# PLSC 597: Modern Measurement

Scaling

February 1, 2018

# The Plan

- Scaling Overview
- Unidimensional Scaling
- Scale Reliability
- Multidimensional Scaling

# Some Terms

- Always $i \in \{1, 2, ...N\}$ observations on $k \in \{1, 2, ...K\}$ indicators.
- $Z$ or $Z_1, Z_2, ...$ will indicate the underlying / latent trait(s) / phenomena
- $D_1, D_2, ...D_K$ are *dichotomous* indicators
- $Y_1, Y_2, ...Y_K$ are *continuous* indicators

# Scaling, In General

**Characteristics:**

- *Comparative* vs. *Non-Comparative* Scaling
- *Subject-centered* vs. *Stumulus-centered*
- *Metric* vs. *Non-Metric*
- *Unidimensional* vs. *Multidimensional*

# (Unidimensional) Scaling: History

**Thurstone (1927, 1929)**

- Non-comparative, subject-centered
- "Law of Comparative Judgment": The degree to which any two stimuli can be discriminated is a direct function of the difference in their status as regards the attribute in question.
- Methods: *paired comparisons*, *successive intervals*, and *equal-appearing intervals*.

**Likert (1932)**

- Non-comparative, subject-centered
- *Summative* + *unidimensional* $\rightarrow$ item construction & selection are key

**Gutman (1944, 1950)** ("scalogram analysis")

- Comparative; both subject- and stimulus-centered
- The response to each item is a simple function of the sum score

See McIver and Carmines (1981) for more...

# Scaling: Dissimilarities and Distances

Dissimilarities Matrix:

$$\Delta = \begin{bmatrix} 0 & \delta_{21} & \delta_{31} & \dots & \delta_{K1} \\ \delta_{12} & 0 & \delta_{32} & \dots & \delta_{K2} \\ \delta_{13} & \delta_{23} & 0 & \dots & \delta_{K3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_{1K} & \delta_{2K} & \delta_{3K} & \dots & 0 \end{bmatrix}$$

where

$$\delta_{ij} = f(\mathbf{Y}_i - \mathbf{Y}_j)$$

# Distance Examples

Euclidean ("L2") Distance:

$$\delta_{ij} = \sqrt{(Y_{i1} - Y_{j1})^2 + (Y_{i2} - Y_{j2})^2 + ...(Y_{iN} - Y_{jN})^2}$$

Manhattan ("L1") Distance:

$$\delta_{ij} = |Y_{i1} - Y_{j1}| + |Y_{i2} - Y_{j2}| + ...|Y_{iN} - Y_{jN}|$$

Minkowski Order-$p$ ("$L^p$") Distance:

$$\delta_{ij} = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

**Goal: Location $N$ points in a low ($p$)-dimensional space such that the Euclidean distances between them approximate $\Delta$.**

That is, find a set of $N \times P$ points $\mathbf{D}$ such that $d_{ij}(\mathbf{D}) \approx \delta_{ij} \; \forall \, i, j$, where

$$d_{ij}(\mathbf{D}) = \sqrt{\sum_{p=1}^{P} (x_{ip} - x_{jp})^2}$$

and $p = \{1, 2, ...P\}$ denotes the dimensionality of the space.

For a $K \times K$ dissimilarity matrix $\Delta$, the *stress* associated with a given set of coordinates $\mathbf{D}$ is:

$$\sigma(\mathbf{D}) = \sum_{i<j} w_{ij}[\delta_{ij} - d_{ij}(\mathbf{D})]^2$$

with weights

$$\mathbf{W} = \begin{bmatrix} 0 & w_{21} & w_{31} & \dots & w_{N1} \\ w_{12} & 0 & w_{32} & \dots & w_{N2} \\ w_{13} & w_{23} & 0 & \dots & w_{N3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{1N} & w_{2N} & w_{3N} & \dots & 0 \end{bmatrix}$$

such that

$$\sum_{i<j} w_{ij}\delta_{ij}^2 = \frac{N(N-1)}{2}.$$

# Scaling Types

Key: Transformations of the dissimilarities.

**Ratio**:

- $\hat{d}_{ij} = \delta_{ij}$ (no transformation)
- A special case of *metric* scaling

**Interval**:

- $\hat{d}_{ij} = a + b(\delta_{ij})$ (linear transformation)
- Also *metric*; "the ratio of differences of distances should be equal to the corresponding ratio of differences in the data"

**Nonlinear**:

- $\hat{d}_{ij} = g(\delta_{ij})$ (e.g., splines)
- Also *metric*

**Ordinal**:

- $\hat{d}_{ij} = f(\delta_{ij})$ such that $\delta_{ij} < \delta_{i'j'} \Rightarrow f(\delta_{ij}) < f(\delta_{i'j'})$
- *Monotone* / rank-preserving transformations
- *Nonmetric*

# Unidimensional Scaling: The Sum Score

Simplest approach: the *sum score*:

$$\hat{Z}_i = \sum_{k=1}^{K} Y_{ik}$$

or

$$\hat{Z}_i = \frac{\sum_{k=1}^{K_i} Y_{ik}}{K_i}.$$

Requires:

- $Var(Y_j) = Var(Y_k) \, \forall \, j \neq k$
- $Cov(Y_j, Z) = Cov(Y_k, Z) \, \forall \, j \neq k$

For $p = 1$, the stress

$$\sigma(\mathbf{D}) = \sum_{i<j} w_{ij}[\delta_{ij} - d_{ij}(\mathbf{D})]^2$$

can be shown to be a function of the orders of the rank orders of the items (see Mair and deLeeuw 2014 for details). This means:

- No need to chose among distances / similarities.
- Solution is via combinatorics.

