions at a time.

Here are some of the sites for downloads:

<http://www.ggobi.org/Download.html>

<http://www.research.att.com/areas/stat/xgobi/>

Web References:

<http://www.analytictech.com/networks/mds.htm>

<http://www.research.att.com/areas/stat/xgobi/>

<http://www.ggobi.org/>

http://industry.ebi.ac.uk/~alan/VisWorkshop99/XGobi_Tutorial/

http://industry.ebi.ac.uk/~alan/VisWorkshop99/XGobi_Talk/sld026.htm

<http://www.biol.ttu.edu/Faculty/FacPages/Strauss/Matlab/matlab.htm>

<http://www.galaxy.gmu.edu/~dcarr/lib/v8n1.pdf>

Example of using MDS on gene expression data:

<http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM11>

http://www.nhgri.nih.gov/DIR/Microarray/Melanoma_Supplement/index.h

Principal Components and Singular Value Decomposition (SVD)

This is a linear dimension-reducing technique for continuous variables that all have the same status.

Description of singular value Decomposition This is the most important

matrix decomposition in statistics.

```
> X=u*v'  
> u=c(3, 1, -1, 2)'  
3 3 6 9 12  
1 1 2 3 4  
-1 -1 -2 -3 -4  
2 2 4 6 8  
> v=(1:4)/sd(X)$sd  
1 21.2132  
2 0.0000  
3 0  
4 0  
> E=10^(-3)*matrix(runif(16
```

```
> XE=X+E
```

```
XE =
```

```
3.0012    5.9993    9.0003    12.0012
1.0006    2.0017    3.0009    3.9994
-0.9999   -1.9999   -3.0014   -3.9994
2.0004    4.0018    5.9993    7.9996
```

```
> svd(XE)$d
```

```
21.2143
0.0028
0.0019
0.0005
```

%An example you can't see with your bare eyes:

```
> X2=u'*v
```

```
X2 =
```

```
0.2468    0.2499    0.0166    0.2487
0.3225    0.3266    0.0218    0.3250
```

```
0.1495    0.1514    0.0101    0.1506
0.4367    0.4423    0.0295    0.4401
> svd(X2)$d
1.0730
0.0000
0
0
> USV=svd(X2)
> U=USV$u
> S=diag(USV$d)
> V=USV$v
> U%*%S%*%t(V)
0.2468    0.2499    0.0166    0.2487
0.3225    0.3266    0.0218    0.3250
```

```
0.1495    0.1514    0.0101    0.1506
0.4367    0.4423    0.0295    0.4401
```

```
> 10000*(X2-U%*%S%*%t(V))
```

```
1.0e-11 *
-0.0278    0.0833   -0.0035    0.0555
-0.0555    0.0555         0    0.0555
         0    0.0555    0.0017    0.0278
-0.0555    0.1665         0    0.1110
```

```
> A=rand(4)
```

```
A =
0.5045    0.4940    0.0737    0.9138
0.5163    0.2661    0.5007    0.5297
0.3190    0.0907    0.3841    0.4644
0.9866    0.9478    0.2771    0.9410
```

```
> flops(0)
```

```
> lup=lu(A)
```

```
L =
```

1.0000	0	0	0
0.5233	1.0000	0	0
0.5114	-0.0406	1.0000	0
0.3234	0.9388	0.7363	1.0000

```
U =
```

0.9866	0.9478	0.2771	0.9410
0	-0.2298	0.3557	0.0373
0	0	-0.0535	0.4342
0	0	0	-0.1945

```
> P*A
```

0.9866	0.9478	0.2771	0.9410
--------	--------	--------	--------

```
0.5163    0.2661    0.5007    0.5297
0.5045    0.4940    0.0737    0.9138
0.3190    0.0907    0.3841    0.4644
```

```
> L*U
```

```
0.9866    0.9478    0.2771    0.9410
0.5163    0.2661    0.5007    0.5297
0.5045    0.4940    0.0737    0.9138
0.3190    0.0907    0.3841    0.4644
```

```
> P*A-L*U
```

```
ans =      1.0e-15 *
          0          0          0          0
-0.1110          0          0          0
          0          0          0          0
-0.0555   -0.0139          0          0
```

Here is what we need to remember:

$$X = USV', V'V = I, U'U = I, S \text{ diagonal } s_i$$

Actually the singular values are the square roots of the eigenvalues of $X'X$. Principal Components **Detecting Linear Dependence**

The first objective of a principal component analysis as often is a reduction of the dimensionality of the problem, of course if, for instance we start out with 12 independent (thus completely unrelated) variables there is no hope of reducing the dimensionality of the problem, and there is no multi-dimensionnal phenomenon. Then all the information will be contained in the sequence of 12 one-dimensionnal analyses.

