BOUNDING, AND SOMETIMES FINDING, THE GLOBAL MINIMUM IN MULTIDIMENSIONAL SCALING

JAN DE LEEUW

ABSTRACT. Meet the abstract. This is the abstract.

1. PROBLEM

In Least Squares Euclidean Metric Multidimensional Scaling (LSEM-MDS) the data are non-negative dissimilarities δ_{ij} between pairs (o_i, o_j), where the o_i come from a set $\mathcal{O}$ of n objects.

LSEM-MDS constructs a mapping $\phi : \mathcal{O} \Rightarrow \mathbb{R}^p$ of the n objects into low dimensional Euclidean space. Let $z_i = \phi(o_i)$, and collect the z_i in an $n \times p$ matrix Z. The mapping ϕ is found in such a way that the Euclidean distances $d_{ij}(Z) = \|z_i - z_j\|$ between the points representing objects optimally approximate the dissimilarities δ_{ij} in the weighted least squares sense. The loss function, traditionally called *stress*, is consequently

$$\sigma(Z) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij}(\delta_{ij} - d_{ij}(Z))^2,$$

where the ω_{ij} are given non-negative weights. The LSEM-MDS problem, which for brevity we shall just call the MDS problem from now on, is to minimize σ over Z.

Date: Friday 28th February, 2014 — 9h 2min — Typeset in LUCIDA BRIGHT.

1.1. **Reparametrization.** The classical formulation of the MDS problem above has some features which are mathematically inconvenient. The matrix Z can be rotated and translated without changing the distances. This means there is a lack of identification which can lead to unnecessary complications in the analysis of the problem. Moreover, if we consider second derivatives, it is clear that mixed second-order partial derivatives for coordinates of the same point (i.e. coordinates in the same row of Z) will have different expressions than mixed second-order partial derivatives for coordinates of different points. Again this is an unnecessary complication which we can easily eliminate. Finally, it would be convenient if we can move from expressions in summation notation to expressions in matrix notation.

Most of this can be accomplished by writing Z in the form

$$Z = \sum_{\ell=1}^{m} \zeta_\ell G_\ell,$$

where the G_ℓ form a basis for a subspace of $n \times p$ matrices. As a special case this subspace can be all of $\mathbb{R}^{n\times p}$. The vector ζ now is the new variable over which we want to minimize σ.

Next we use unit vectors e_i and e_j to write

$$d_{ij}^2(Z) = (z_i - z_j)'(z_i - z_j) = (e_i - e_j)'ZZ'(e_i - e_j) = \mathbf{tr}\ Z'E_{ij}Z,$$

where $E_{ij} = (e_i - e_j)(e_i - e_j)'$. Combining this with the new parametrization gives

$$d_{ij}^2(Z) = \sum_{\nu=1}^{m}\sum_{\ell=1}^{m} \zeta_\nu\zeta_\ell\ \mathbf{tr}\ G_\nu' E_{ij} G_\ell.$$

If we define the $m\times m$ matrices V_{ij} with elements $\{V_{ij}\}_{\nu\ell} = \mathbf{tr}\ G_\nu' E_{ij} G_\ell$, then

$$d_{ij}^2(Z) = \zeta' V_{ij} \zeta.$$

The expression for stress becomes

$$\sigma(\zeta) = 1 - \sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij}\delta_{ij}\sqrt{\zeta' V_{ij}\zeta} + \frac{1}{2}\zeta'\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\omega_{ij}V_{ij}\right]\zeta,$$

where we have scaled the dissimilarties, without loss of generality, such that

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\delta_{ij}^{2} = 1.$$

Our next simplification is to get rid of double subscripts. Suppose there are K pairs (i,j) for which $\omega_{ij} > 0$. Number these pairs using a single subscript k, and we find

$$\sigma(\zeta) = 1 - \sum_{k=1}^{K} \omega_k\delta_k\sqrt{\zeta' V_k \zeta} + \frac{1}{2}\zeta'\left[\sum_{k=1}^{K}\omega_k V_k\right]\zeta.$$

Now use a change of coordinates by using the eigen-decomposition $\sum_{k=1}^{K}\omega_k V_k = L\Lambda L'$. The new variables are $x = \Lambda^{\frac{1}{2}}L'x\zeta$ and we define

$$A_k = \Lambda^{-\frac{1}{2}}L'V_k L\Lambda^{-\frac{1}{2}},$$

so that $\sum_{k=1}^{K}\omega_k A_k = I$, and

$$x'A_k x = \zeta' V_k \zeta.$$

Thus

$$\sigma(x) = 1 - \sum_{k=1}^{K}\omega_k\delta_k\sqrt{x'A_k x} + \frac{1}{2}x'x.$$

Two more steps are needed to get the problem into its final form. First we simply define $w_k = \omega_k\delta_k$. Then we use homogeneity in the form

$$\min_{x}\sigma(x) = \min_{x'x=1}\min_{\alpha} 1 - \alpha\sum_{k=1}^{K} w_k\sqrt{x'A_k x} + \frac{1}{2}\alpha^2,$$

and carrying out the inner minimization over α shows our MDS problem can equivalently be formulated as the maximization of the homogeneous convex function

$$\rho(x) = \sum_{k=1}^{K} w_k\sqrt{x'A_k x}$$

over the unit sphere $x'x = 1$, or, equivalently, over the unit ball $x'x \leq 1$. Note that $\rho(x) \leq 1$ for each $x \in S$, with equality if and only if $w = d(x)$. Also $\rho(x) > 0$ for all $x \neq 0$, i.e. ρ is a norm.

Alternatively we can also formulate the MDS problem as the maximization of the non-linear "Rayleigh quotient"

$$\lambda(x) = \frac{\rho(x)}{\|x\|},$$

which is a ratio of two norms. In **deleeuw_C_77** this ratio-of-norms formulation was used in connection with the general results of **robert_67**.

2. Derivatives

It is convenient to define the functions $d_k(x) \triangleq \sqrt{x'A_kx}$ and collect them in the vector $d(x)$. Also define

$$B(x) \triangleq \sum_{k=1}^{K} w_k s_k(x) A_k,$$

with

$$s_k(x) = \begin{cases} \frac{1}{d_k(x)} & \text{if } d_k(x) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\rho(x) = x'B(x)x$, and if $w = d(x)$ then $B(x) = I$.

Remember that ρ has a local maximum at $x \in S$ if there is an open ball $\mathcal{B}_\epsilon(x)$ such that $\rho(x) \geq \rho(y)$ for all $y \in \mathcal{B}_\epsilon(x) \cap S$.