# Example: Cities

```
> Cities[,c(1,4)]
                      city  longitude
New York          New York  -74.00594
Los Angeles   Los Angeles -118.24368
Chicago           Chicago  -87.62980
Houston           Houston  -95.36980
Philadelphia Philadelphia  -75.16522
Phoenix           Phoenix -112.07404
San Antonio   San Antonio  -98.49363
San Diego       San Diego -117.16108
Dallas             Dallas  -96.79699
San Jose         San Jose -121.88633
```

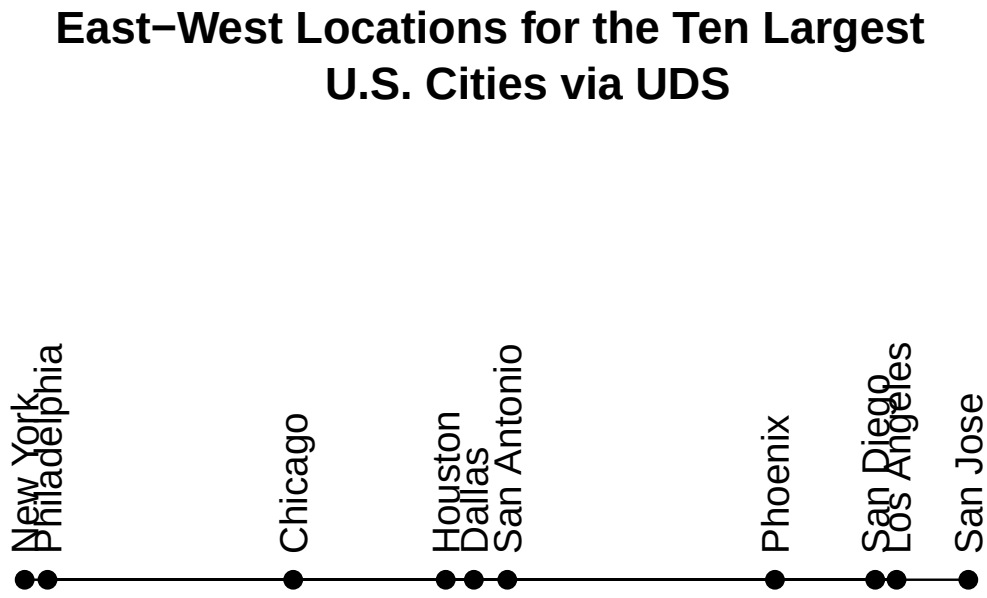
# Cities: Euclidean Distance Matrix ($\Delta$)

```
> CityLong <- data.frame(t(Cities$longitude)) # longitudes in a row
> colnames(CityLong) <- t(Cities$city) # names
> D1long <- dist(t(CityLong)) # distance object
> D1long
              New York Los Angeles Chicago Houston Philadelphia Phoenix San Antonio San Diego Dallas
Los Angeles       44.2
Chicago           13.6        30.6
Houston           21.4        22.9     7.7
Philadelphia       1.2        43.1    12.5    20.2
Phoenix           38.1         6.2    24.4    16.7         36.9
San Antonio       24.5        19.8    10.9     3.1         23.3    13.6
San Diego         43.2         1.1    29.5    21.8         42.0     5.1        18.7
Dallas            22.8        21.4     9.2     1.4         21.6    15.3         1.7      20.4
San Jose          47.9         3.6    34.3    26.5         46.7     9.8        23.4       4.7   25.1
```

# Unidimensional Scaling

```
> library(smacof)
> UDS <- uniscale(D1long)
> UDS

Call: uniscale(delta = D1long)

Final stress value: 4.1e-16
Number of accepted permutations: 180836
Number of possible permutations: 3628800
Number of objects: 10

> UDS$conf
    New York  Los Angeles      Chicago      Houston Philadelphia
    -1.04236      0.75350     -0.48929     -0.17508     -0.99530
     Phoenix  San Antonio    San Diego       Dallas     San Jose
     0.50304     -0.04827      0.70955     -0.11715      0.90137
```

# Cities: UDS Coordinates



# Another Example: SCOTUS Votes

```
> head(SCOTUS)
  id Rehnquist Stevens OConnor Scalia Kennedy Souter Thomas Ginsburg Breyer
  1          0       1       0      0       0      0      0        0      0
  2          0       0       0      0       0      0      0        0      0
  3          0       0       0      0       0      0      0        0      0
  4          1       1       1      1       1      1      1        1      1
  5          0       1       0      0       1      1      0        1      1
  7          1       1       1      0       1      1      0        1      1
```

# SCOTUS: Sum Scores and Distances

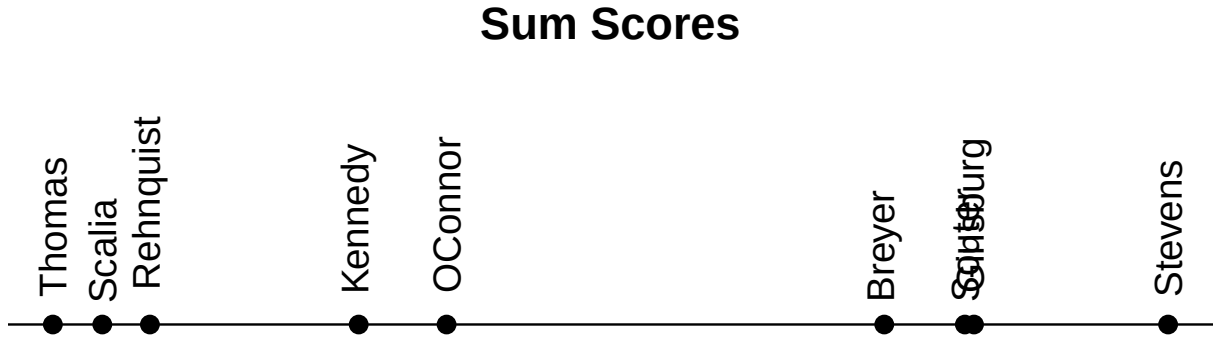
```
> SumScores <- colSums(SCOTUS[,2:10],na.rm=TRUE) / nrow(SCOTUS)
> SumScores
Rehnquist   Stevens   OConnor    Scalia   Kennedy    Souter    Thomas  Ginsburg  Breyer
   0.2838    0.6906    0.4023    0.2648    0.3672    0.6094    0.2451    0.6130  0.5772
```