Careful : I did not say that it is sufficient in the general case to look and

verify that all the correlation coefficients were zero, because if the data are not normal, there could still be a zero-correlation and non independence. For instance there could be higher order dependencies (quadratic, ..).

- Start by recentring X , from now on consider X centered ie $\mathbf{1}_n X = 0$,
- Cols of $V \rightarrow$ are new variables,
- Principal Components $C = US \quad C'C = S^2 \quad C = XV$,
- Principal axes $Z = U'X = SV'$,

- Distance between two points,

$$(\mathbf{x}_{k.} - \mathbf{x}_{l.})' (\mathbf{x}_{k.} - \mathbf{x}_{l.}) = \sum_{j=1}^T (c_{kj} - c_{lj})^2,$$

- Transition Formulae $Z = S^{-1}C'X$ $C = XZ'S^{-1}$,

X centered, all points (observations) same weight.

$$\mathbf{1}' X = 0 \quad \text{and} \quad x_{ij} = \sum_{t=1}^r x_{it} s_t v_{jt},$$

p variables can be replaced by the r columns of v .

$$x_{ij}^{(k)} = \sum_{t=1}^k u_{it} s_t v_{jt}$$

is the best (optimal) approximation of rank k for X . The columns of V are the directions along which the variances are maximal. Principal components are the coordinates of the observations on the basis of the new variables (namely the columns of V) and they are the rows of $G = XV = US$. The components are orthogonal and their lengths are the singular values $C'C = SU'US = S^2$. In the same way the principal axes are defined as the rows of the matrix $Z = U'X = SV'$. Coordinates of the variables on the basis of columns of U .

$$Z = S^{-1}C'X \quad \text{and} \quad G = XZ'S^{-1}.$$

However, this decomposition will be highly dependent upon the unity of measurement (scale) on which the variables are measured. It is only used, in fact, when the $X_{.k}$ are all of the same “order”. Usually what is done is that a weight is assigned to each variable that takes into account its overall scale. This weight is very often inversely proportional to the variable’s variance.

So we define a different metric between observations instead of

$$d(x_{i.}, x_{j.}) = (x_{i.} - x_{j.})', (x_{i.} - x_{j.})$$

$$d(x_{i.}, x_{j.}) = (x_{i.} - x_{j.}), Q(x_{i.} - x_{j.})$$

$$Q = \begin{pmatrix} \frac{1}{\sigma_1^2} & & \\ & \dots & \\ & & \frac{1}{\sigma_p^2} \end{pmatrix}.$$

The same can be said of the observations; some may be more “important” than others (resulting from a mean of a group which is larger). Example for analysing the Scores Example The scores data are a first example for understanding principal components. Mardia, Kent and Bibby’s data on 88 students: **Score data**

c	c	o	o	o
mec	vec	alg	ana	stat
77	82	67	67	81
63	78	80	70	81
75	73	71	66	81
55	72	63	70	68
63	63	65	70	63
53	61	72	64	73
51	67	65	65	68

59	70	68	62	56
62	60	58	62	70
64	72	60	62	45
52	64	60	63	54
55	67	59	62	44
50	50	64	55	63
65	63	58	56	37
31	55	60	57	73
60	64	56	54	40
44	69	53	53	53
42	69	61	55	45
.....				
30	24	43	33	25
3	9	51	47	40

```
7  51  43  17  22
15 40  43  23  18
15 38  39  28  17
5  30  44  36  18
12 30  32  35  21
5  26  15  20  20
0  40  21  9  14
```

```
> round(cov(scor88))
```

```
  306   127   102   106   117
  127   173    85    95    99
  102    85   113   112   122
  106    95   112   220   156
  117    99   122   156   298
```

```
> round((scorc88')*(scorc88)/87)
```

```
306    127    102    106    117
127    173     85     95     99
102     85    113    112    122
106     95    112    220    156
117     99    122    156    298
```

```
> corrccoef(scor88)
```

```
 1.0000    0.5534    0.5468    0.4094    0.3891
 0.5534    1.0000    0.6096    0.4851    0.4364
 0.5468    0.6096    1.0000    0.7108    0.6647
 0.4094    0.4851    0.7108    1.0000    0.6072
 0.3891    0.4364    0.6647    0.6072    1.0000
```

```
> mean(scor88)
```

```
ans = 38.95    50.59    50.60    46.68    42.3068
```

```
> scorc88=scor88-rep(1,88)%*%apply(scor88,2,mean)
```

```
> apply(scorc88,2,mean)
```

```
1.0e-13 *
```

```
0.1421 -0.1970 -0.0565 -0.0436 -0.0363
```

```
> USV= svd(scorc88)
```

```
> S=USV$d
```

```
> V=USV$v
```

```
> diag(S)
```

```
244.4752 0 0 0 0
```

```
0 132.6034 0 0 0
```

```
0 0 95.0053 0 0
```

```
0 0 0 85.8070 0
```

```
0 0 0 0 52.8898
```

```
> S^2/87
```

```
ans =
```

```
686.9898      0      0      0      0
      0 202.1111      0      0      0
      0      0 103.7473      0      0
      0      0      0 84.6304      0
      0      0      0      0 32.1533
```

```
> V
```

```
V = 0.5054 -0.7487 -0.2998 0.2962 -0.0794
     0.3683 -0.2074 0.4156 -0.7829 -0.1889
     0.3457 0.0759 0.1453 -0.0032 0.9239
     0.4511 0.3009 0.5966 0.5181 -0.2855
     0.5347 0.5478 -0.6003 -0.1757 -0.1512
```

```
> VE=eig(cov(scorc88))$v
```

```
VE =
     0.2962 -0.2998 0.0794 -0.7487 0.5054
```

```
-0.7829    0.4156    0.1889   -0.2074    0.3683
-0.0032    0.1453   -0.9239    0.0759    0.3457
 0.5181    0.5966    0.2855    0.3009    0.4511
-0.1757   -0.6003    0.1512    0.5478    0.5347
```