Theorem 2.1. *If ρ has a local maximum on the unit sphere S at x then $B(x)x = \rho(x)x$ and $d(x) > 0$.*

Proof. Take y such that $y'y < \epsilon, x'y = 0$. Then

$$\tilde{x} \triangleq \frac{x + y}{\|x + y\|}$$

is in $\mathcal{B}_\epsilon(x) \cap S$, and

$$\rho(\tilde{x}) = \rho(x) + x'B(x)y + \sum_{k \in K_0} w_k d_k(y) + o(\|y\|),$$

where K_0 is the set of all k such that $d_k(x) = 0$. □

If $d(x) > 0$ we have for the first derivatives $\mathcal{D}\rho(x) = B(x)x$. Thus the first order necessary condition for a local maximum on the sphere at x is $B(x)x = \rho(x)x$, i.e. x is a normalized eigenvector of $B(x)$ with eigenvalue $\rho(x)$.

Again if $d(x) > 0$ we have second derivatives

$$\mathcal{D}^2\rho(x) = \sum_{j=1}^{m} \frac{w_i}{d_i(x)} \left[A_i - \frac{A_i x x' A_i}{x' A_i x}\right].$$

Note that $\mathcal{D}^2\rho(x)$ is positive semi-definite and that $\mathcal{D}^2\rho(x)x = 0$ for all x. Also, in the Loewner order, $\mathcal{D}^2\rho(x) \lesssim B(x)$ for all x.

The second order necessary condition for a local maximum at $x \in S$ is that $\mathcal{D}^2\rho(x) \lesssim \rho(x)I$, i.e. all eigenvalues of $\mathcal{D}^2\rho(x)$ must be less than or equal to $\rho(x)$.

It may seem that requiring $d(x) > 0$ limits the generality of our results. There are two ways of dealing with the problem that ρ is not differentiable if one or more of the $d_k(x)$ are zero. The first is to use subdifferentials, the second is to use the result that at a local maximum we have $d(x) > 0$.

3. Lower Bounds for ρ

By Cauchy-Schwartz $\sqrt{x' A_j x}\sqrt{y' A_j y} \geq x' A_j y$ with equality if $x = y$. Thus if $d(y) > 0$

$$\rho(x) \geq \eta(x, y) \overset{\Delta}{=} y' B(y) x,$$

and in fact

$$\rho(x) = \max_{y} \eta(x, y),$$

where the maximum is attained for $y = x$.

Define

$$\mu(y) \overset{\Delta}{=} \max_{x' x = 1} \eta(x, y) = \|B(y) y\|,$$

with $\|\bullet\|$ the Euclidean or Frobenius norm. It follows that

$$\max_{x'x=1}\rho(x) = \max_{x'x=1}\max_{y}\eta(x,y) = \max_{y}\max_{x'x=1}\eta(x,y) = \max_{y}\mu(y).$$

4. The SMACOF algorithm

The SMACOF algorithm developed for metric and non-metric multidimensional scaling [**deleeuw_C_77**, **deleeuw_heiser_C_77**, **deleeuw_A_88b**, **deleeuw_mair_A_09c**], is an iterative procedure in which we update our current iterate $x^{(k)}$ by the simple rule

$$y^{(k)} = B(x^{(k)})x^{(k)},$$
$$x^{(k+1)} = \frac{y^{(k)}}{\|y^{(k)}\|}.$$

Clearly

$$\rho(x^{(k+1)}) \geq \eta(x^{(k+1)},x^{(k)}) \geq \eta(x^{(k)},x^{(k)}) = \rho(x^{(k)}).$$

It is shown, first in **deleeuw_C_77**, that this forces convergence of the SMACOF sequence to a stationary point with $B(x)x = \rho(x)x$. **deleeuw_A_88b** shows that convergence is linear, but often slow with a convergence factor close to unity.

5. Upper bound for ρ

By the AMGM inequality $\sqrt{x'A_jx}\sqrt{y'A_jy} \leq \frac{1}{2}\{x'A_jx+y'A_jy\}$ with equality if $x = y$. Thus

$$\rho(x) \leq \theta(x,y) \triangleq \frac{1}{2}\{x'B(y)x + \rho(y)\},$$

and

$$\rho(x) = \min_{y}\theta(x,y),$$

with the minimum attained for $y = x$. Also define

$$\tau(x) \triangleq \max_{y'y=1}\theta(y,x) = \frac{1}{2}\{\|B(x)\|_\infty + \rho(x)\},$$

where $\|\bullet\|_\infty$ is the spectral norm, in this case the largest eigenvalue.

It follows that

$$\max_{x'x=1} \rho(x) \leq \tau(y)$$

for all y, i.e.

$$\max_{x'x=1} \rho(x) \leq \min_{y} \tau(y).$$

Also note that $\tau(x) \geq \theta(x,x) = \rho(x)$ for all x.

6. Detecting a Global Maximum

Suppose $\overline{x} \in S$. Then

$$\rho(\overline{x}) \leq \max_{x'x=1} \rho(x) \leq \tau(\overline{x}).$$

It follows that if $\rho(\overline{x}) = \tau(\overline{x})$ then $\overline{x}$ is the global maximizer of ρ on the unit sphere.

At a limit point $\overline{x}$ of the SMACOF sequence we have $B(\overline{x})\overline{x} = \rho(\overline{x})\overline{x}$. If $\rho(\overline{x})$ is actually the largest eigenvalue of $B(\overline{x})$ then $\rho(\overline{x}) = \tau(\overline{x})$ and $\overline{x}$ is the global maximizer.

SMACOF may not give us a point with $\rho(x) = \tau(x)$, or it may only give us that point from certain starting points. We can always use other algorithms, such as Newton's method or inverse iteration, to try to find a point with $\rho(x) = \tau(x)$, or, equivalently, with $\rho(x)$ the largest eigenvalue of $B(x)$.

What we find in any case is an upper bound for the global maximum. But of course this upper bound may be useless, because it may be larger than one. Another strategy would be to use some other optimization method to directly minimize τ over the unit sphere.

6.1. **Relaxation.**

The global maximum result can be shown in yet another way, using somewhat more complicated machinery than the elementary inequalities we used so far. A concave relaxation of our optimization problem is to maximize $\rho(C) \triangleq \sum_{j=1}^{m} w_j \sqrt{\mathbf{tr}\ A_j C}$

over all $C \gtrsim 0$ with $\mathbf{tr}\ C = 1$. The necessary and sufficient conditions for a maximum are

$$\begin{aligned} B(C) - \rho(C)I &\lesssim 0,\\ \mathbf{tr}\ C(B(C) - \rho(C)I) &= 0,\\ C &\gtrsim 0,\\ \mathbf{tr}\ C &= 1, \end{aligned}$$

where, of course,

$$B(C) \triangleq \sum_{j=1}^{m} \frac{w_j}{\sqrt{\mathbf{tr}\ A_jC}} A_j.$$

The solution is of the form $C = xx'$, with $x'x = 1$, if and only if $\rho(C)$ is the largest eigenvalue of $B(C)$, and x is the corresponding normalized eigenvector. In that case we have also found the global maximum of ρ over $x'x = 1$.