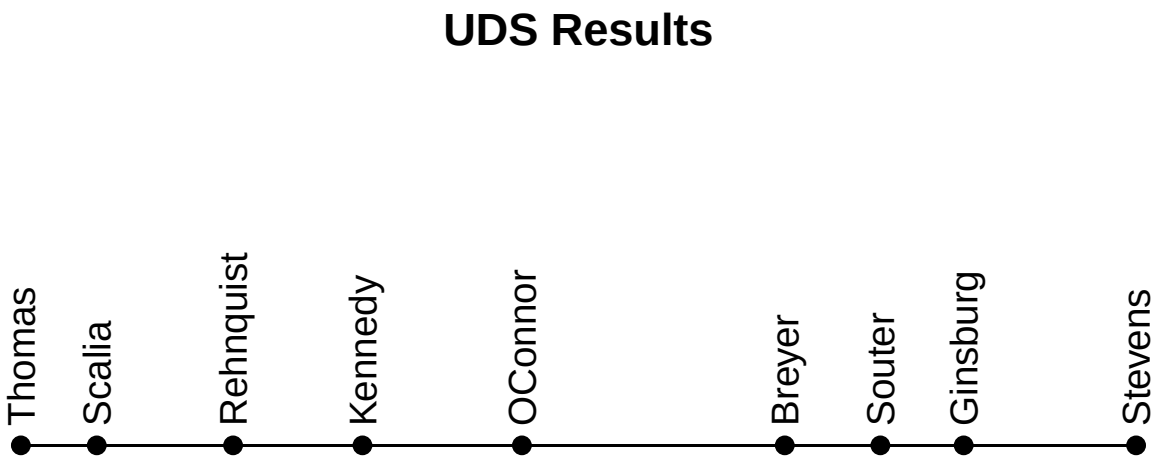
```
> D1SCOTUS <- dist(t(SCOTUS[,2:10]))
> D1SCOTUS
          Rehnquist Stevens OConnor Scalia Kennedy Souter Thomas Ginsburg
Stevens       25.61
OConnor       15.94   22.32
Scalia        13.49   26.80   18.38
Kennedy       13.27   23.62   16.31  15.62
Souter        22.52   15.39   18.73  23.56   20.57
Thomas        13.45   26.81   18.36  10.15   16.03  23.96
Ginsburg      22.72   15.10   19.65  24.00   20.88  11.27  24.64
Breyer        22.29   16.09   18.25  24.02   20.66  13.42  24.78    12.69
```

# SCOTUS: UDS

```
> SCOTUS.UDS <- uniscale(D1SCOTUS)

> SCOTUS.UDS

Call: uniscale(delta = D1SCOTUS)

Final stress value: 0.317
Number of accepted permutations: 347136
Number of possible permutations: 362880
Number of objects: 9
```

# Sum Scores and UDS



# Reliability: Cronbach's $\alpha$

For a scale $S$ that is a sum of $K$ separate items $Y_1, Y_2, \ldots Y_K$,

$$\alpha = \frac{K}{K-1}\left(1 - \frac{\sum_{k=1}^{K} \sigma^2_{Y_k}}{\sigma^2_S}\right)$$

where

- $\sigma^2_{Y_k}$ is the variance of item $k$ and
- $\sigma^2_S$ is the variance of the scale $S$.

$\alpha$: "The expected correlation of two tests that measure the same construct."

# Cronbach's $\alpha$ (continued)

$\alpha$:

- "The expected correlation of two tests that measure the same construct."
- A *lower bound* to reliability.
- If $S = D_1 + D_2 + ... + D_K$, then $\sigma^2_{Y_k} = P_k(1 - P_k)$.
- $\alpha \in [0, 1]$ (theoretically)
- Rule of thumb: $\alpha \geq 0.7$ is "adequate"

Limitations (from Sijtsma 2009):

- Requires equal item variances, equal item covariances, and unidimensionality
- Among the "lower bounds," it's among the smallest
- A better one is the "greatest lower bound" (*glb*), but even it has problems...

# Alpha: A Simulation

```
> # Z is the underlying thing we're trying to measure...
>
> N <- 400
> set.seed(7222009)
> Z <- rt(N,5)
> plot(density(Z))
>
> # Binary indicators... three strongly correlated with Z:
>
> Z1 <- rbinom(N,1,pbinom(Z,1,0.5))
> Z2 <- rbinom(N,1,pbinom(Z,1,0.5))
> Z3 <- rbinom(N,1,pbinom(Z,1,0.5))
>
> # ...two correlated with each other, uncorrelated with Z:
>
> Y <- rt(N,5)
> Y1 <- rbinom(N,1,pbinom(Y,1,0.5))
> Y2 <- rbinom(N,1,pbinom(Y,1,0.5))
>
> # ... and two uncorrelated with anything:
>
> X1 <- rbinom(N,1,0.5)
> X2 <- rbinom(N,1,0.5)
```

# Alpha: A Simulation

```
> simAlpha <- alpha(simDF, check.keys=TRUE)
Warning message:
In alpha(simDF, check.keys = TRUE) :
  Some items were negatively correlated with total scale and were automatically reversed.
 This is indicated by a negative sign for the variable name.
> simAlpha

Reliability analysis
Call: alpha(x = simDF, check.keys = TRUE)

  raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd
      0.54      0.55    0.66      0.15 1.2 0.036 0.46 0.25

 lower alpha upper     95% confidence boundaries
0.47 0.54 0.61

 Reliability if an item is dropped:
    raw_alpha std.alpha G6(smc) average_r  S/N alpha se
Z1       0.39      0.40    0.53      0.10 0.68    0.047
Z2       0.42      0.43    0.55      0.11 0.75    0.045
Z3       0.41      0.42    0.56      0.11 0.73    0.046
Y1-      0.50      0.51    0.58      0.15 1.04    0.038
Y2-      0.49      0.51    0.57      0.15 1.03    0.039
X1-      0.61      0.61    0.72      0.21 1.59    0.031
X2       0.60      0.60    0.71      0.20 1.53    0.032

 Item statistics
      n raw.r std.r  r.cor r.drop mean   sd
Z1  400  0.70  0.71  0.711  0.511 0.33 0.47
Z2  400  0.65  0.66  0.657  0.450 0.34 0.48
Z3  400  0.66  0.68  0.654  0.470 0.31 0.46
Y1- 400  0.52  0.51  0.457  0.270 0.62 0.49
Y2- 400  0.52  0.52  0.468  0.277 0.62 0.49
X1- 400  0.27  0.26 -0.014 -0.019 0.50 0.50
X2  400  0.30  0.29  0.020  0.013 0.51 0.50
```