LE =

```
84.6304         0         0         0         0
      0  103.7473         0         0         0
      0         0  32.1533         0         0
      0         0         0  202.1111         0
      0         0         0         0  686.9898
```

evscores=diag(S^2/87)

cumsum(evscores/sum(evscores))'

```
0.6191    0.8013    0.8948    0.9710    1.0000
```

ap=S%*%V

```
plot(ap(:,1),ap(:,2),'.')
      Axe      1      Axe      2
      coord. cta ctr coord. cta ctr
V1  13.20 255 574 -10.60 561 371
V2   9.60 136 539  -2.93  43  50
V3   9.01 119 727   1.07   6  10
V4  11.80 204 634   4.25  91  83
V5  13.90 286 660   7.74 300 204
```

```
function out=pca(data,n,p,k){
#Do a principal component analysis
#Return the ratio of the variance
#explained by the first k components
```

```
#
data=data-rep(1,n)*apply(data,2,mean)
USV = svd(data)
S=USV$d
S2=diag(S).^2
out=sum(S2(1:k))/sum(S2(1:p))
renorm=scor88*(diag(vars)^(-0.5))
```

Nonparametric Bootstrap of the ratio of the variance of the first few(k) axes.

```
function out=bpca(B,orig,k)
#Bootstrap of the PCA of a dataset
#Output the proportion
#of variance for the first k axes
```

```
[n,p]=size(orig)
for (b =(1:B))
    newsample=bsample(orig)
    out(b)=pca(newsample,n,p,k)
end %-----
```

```
function out=bsample(orig)
#Function to create one resample from
#the original sample orig, where
#orig is the original data, and is a matrix
    [n,p]=size(orig)
    indices=randint(1,n,n)+1
    out=orig(indices,:)
```

```
#-----
```

```
function out=pca(data,n,p,k)
```

```
#Do a principal component analysis
```

```
#Return the ratio of the variance
```

```
#explained by the first k components
```

```
data=data-ones(n,1)*mean(data)
```

```
[U,S,V] = svd(data,0)
```

```
S2=diag(S).^2
```

```
out=sum(S2(1:k))/sum(S2(1:p))
```

```
#-----
```

```
> resbpca10K1=bpca(10000,scor88,1)
```

```
> resbpca10K=bpca(10000,scor88,2)
> hist(resbpca10K,100)
```

one component

How to generate a multivariate normal Choleski decomposition of a matrix is the decomposition of A of the form

$$A = L'L$$

```
> chl=chol(cov(scor88))
> round(chl'*chl)
  306   127   102   106   117
  127   173    85    95    99
  102    85   113   112   122
```

106	95	112	220	156
117	99	122	156	298

```
> xs=randn(88,5)
```

```
> mean(xs)
```

0.0480	-0.2056	0.1209	0.1097	0.1051
--------	---------	--------	--------	--------

```
> corrcoef(xs)
```

1.0000	-0.1623	-0.0664	-0.1947	-0.1042
-0.1623	1.0000	0.1412	0.0361	0.0600
-0.0664	0.1412	1.0000	-0.2334	0.0056
-0.1947	0.0361	-0.2334	1.0000	-0.0734
-0.1042	0.0600	0.0056	-0.0734	1.0000

```
> ys=xs*ch1
```

```
> cov(ys)
```

225.8649	65.7374	57.5737	29.1913	40.6952
----------	---------	---------	---------	---------

65.7374	149.5844	74.5579	74.1853	83.3744
57.5737	74.5579	93.5467	66.5157	89.7249
29.1913	74.1853	66.5157	152.4465	88.0262
40.6952	83.3744	89.7249	88.0262	210.1678