6.2. Full Dimensional Scaling.

6.3. Global Minimum in a Plane.

7. Code

7.1. Parametrization.

7.2. SMACOF.

7.3. Auxilaries.

8. Examples

8.1. Perfection. We start with a simple example in which the global maximum is known, because we can attain perfect fit. The dissimilarities are actual distances between points in the plane.

INSERT FIGURE 1 ABOUT HERE

$\boxed{\text{INSERT FIGURE 2 ABOUT HERE}}$

$\boxed{\text{INSERT TABLE 1 ABOUT HERE}}$

$\boxed{\text{INSERT TABLE 2 ABOUT HERE}}$

8.2. **Ekman.** As our example we use the Ekman color data. **ekman_54** collected data to study perception of 14 different colors. Starting from wavelength 434, the colors range from bluish-purple, blue, green, yellow, to red. Every pair of colors was judged by 31 respondents from having "no similarity" to being "identical". Similarity judgments are scaled from 0 (no similarity) to 1 (identical). The averages of the similarities over the 31 respondents constitute the data, which are available, for example, in the R package `smacof` [**deleeuw_mair_A_09c**].

Because the data are similarities they have to be transformed by an inverse monotone transformation before we can fit distances to them. The usual transformation people use is $\delta_{ij} = 1 - s_{ij}$, but to a large extent the choice is arbitrary (which is one of the main reasons nonmetric multidimensional scaling was invented).

$\boxed{\text{INSERT FIGURE 3 ABOUT HERE}}$

$\boxed{\text{INSERT FIGURE 4 ABOUT HERE}}$

$\boxed{\text{INSERT TABLE 3 ABOUT HERE}}$

$\boxed{\text{INSERT TABLE 4 ABOUT HERE}}$

1.0565

8.3. **Ekman Transformed.** Alternatively, we can monotonically transform the Ekman data to get a better fit. Taking our clue from Figure 3 we use the transformation $(1 - s_{ij})^3$. The results are ... and it is clear that in this case the SMACOF algorithm fid the global maximum of ρ.

INSERT FIGURE 5 ABOUT HERE

INSERT FIGURE 6 ABOUT HERE

INSERT TABLE 5 ABOUT HERE

INSERT TABLE 6 ABOUT HERE

8.4. **Equal Dissimilarities.** 1.1537

INSERT FIGURE 7 ABOUT HERE

INSERT FIGURE 8 ABOUT HERE

INSERT TABLE 7 ABOUT HERE

INSERT TABLE 8 ABOUT HERE

8.5. **Political Parties.**

INSERT FIGURE 9 ABOUT HERE

INSERT FIGURE 10 ABOUT HERE

INSERT TABLE 9 ABOUT HERE

INSERT TABLE 10 ABOUT HERE

APPENDIX A. CODE

A.1. Code for Subsection 7.1.

```r
make_x<-function (n,p) {
r<-(p*n)-(p*(p+1)/2); l<-1
x<-array(0,c(n,p,r))
for (i in 1:p)
  {
  qrq<-as.matrix(qr.Q(qr(outer(1:(n-i+1),0:(n-i),"^")))[,2:(n-i+1)])
        for (k in 1:(n-i))
                {
                x[1:(n-i+1),i,l]<-qrq[,k]
                l<-l+1
                }
        }
return(x/sqrt(n))
}

make_a_from_x<-function(x) {
n<-dim(x)[3]; m<-dim(x)[1]; mm<-m*(m-1)/2
c<-array(0,c(n,n,mm))
for (s in 1:n) for (t in 1:n)
        {
        prd<-x[,,s]%*%t(x[,,t]); k<-1
        for (i in 1:(m-1)) for (j in (i+1):m)
                {
                c[s,t,k]<-prd[i,i]+prd[j,j]-(prd[i,j]+prd[j,i])
                k<-k+1
                }
        }
return(c)
}
```

A.2. Code for Subsection 7.2.

```
smacof <- function (d, a, x, imx = 9999, eps = 1e-10, out = 5) {
  itel <- 1
  rval <- NULL
  tval <- NULL
  aval <- NULL
  xold <- x
        repeat {
                xnew <- smacof_up (d, a, xold)
    rval <- c (rval, rho (d, a, xnew))
    tval <- c (tval, bnd (d, a, xnew))
    aval <- c (aval, aps <- max (abs (xold - xnew)))
                if ((itel == imx) || (aps < eps)) break ()
                itel <- itel + 1
            xold <- xnew
        }
    for (i in c(1 : out,(itel- out + 1):itel))
    cat (formatC (i, width = 4, format = "d"),
      formatC (aval[i], di = 10, wi = 13, fo = "f"),
      formatC (rval[i], di = 10, wi = 13, fo = "f"),
      formatC (tval[i], di = 10, wi = 13, fo = "f"), "\n")
    return (xnew)
}
```

A.3. **Code for Subsection 7.3.**

```
dvec <- function (a, x) {
  m <- dim (a)[3]
  d <- rep (0, m)
        for (i in 1 : m) {
      d[i] <- sqrt ( sum (x * (a[,,i]%*% x)))
        }
        return (d)
}

rho <- function (delta, a, x) {
        sum(delta * dvec (a, x))
}

bnd <- function (delta, a, x) {
  me <- max (eigen (bmat (delta, a, x), o = TRUE) $ values)
  return ((rho (delta, a, x) + me) / 2)
}

bmat <- function (delta, a, x) {
  m <- dim(a)[3]
  n <- dim(a)[1]
        b <- matrix (0, n, n)
        d <- dvec (a, x)
        for (i in 1:m) {
                b <- b + (delta[i]/d[i])*a[,,i]
        }
        return (b)
}

smacof_up <- function (delta, a, x) {
        y <- bmat (delta, a, x) %*% x
    return (y / sqrt (sum (y ^ 2)))
}
```