# Alpha: A Simulation

```
> # Now with only the items that "should" be there:
>
> simAlpha2 <- alpha(simDF[,1:3], check.keys=TRUE)
> simAlpha2

Reliability analysis
Call: alpha(x = simDF[, 1:3], check.keys = TRUE)

  raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd
      0.84      0.84    0.78      0.64 5.4 0.014 0.33 0.41

 lower alpha upper     95% confidence boundaries
0.82 0.84 0.87

 Reliability if an item is dropped:
   raw_alpha std.alpha G6(smc) average_r S/N alpha se
Z1      0.77      0.77    0.63      0.63 3.4    0.023
Z2      0.77      0.77    0.63      0.63 3.4    0.023
Z3      0.80      0.80    0.67      0.67 4.1    0.020

 Item statistics
     n raw.r std.r r.cor r.drop mean   sd
Z1 400  0.88  0.88  0.79   0.72 0.33 0.47
Z2 400  0.88  0.88  0.79   0.72 0.34 0.48
Z3 400  0.86  0.86  0.75   0.69 0.31 0.46
```

# Reliability: SCOTUS Data

```
> SCOTUSAlpha <- alpha(SCOTUS[,2:10],check.keys=TRUE)
> SCOTUSAlpha

Reliability analysis
Call: alpha(x = SCOTUS[, 2:10], check.keys = TRUE)

  raw_alpha std.alpha G6(smc) average_r S/N    ase mean   sd
       0.9       0.9    0.93      0.49 8.6 0.0044 0.45 0.35

 lower alpha upper     95% confidence boundaries
0.89 0.9 0.9

 Reliability if an item is dropped:
          raw_alpha std.alpha G6(smc) average_r S/N alpha se
Rehnquist      0.88      0.88    0.91      0.48 7.4   0.0049
Stevens        0.89      0.89    0.92      0.51 8.4   0.0045
OConnor        0.88      0.88    0.92      0.48 7.3   0.0051
Scalia         0.89      0.89    0.91      0.50 7.9   0.0046
Kennedy        0.88      0.88    0.92      0.48 7.3   0.0051
Souter         0.88      0.88    0.91      0.47 7.2   0.0051
Thomas         0.89      0.89    0.91      0.50 8.0   0.0046
Ginsburg       0.88      0.88    0.91      0.48 7.4   0.0051
Breyer         0.88      0.88    0.91      0.49 7.5   0.0050
```

# Multidimensional Scaling

Types...

- **Metric** - Inputs are interval/ratio-level measures; transformations from $\delta_{ij}$ to $d_{ij}$ are cardinal-valued.
- **Non-Metric** - Inputs are binary or ordinal-level; transformations are rank-preserving *only*.

General steps:

- Generate a dissimilarity matrix $\Delta$ (chosing distance metric)
- Choose the dimensionality $p$
- Choose the type of scaling
- Fit model + interpret the results
- Assess model fit + conduct diagnostics

# MDS: Model Fit / Diagnostics

Kruskal's rule of thumb for stress:

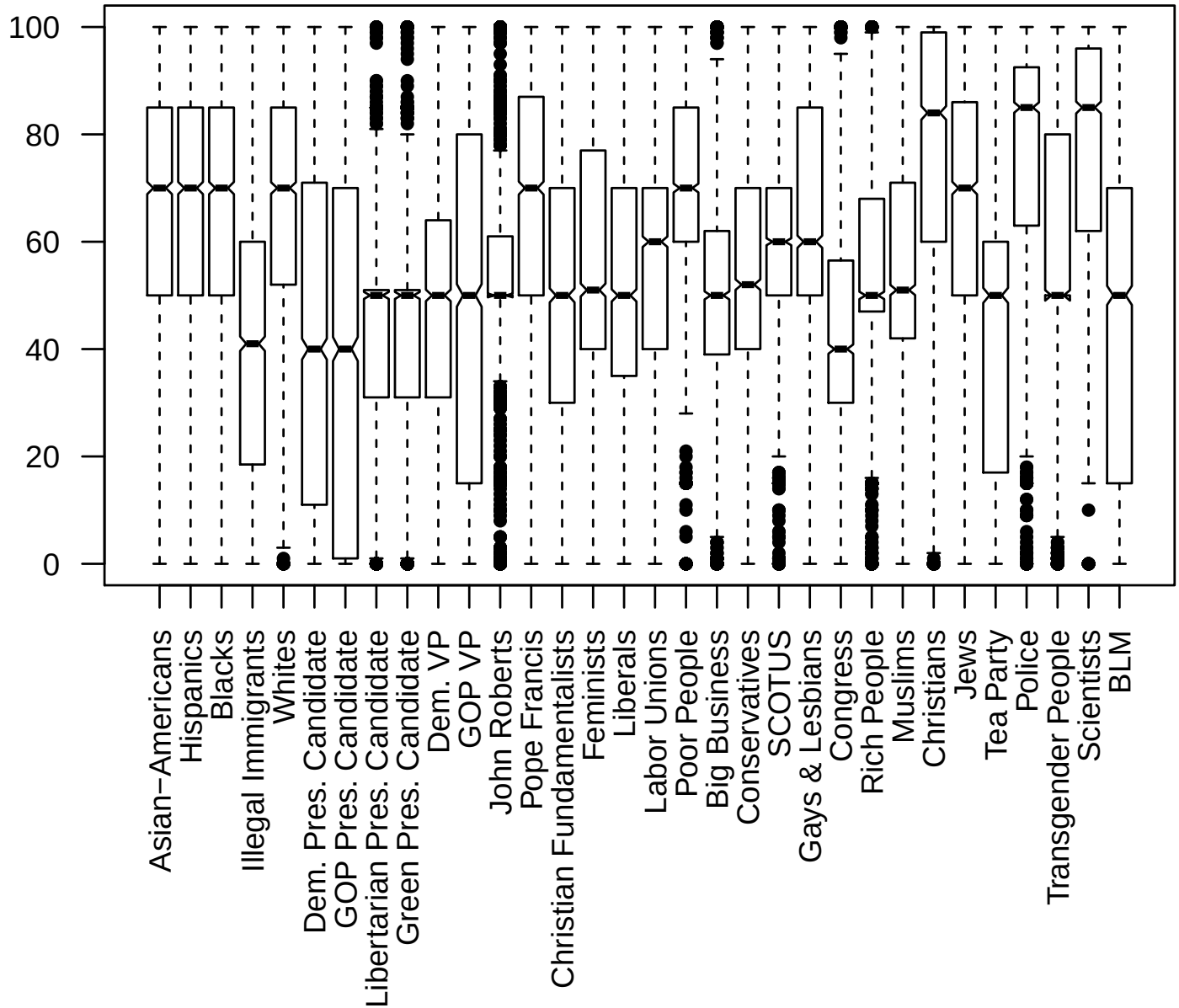
- $\sigma(\mathbf{D}) = 0.20 \rightarrow$ "poor"
- $\sigma(\mathbf{D}) = 0.10 \rightarrow$ "fair"
- $\sigma(\mathbf{D}) = 0.05 \rightarrow$ "good"
- $\sigma(\mathbf{D}) = 0.025 \rightarrow$ "excellent"
- $\sigma(\mathbf{D}) = 0 \rightarrow$ "perfect"

Key diagnostic: **Shepard plot**: a scatterplot of $\delta_{ij}$ vs. $\hat{d}_{ij}$...