Parametric Bootstrap Simulations

```
> xs=randn(88,5)
```

```
> mean(xs)
```

```
ans= 0.013  0.035 -0.151 -0.059  0.033
```

```
> ys=xs*ch1
```

```
> round(cov(ys))
```

309	169	147	130	150
169	237	137	160	146
147	137	161	158	149

130 160 158 252 172

150 146 149 172 272

```
function out=spca(B,orig,k)
#Parametric Bootstrap of the PCA of a dataset
#Output the proportion
#of variance for the first k axes
out=zeros(1,B)
[n,p]=size(orig)
chl=chol(cov(orig))
for (b =(1:B))
    rands=randn(n,p)
    newsample=rands*chl
    out(b)=pca(newsample,88,5,k)
end
```

```
r100k=spca(100000, scor88, 2)
```

```
hist(r100k, 200)
```

ppca14inHistogram of first component's weight

Correspondence Analysis

Correspondence Analysis for Expression Data

There are very few books or articles in English on the subject, Michael Greenacre who studied in France under Benzecri[?] has written several very readable texts on the subject[?].

With today's explosion of genetic data in the form of contingency tables both two-way and three-way, this method is very useful in applications such as gene expression microarray data.

Correspondence analysis provides students at the masters level in statistics, applied mathematics, biology and engineering with an opportunity for performing multivariate data analysis without having to use prepackaged programs.

Aims and relevant data

We can make a dichotomy of data-mining and multivariate statistical methods into two groups, one series of methods confers a particular status to one variable or set of variables, these are to be predicted or explained, they are the response. Methods include regression, multiple response regression, discriminant analysis, analysis of variance depending on whether the explanatory variables are categorical or continuous. These are not the ones we are going to study here.

Correspondence analysis is useful when all the variables have the same status. This is sometimes called unsupervised learning, and includes clustering, principal components as well. They have in common the creation of a new set of variables that simplify the arrays at hand. In the

case of clustering, the new variable is a categorical, in correspondence analysis and principal components the new variables are continuous and enable the construction of useful new graphical representations of the data.

Correspondence analysis is an exploratory method because it does not presuppose any model for the data, as do Goodman's bilinear methods or factor analysis models for instance. Correspondence analysis and principal components can both be extended to three-way arrays, for instance for the analysis of bootstrap permutation tests or time series of matrices. Such data are often called data cubes.

Correspondence Analysis

Correspondence analysis (CA, also called homogeneity analysis and reciprocal averaging), can be used to analyse several types of multivariate data. All involve some categorical variables. Here are some examples of the type of data that can be decomposed using this method:

- Contingency Tables (cross between two categorical variables)
- Multiple Contingency Tables (cross between several categorical variables).
- Binary tables obtained by cutting continuous variables into classes and then recoding both these variables and any extra categorical variables

into 0/1 tables, 1 indicating presence in that class. So for instance a continuous variable cut into three classes will provide three new binary variables of which only one can take the value 1 for any given observation.

To first approximation, correspondence analysis can be understood as an extension of principal components analysis (PCA) where the variance in PCA is replaced by an inertia proportional to the χ^2 distance of the table from independence. CA decomposes this measure of departure from independence along axes that are orthogonal according to the χ^2 inner product. If we are comparing two categorical variables, the simplest possible model is that of independence in which case the counts in the table would obey approximately the margin products identity for a $m \times p$ contingency table with a total sample size of $n = \sum_{i=1}^m \sum_{j=1}^p n_{ij} = n \dots$

Independence means

$$n_{ij} \doteq \frac{\bar{n}_i \bar{n}_j}{\bar{n}}$$

can also be written: $N \doteq cr'n$, where

$$c = \frac{1}{n} N 1_m \quad \text{and} \quad r' = \frac{1}{n} N' 1_p$$

The departure from independence is measured by the χ^2 statistic

$$\chi^2 = \sum_{i,j} \left[\frac{(n_{ij} - \frac{\bar{n}_i \bar{n}_j}{\bar{n}})^2}{\frac{\bar{n}_i \bar{n}_j}{\bar{n}}} \right]$$

Example: Eye color -Hair color

Here is a simple contingency table as our example from Snee (1974)[?].

eyes	Black	Brunette	Red	Blonde
Brown	68	20	15	5
Blue	119	84	54	29
Hazel	26	17	14	14
Green	7	94	10	16

```
> chisq.test(eyes)
```

```
    Pearson's Chi-squared test
```

```
data:  eyes, X-squared = 138.2898,
```

```
df = 9, p-value = < 2.2e-16
```

This is a very extreme point in the χ^2 distribution. The inertia is the χ -squared statistic divided by the number of observations ($\text{sum}(\text{eyes})$); Here, the inertia is $138.3/592 = 0.2336$. CA decomposes this inertia into the sum of eigenvalues of a symmetrized reweighted version of the