APPENDIX B. FIGURES

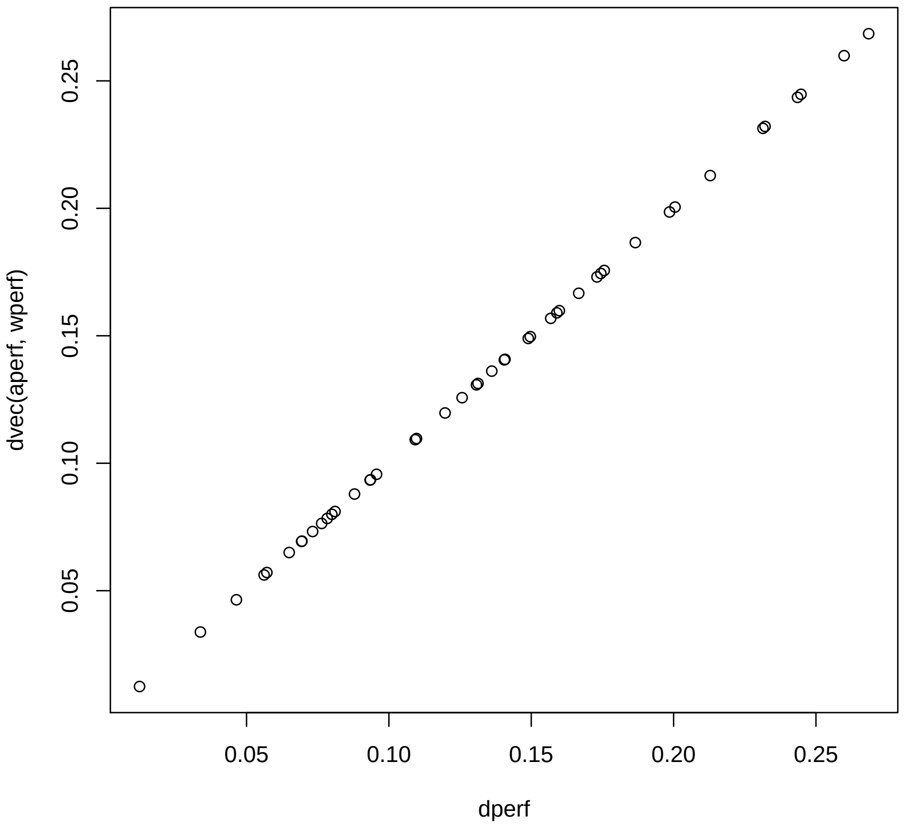

FIGURE 1. Fitplot Perfect Example

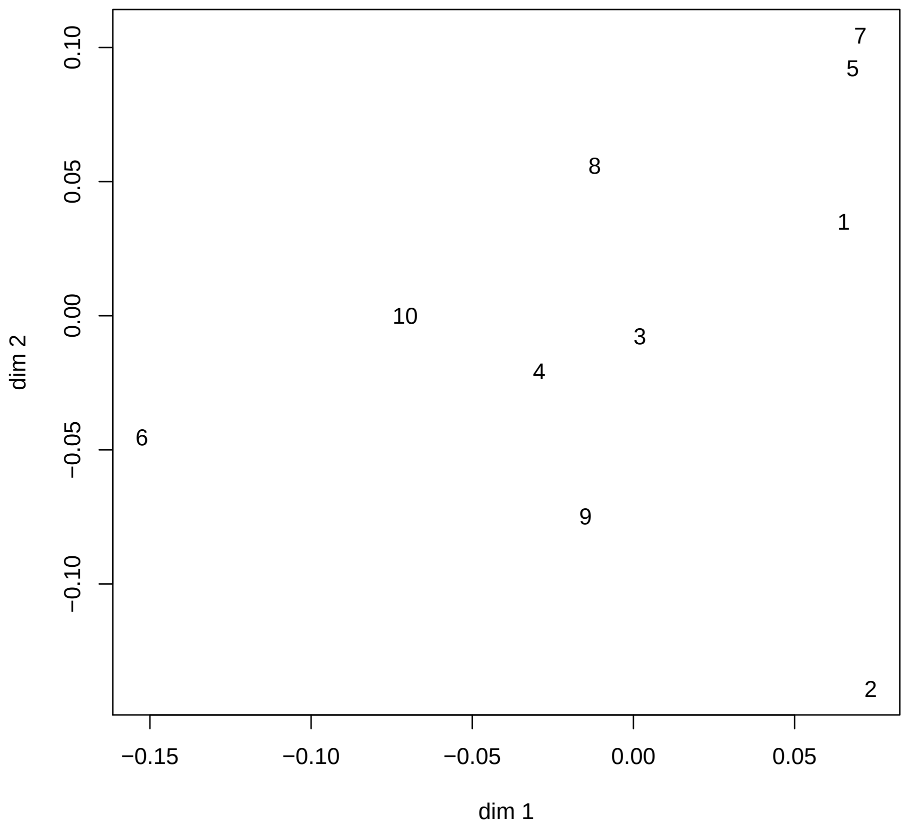

FIGURE 2. Configuration Plot Perfect Example

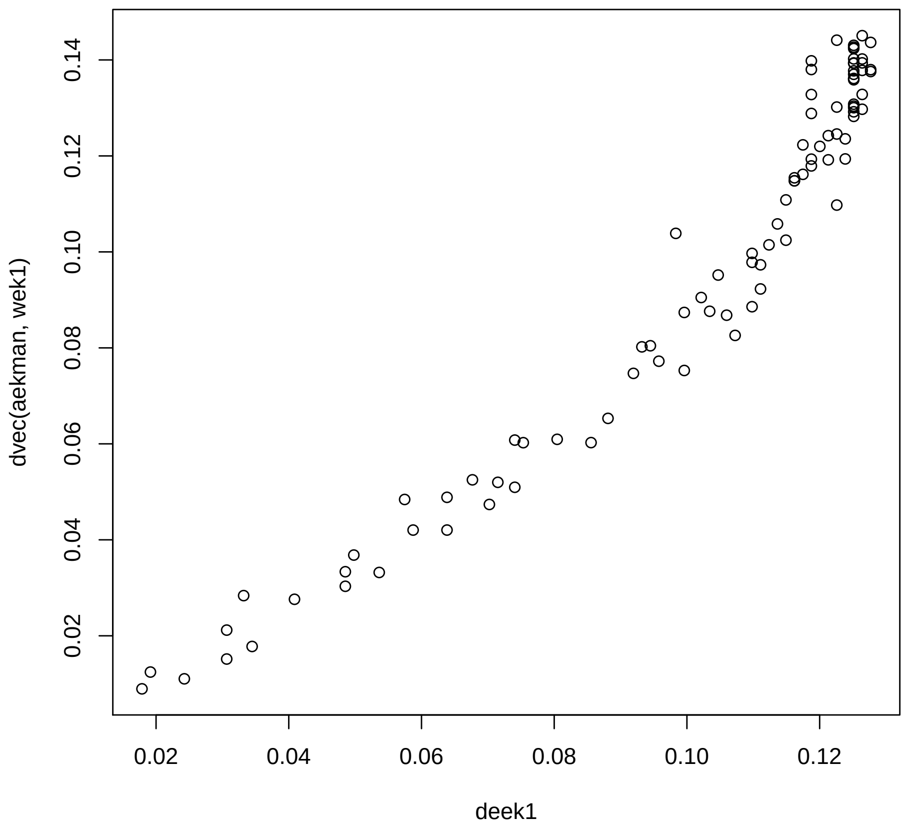

FIGURE 3. Fitplot Ekman Example

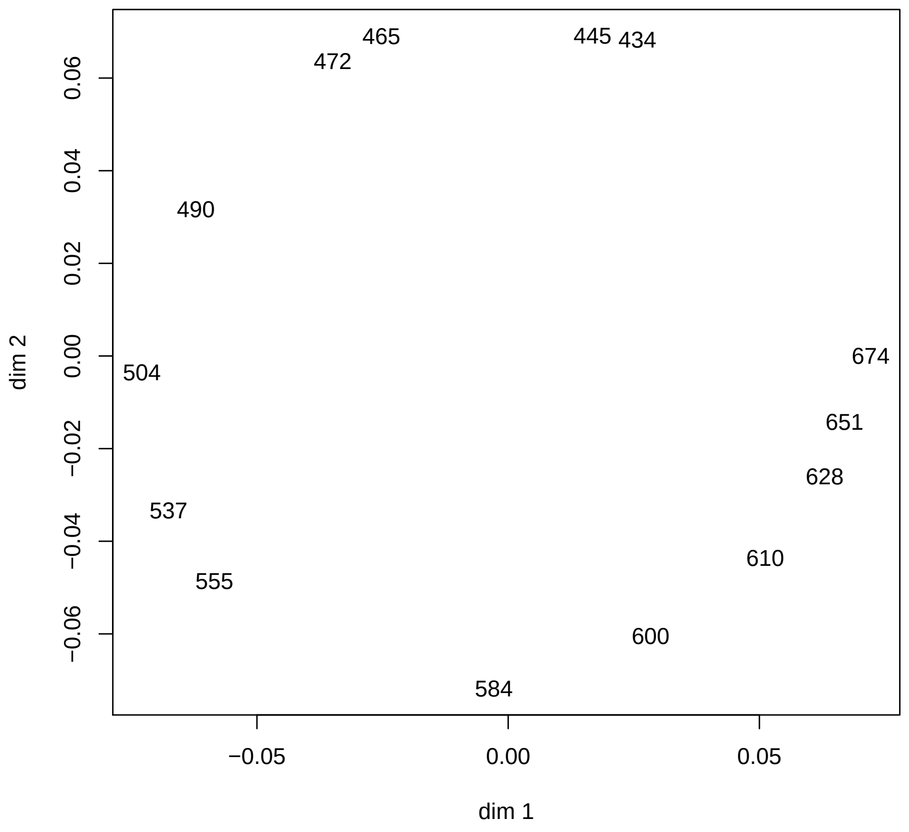

FIGURE 4. Configuration Plot Ekman Example

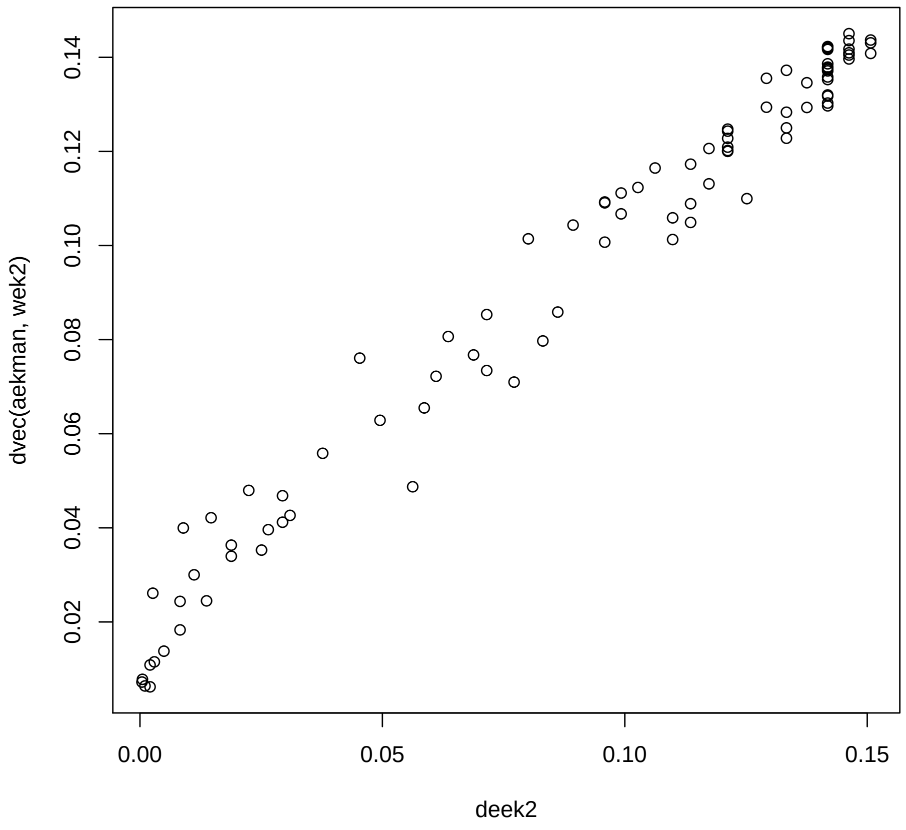

FIGURE 5. Fit Plot Transformed Ekman Example

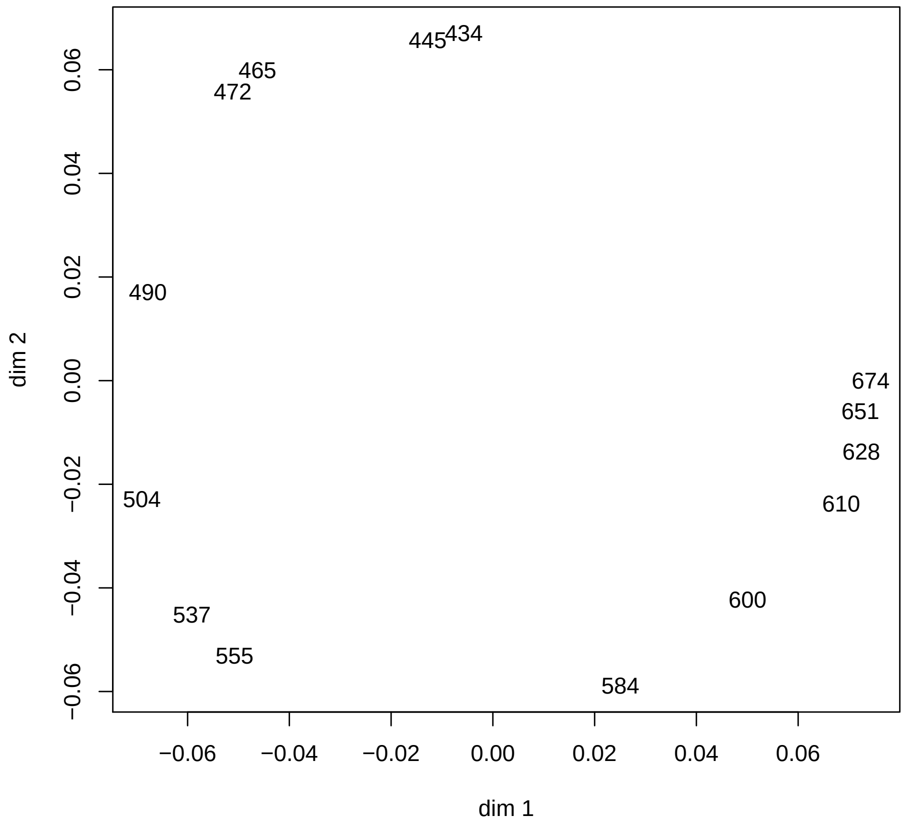

FIGURE 6. Configuration Plot Transformed Ekman Example

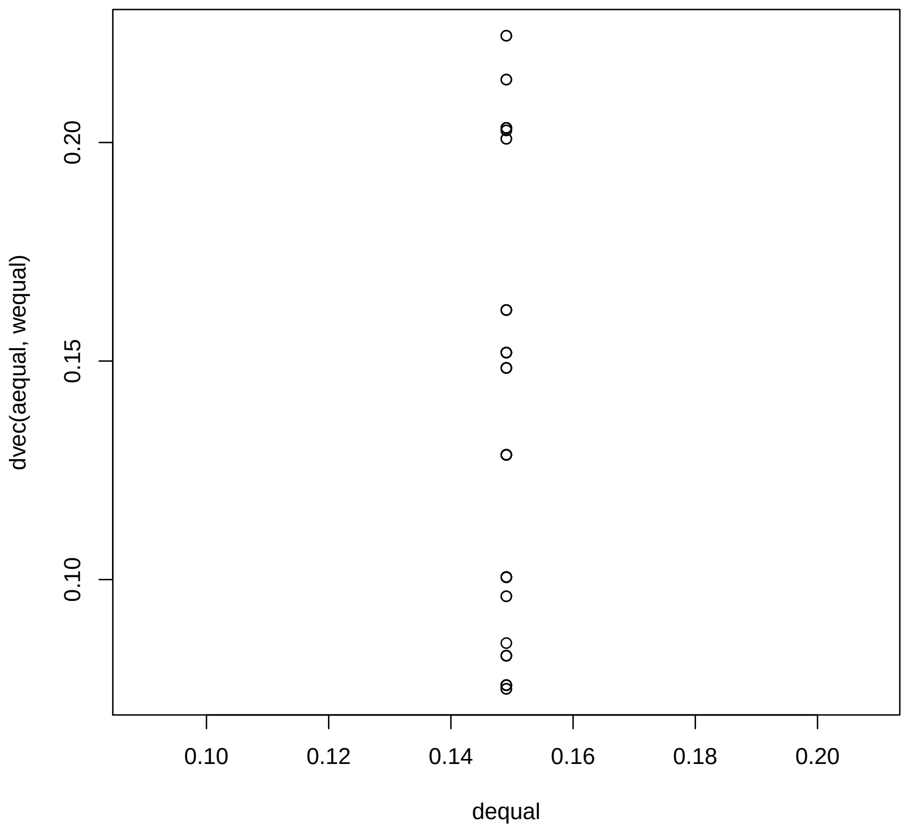

FIGURE 7. Fit Plot Equal Dissimilarity Example

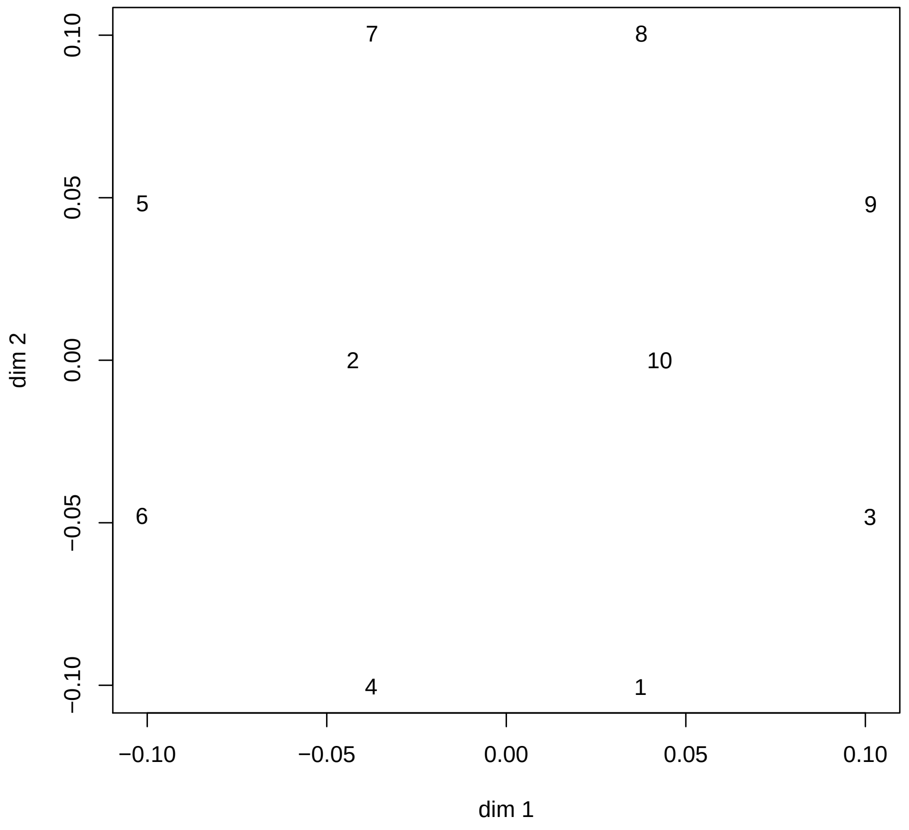

FIGURE 8. Configuration Plot Equal Dissimilarity Example