- Illustrates model "fit"
- Also illustrates the transformation of the $\delta_{ij}$s

Also **permutation tests** for model fit...

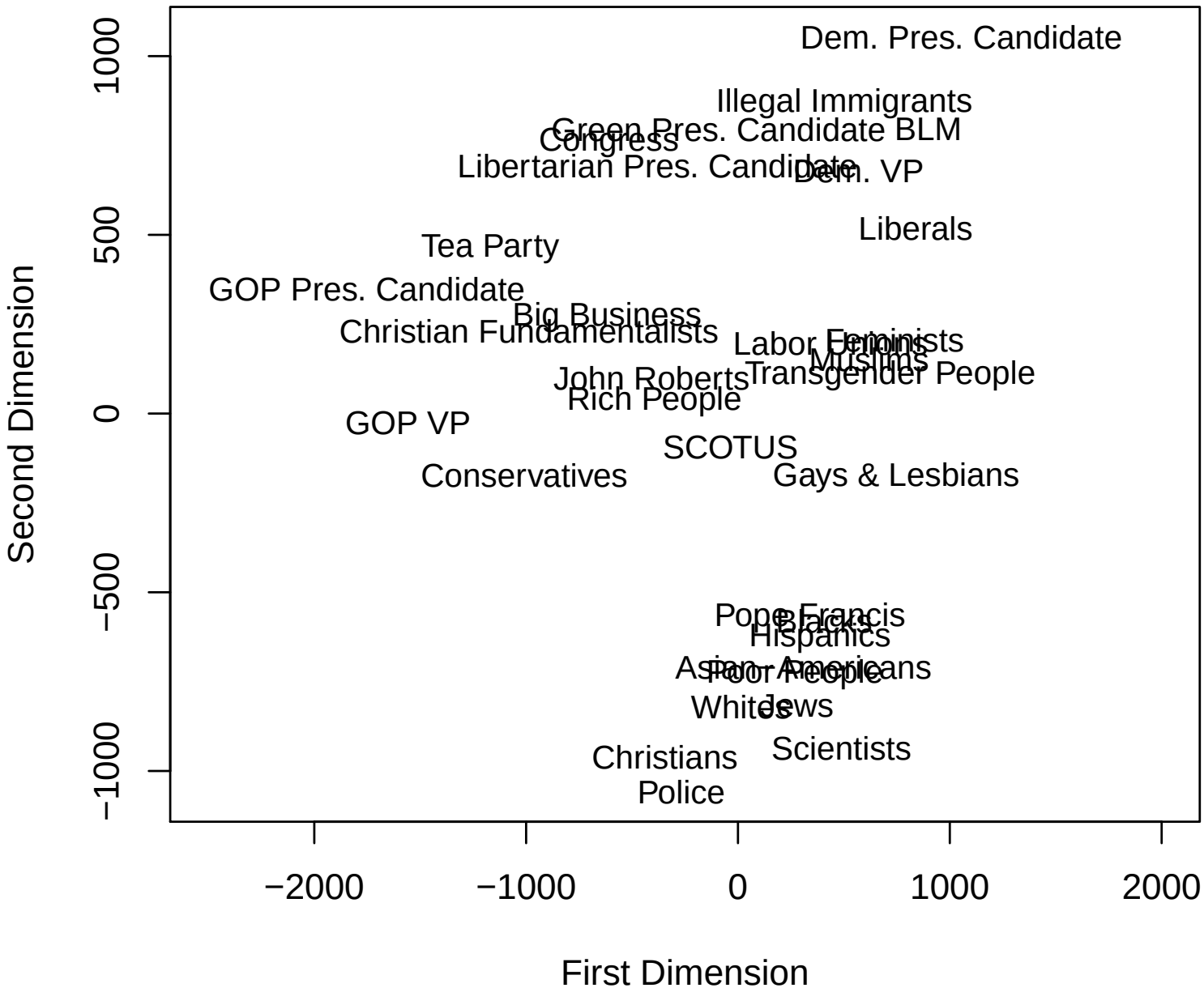
# Example: 2016 ANES Thermometer Scores



# MDS using cmdscale

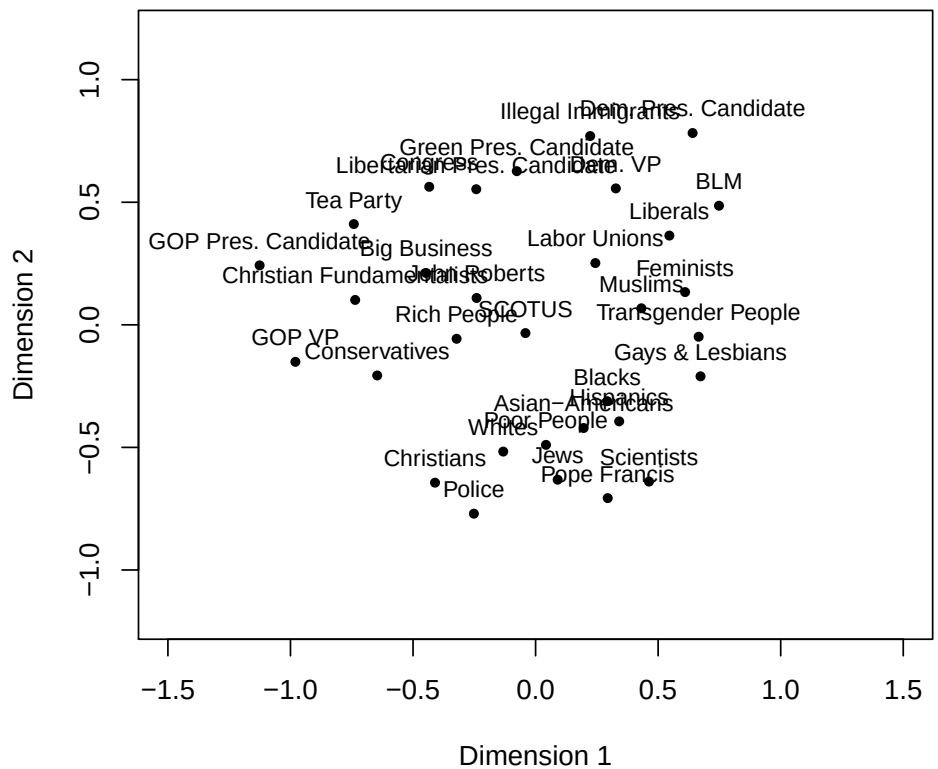
```
> MDS2.alt <- cmdscale(ThermDist,k=2)

> head(MDS2.alt)
                          [,1]   [,2]
Asian-Americans         309.90 -709.9
Hispanics               387.51 -627.6
Blacks                  407.03 -580.7
Illegal Immigrants      501.74  866.3
Whites                   14.72 -821.5
Dem. Pres. Candidate   1054.53 1053.2
```