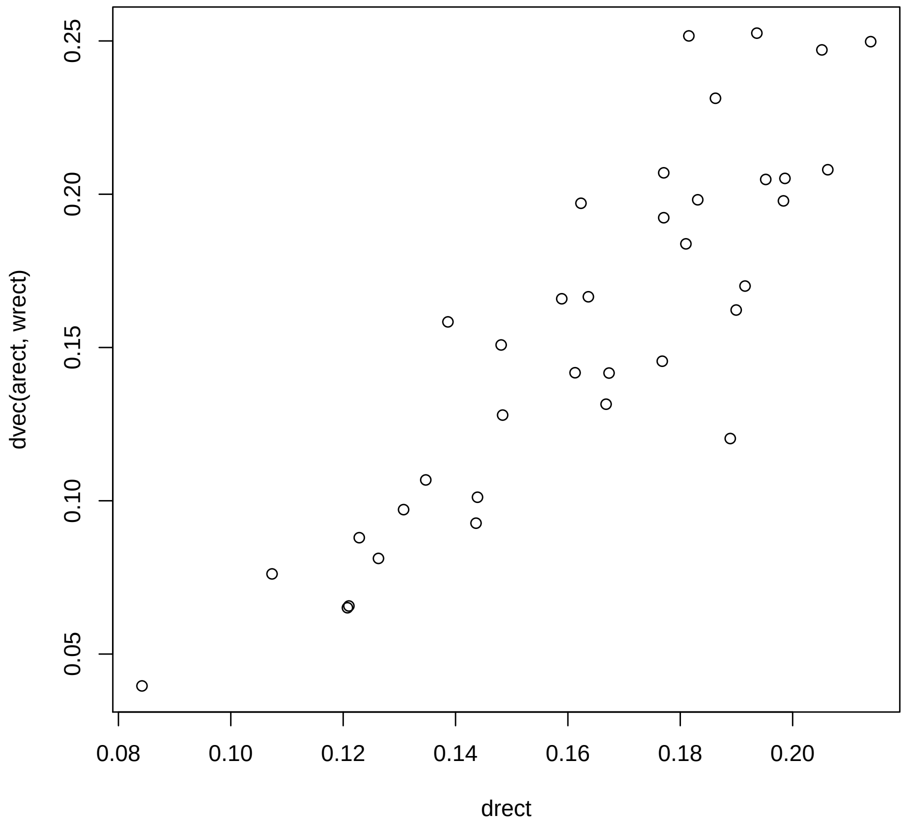

FIGURE 9. Fit Plot Political Parties Example

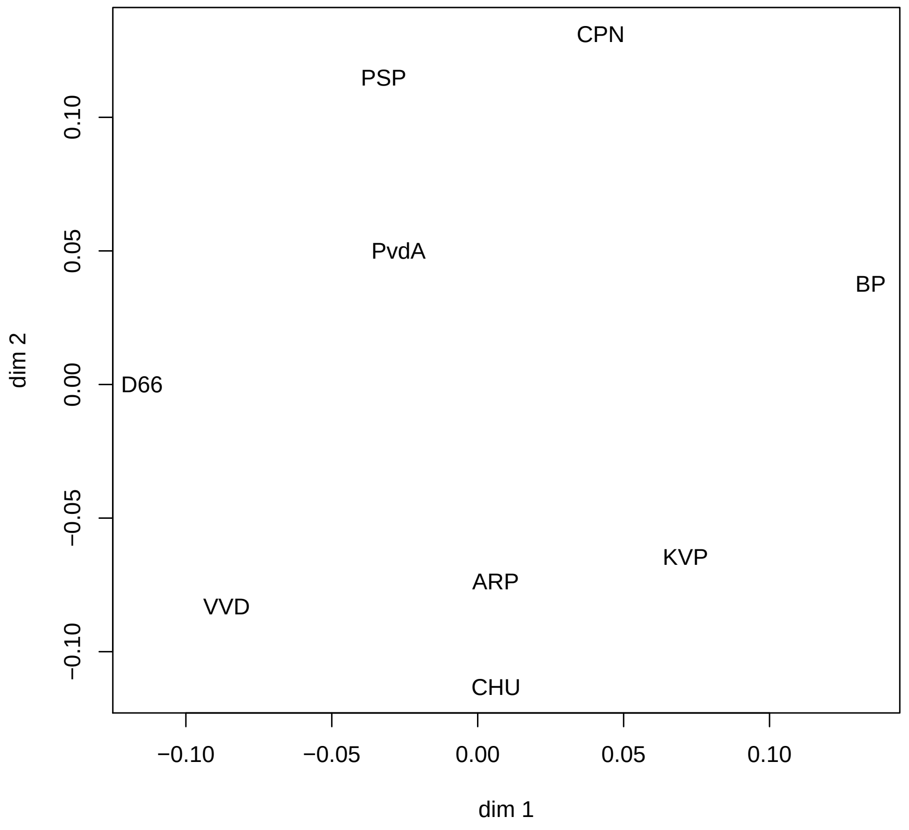

FIGURE 10. Configuration Plot Poltical Parties Example

Appendix C. Tables

```
##    1  0.2828859387  0.8627966401  1.6710789158
##    2  0.0883945234  0.8978517322  1.4443008794
##    3  0.1024874502  0.9329287702  1.1938014563
##    4  0.0490252381  0.9438094448  1.2247689847
##    5  0.0343674044  0.9500374955  1.2983859077
## 1126  0.0000000001  1.0000000000  1.0000000015
## 1127  0.0000000001  1.0000000000  1.0000000015
## 1128  0.0000000001  1.0000000000  1.0000000015
## 1129  0.0000000001  1.0000000000  1.0000000015
## 1130  0.0000000001  1.0000000000  1.0000000014
```