# MDS Plot, using `cmdscale`

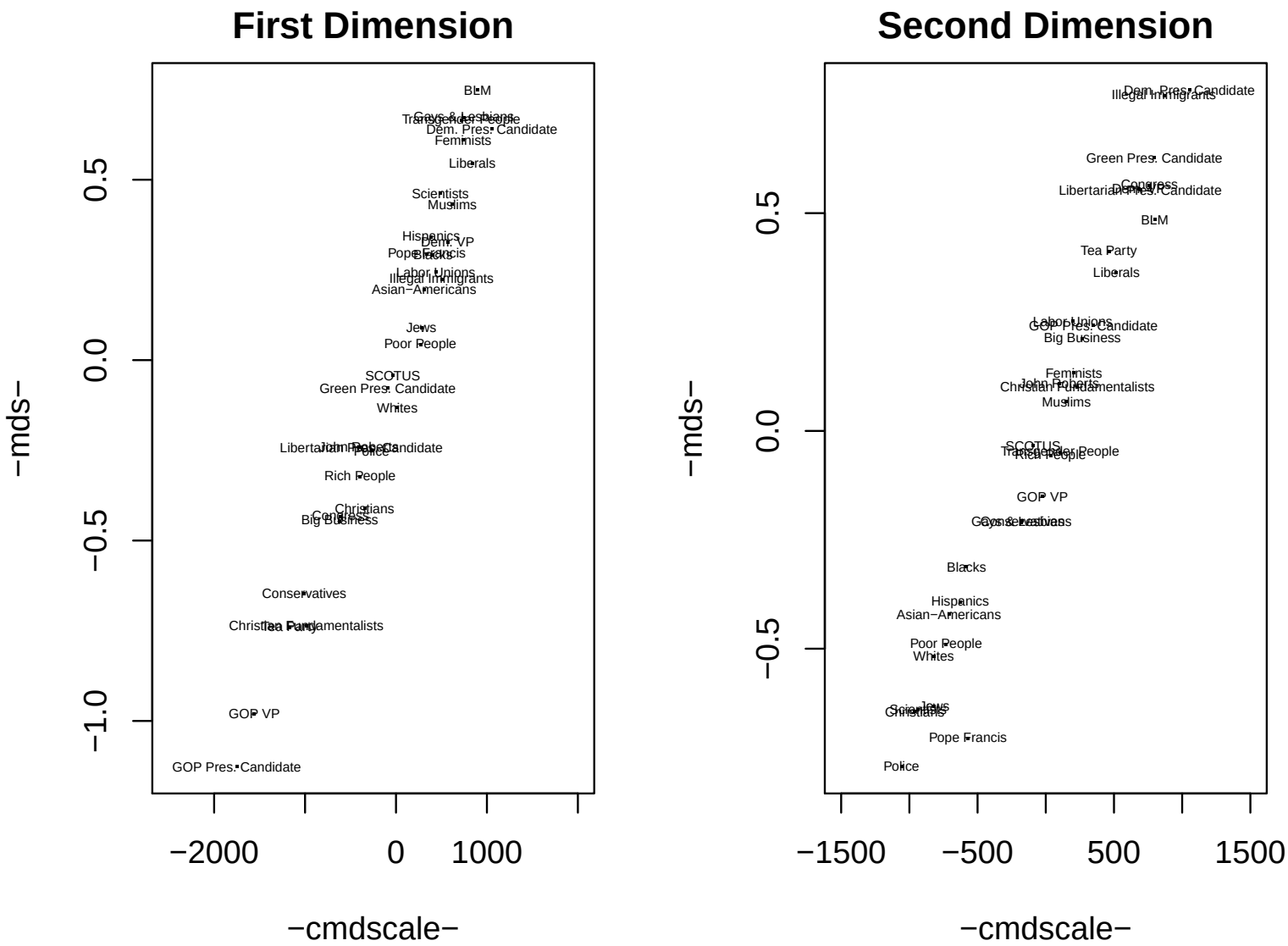


# MDS using mds (ratio scaling)

```
> MDS2 <- mds(ThermDist, ndim=2)
> MDS2

Call:
mds(delta = ThermDist, ndim = 2)

Model: Symmetric SMACOF
Number of objects: 32
Stress-1 value: 0.221
Number of iterations: 87
```



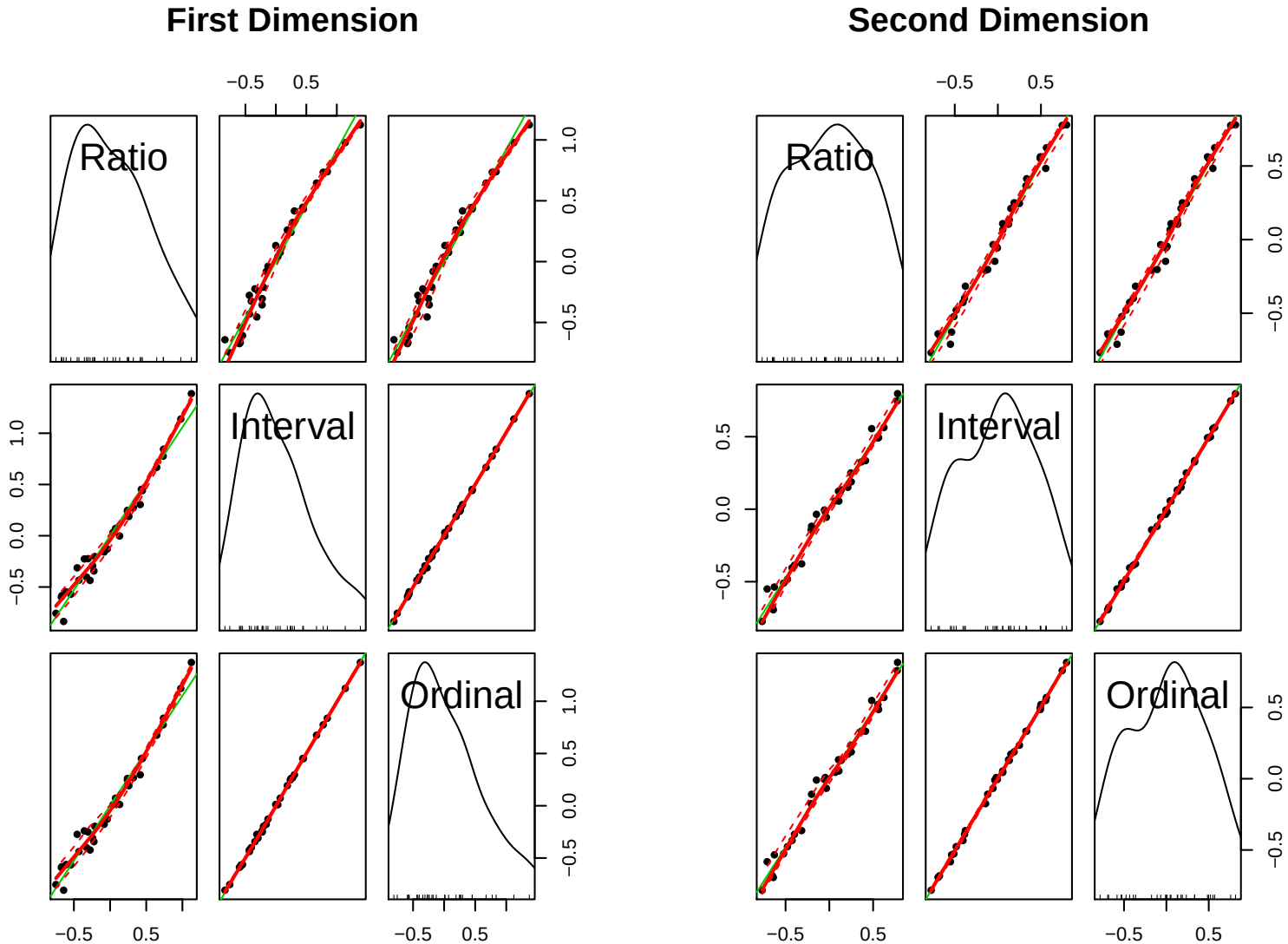
# cmdscale and mds Comparison



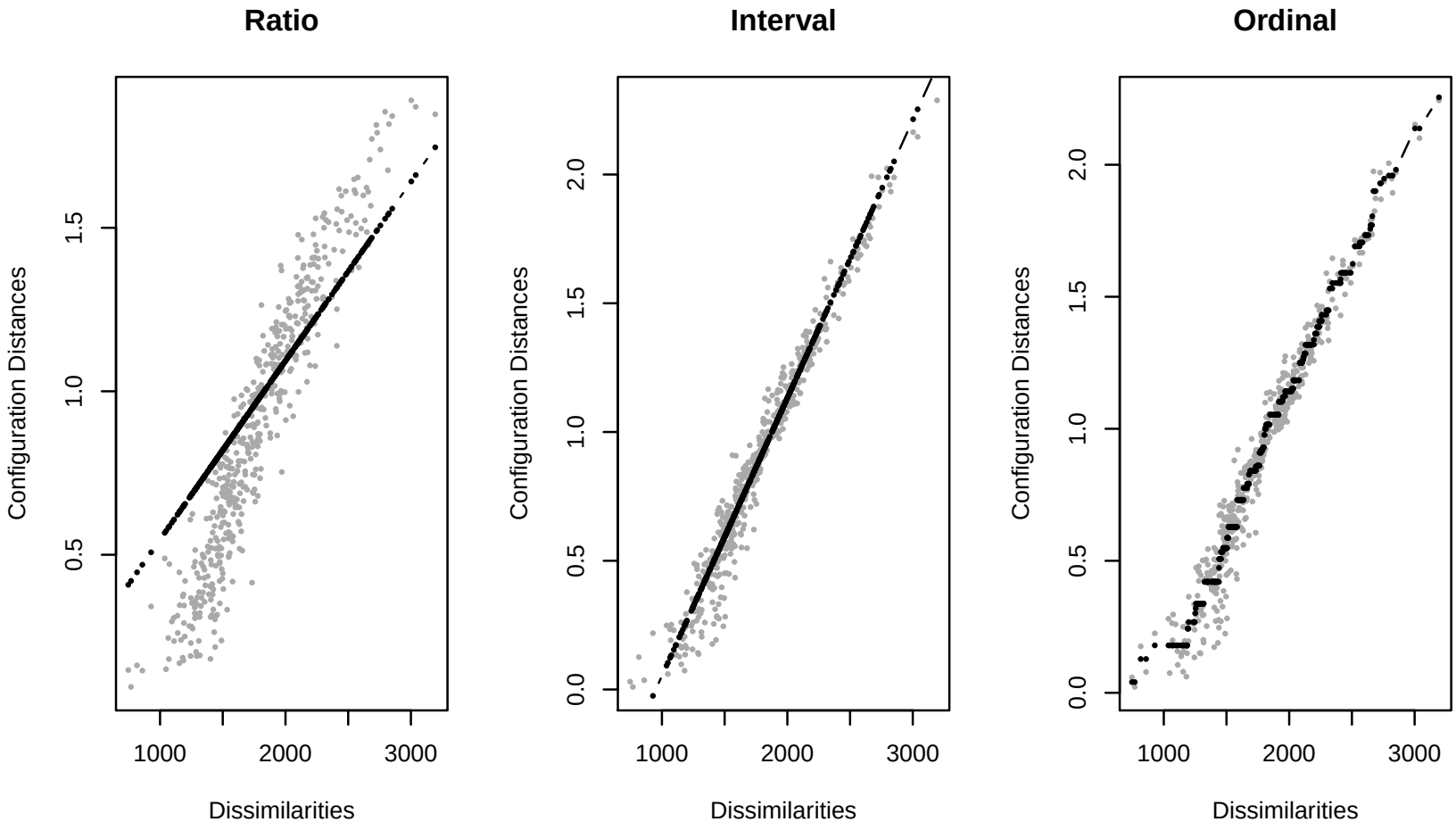
# Alternatives: Interval and Ratio Scaling

```
> MDS2.int <- mds(ThermDist, ndim=2, type="interval")
> MDS2.int

Call:
mds(delta = ThermDist, ndim = 2, type = "interval")

Model: Symmetric SMACOF
Number of objects: 32
Stress-1 value: 0.087
Number of iterations: 17

> # Ordinal:
>
> MDS2.ord <- mds(ThermDist, ndim=2, type="ordinal")
> MDS2.ord

Call:
mds(delta = ThermDist, ndim = 2, type = "ordinal")

Model: Symmetric SMACOF
Number of objects: 32
Stress-1 value: 0.071
Number of iterations: 22
```