Table 1. Iterations Perfect Fit Example

```
##       eval_b            eval_d
##  [1,] 1.000000002871871 0.985137490254618
##  [2,] 1.000000001026648 0.865746878273199
##  [3,] 1.000000000874269 0.852817023999527
##  [4,] 1.000000000439181 0.813990790816734
##  [5,] 1.000000000296371 0.732808283436156
##  [6,] 1.000000000253265 0.708543670657466
##  [7,] 1.000000000251492 0.658064928770238
##  [8,] 1.000000000153323 0.621950668138150
##  [9,] 1.000000000024888 0.393553850448150
## [10,] 1.000000000024694 0.358066829687699
## [11,] 1.000000000014704 0.333183794870067
## [12,] 1.000000000014658 0.289709433096711
## [13,] 0.999999999964475 0.233228338634835
## [14,] 0.999999999840433 0.189885389216342
## [15,] 0.999999999656406 0.138867036737792
## [16,] 0.999999999366333 0.131411165218272
## [17,] 0.999999999022164 0.000000000000000
```

Table 2. Eigenvalues Perfect Fit Example

```
##    1  0.1367982202  0.8961513961  1.5285728761
##    2  0.0516321456  0.9072784567  1.6143430672
##    3  0.0373024738  0.9142947586  1.7092188277
##    4  0.0363891971  0.9208090203  1.6034453538
##    5  0.0433561022  0.9291440040  1.5161422656
##  630  0.0000000001  0.9913560174  1.1211412689
##  631  0.0000000001  0.9913560174  1.1211412689
##  632  0.0000000001  0.9913560174  1.1211412689
##  633  0.0000000001  0.9913560174  1.1211412689
##  634  0.0000000001  0.9913560174  1.1211412689
```

TABLE 3. Iterations Ekman Example

```
##       eval_b           eval_d
##  [1,] 1.250926520444054 0.956555551110624
##  [2,] 1.250850464699742 0.742591322609435
##  [3,] 1.233584530386040 0.741596486100723
##  [4,] 1.213627350885557 0.712340690852582
##  [5,] 1.213537930909448 0.694124899319038
##  [6,] 1.206885262008840 0.680020733480407
##  [7,] 1.206883360223037 0.679184461704678
##  [8,] 1.196060010926107 0.648679320924662
##  [9,] 1.166098307047335 0.648067662284340
## [10,] 1.165038150399025 0.642157423620363
## [11,] 1.164633467773975 0.618335860699482
## [12,] 1.157434036340699 0.606007907192313
## [13,] 1.156504236885987 0.565302721631988
## [14,] 1.144218607531108 0.539005963198279
## [15,] 1.139640209751063 0.521219432580985
## [16,] 1.138216490520971 0.506542578259075
## [17,] 1.124171972025897 0.501711463301747
## [18,] 1.124021686883923 0.501189815725186
## [19,] 1.123373646956519 0.486075947280501
## [20,] 1.111418129925843 0.476839357064972
## [21,] 1.110028216405279 0.471685093109118
## [22,] 1.018475613681323 0.464352071102484
## [23,] 0.991356017585882 0.459055074251864
## [24,] 0.991356017422441 0.453438402811117
## [25,] 0.991356017258188 0.000000000000002
```

TABLE 4. Eigenvalues Ekman Example

```
##    1  0.1602105541  0.8468510676  2.2510715548
##    2  0.0786029230  0.8798238860  1.6356867057
##    3  0.1216613204  0.9160458627  1.3027771443
##    4  0.1087884329  0.9422601407  1.1872068272
##    5  0.0534601459  0.9526880553  1.1927043524
##  467  0.0000000001  0.9944723164  0.9944723165
##  468  0.0000000001  0.9944723164  0.9944723165
##  469  0.0000000001  0.9944723164  0.9944723165
##  470  0.0000000001  0.9944723164  0.9944723165
##  471  0.0000000001  0.9944723164  0.9944723165
```

TABLE 5. Iterations Transformed Ekman Example

```
##       eval_b           eval_d
##  [1,] 0.994472316589279 0.955272432394409
##  [2,] 0.994472316441239 0.532179686845551
##  [3,] 0.994472316275627 0.528325169961513
##  [4,] 0.975276641636359 0.524928936520574
##  [5,] 0.918392286760243 0.516911963381317
##  [6,] 0.914545534180620 0.513992971953656
##  [7,] 0.902882622745617 0.510944562443364
##  [8,] 0.892978107681614 0.507629253089380
##  [9,] 0.858166545668940 0.503558722710835
## [10,] 0.851317929012582 0.493368971788030
## [11,] 0.847978592424081 0.478858664963231
## [12,] 0.826341693229653 0.467256905843421
## [13,] 0.825216729078988 0.443060594930487
## [14,] 0.812369892118908 0.409221840078421
## [15,] 0.810053558860483 0.392194988805236
## [16,] 0.802334235428847 0.359533704238104
## [17,] 0.788853804550027 0.346528760530640
## [18,] 0.788571458870336 0.328515493691904
## [19,] 0.787275722639679 0.311790775120505
## [20,] 0.785514438894769 0.286011075866976
## [21,] 0.782095473190072 0.279512770622884
## [22,] 0.747373100628323 0.265812457042889
## [23,] 0.743546540859478 0.246547251488382
## [24,] 0.724272714694104 0.215386002325382
## [25,] 0.724242611647812 0.000000000000000
```

TABLE 6. Eigenvalues Transformed Ekman Example

```
##    1  0.2073043988  0.9252419183  1.3692527715
##    2  0.0499881186  0.9344437181  1.2758373661
##    3  0.0303845837  0.9378386577  1.2444455790
##    4  0.0210616372  0.9395114759  1.2320125138
##    5  0.0158712289  0.9405357102  1.2287380149
## 9995  0.0000004204  0.9428403021  1.2251844552
## 9996  0.0000004201  0.9428403021  1.2251844551
## 9997  0.0000004199  0.9428403021  1.2251844551
## 9998  0.0000004196  0.9428403021  1.2251844550
## 9999  0.0000004193  0.9428403021  1.2251844549
```

TABLE 7. Iterations Equal Dissimilarities Example

```
##       eval_b            eval_d
##  [1,] 1.507528607732107 0.942204978711711
##  [2,] 1.450716684285295 0.939457865472898
##  [3,] 1.402118806157293 0.912444379575352
##  [4,] 1.295092107761670 0.794520786818371
##  [5,] 1.295092106884599 0.710334080871150
##  [6,] 1.284232482903013 0.708661588556831
##  [7,] 1.284174610233577 0.624708012592077
##  [8,] 1.239139865652684 0.611057390513324
##  [9,] 1.239139863071964 0.588144851458982
## [10,] 1.170199558981717 0.578670417061077
## [11,] 1.167244671882294 0.539084161968712
## [12,] 1.159487798277609 0.513973142478121
## [13,] 1.159487732691924 0.463592087997280
## [14,] 0.963130470040080 0.451312791272332
## [15,] 0.942841132490394 0.435070069818116
## [16,] 0.942840302104113 0.374811249371456
## [17,] 0.942839471563858 0.000000000000000
```

TABLE 8. Eigenvalues Equal Dissimilarities Example

```
##    1  0.1197560393  0.9219482120  1.7146252500
##    2  0.0614805420  0.9362799675  1.3317553643
##    3  0.0441933052  0.9453063199  1.2384663035
##    4  0.0331834844  0.9495161940  1.2481757071
##    5  0.0232007321  0.9516157157  1.2717161587
##  713  0.0000000001  0.9775327625  1.1931096241
##  714  0.0000000001  0.9775327625  1.1931096241
##  715  0.0000000001  0.9775327625  1.1931096241
##  716  0.0000000001  0.9775327625  1.1931096241
##  717  0.0000000001  0.9775327625  1.1931096241
```

TABLE 9. Iterations Political Parties Example

```
##       eval_b            eval_d
##  [1,] 1.408686485808786 0.957366809853854
##  [2,] 1.408271968068399 0.940005223961766
##  [3,] 1.289190398501312 0.908923270278075
##  [4,] 1.284756326849329 0.841835081792442
##  [5,] 1.249477953332152 0.745848211558783
##  [6,] 1.249458262592434 0.692558258361098
##  [7,] 1.153443157524002 0.555393535206836
##  [8,] 1.145854901855972 0.549355043170663
##  [9,] 1.129486865694092 0.491590903210073
## [10,] 1.118891530325019 0.477382546831601
## [11,] 1.102175889468110 0.439231703398564
## [12,] 1.022866672364758 0.388156923076463
## [13,] 0.977532762607197 0.376916935457207
## [14,] 0.977532762506475 0.295384011669072
## [15,] 0.977532762396071 0.000000000000001
```

TABLE 10. Eigenvalues Political Parties Example

DEPARTMENT OF STATISTICS, UNIVERSITY OF CALIFORNIA, LOS ANGELES, CA 90095-1554

E-mail address, Jan de Leeuw: `deleeuw@stat.ucla.edu`

URL, Jan de Leeuw: `http://gifi.stat.ucla.edu`