# Ratio-Interval-Ordinal Comparison



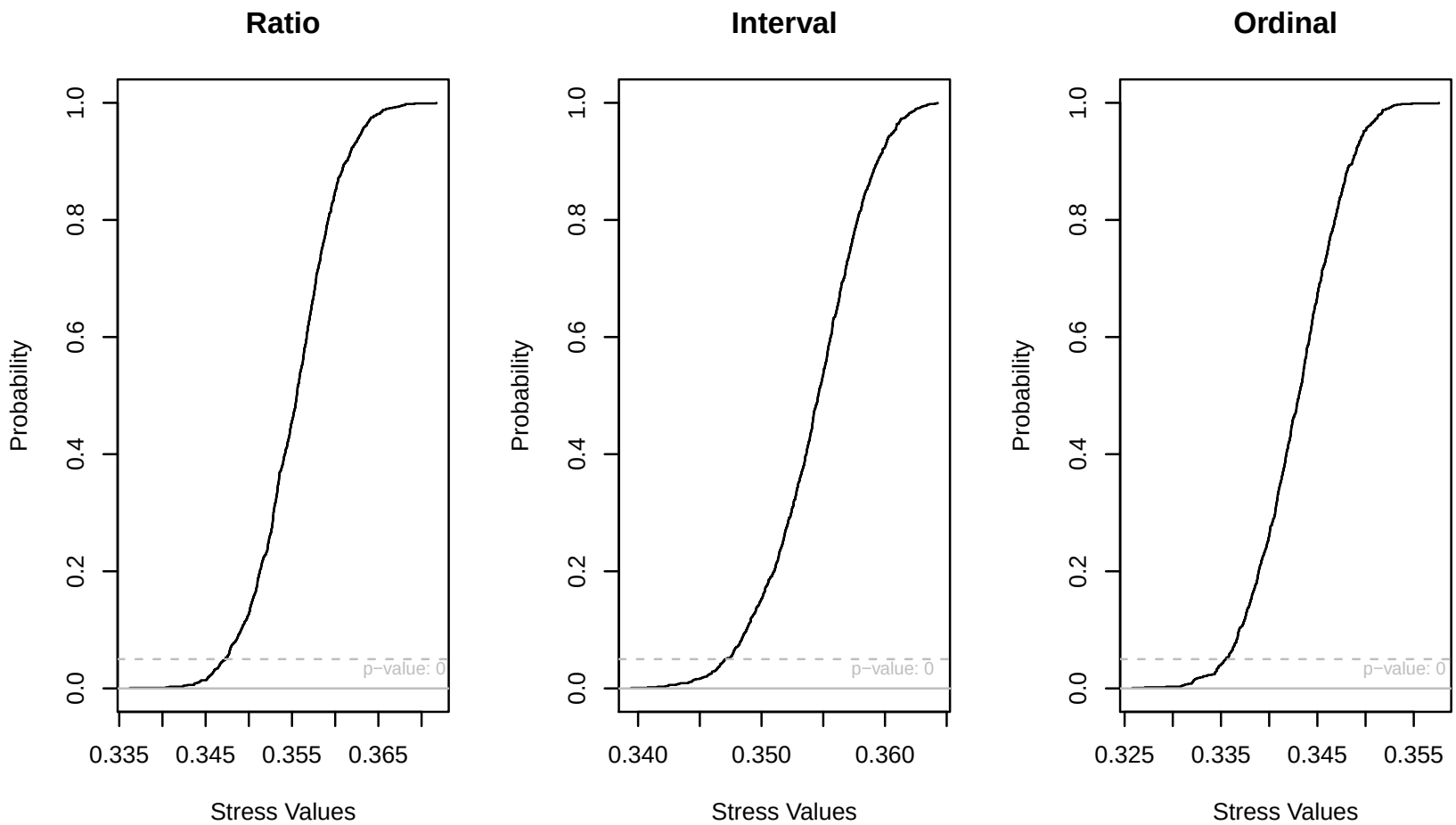
# Diagnostics: Shepard Diagrams



# Diagnostics: Permutation Tests

```
> ThermR.permtest<-permtest(MDS2,nrep=1000,verbose=FALSE)
> ThermR.permtest

Call: permtest.smacof(object = MDS2, nrep = 1000, verbose = FALSE)

SMACOF Permutation Test
Number of objects: 32
Number of replications (permutations): 1000

Observed stress value: 0.221
p-value: 0

> ThermI.permtest<-permtest(MDS2.int,nrep=1000,verbose=FALSE)
> ThermI.permtest

Call: permtest.smacof(object = MDS2.int, nrep = 1000, verbose = FALSE)

SMACOF Permutation Test
Number of objects: 32
Number of replications (permutations): 1000

Observed stress value: 0.087
p-value: 0

> ThermO.permtest<-permtest(MDS2.ord,nrep=1000,verbose=FALSE)
> ThermO.permtest

Call: permtest.smacof(object = MDS2.ord, nrep = 1000, verbose = FALSE)

SMACOF Permutation Test
Number of objects: 32
Number of replications (permutations): 1000

Observed stress value: 0.071
p-value: 0
```

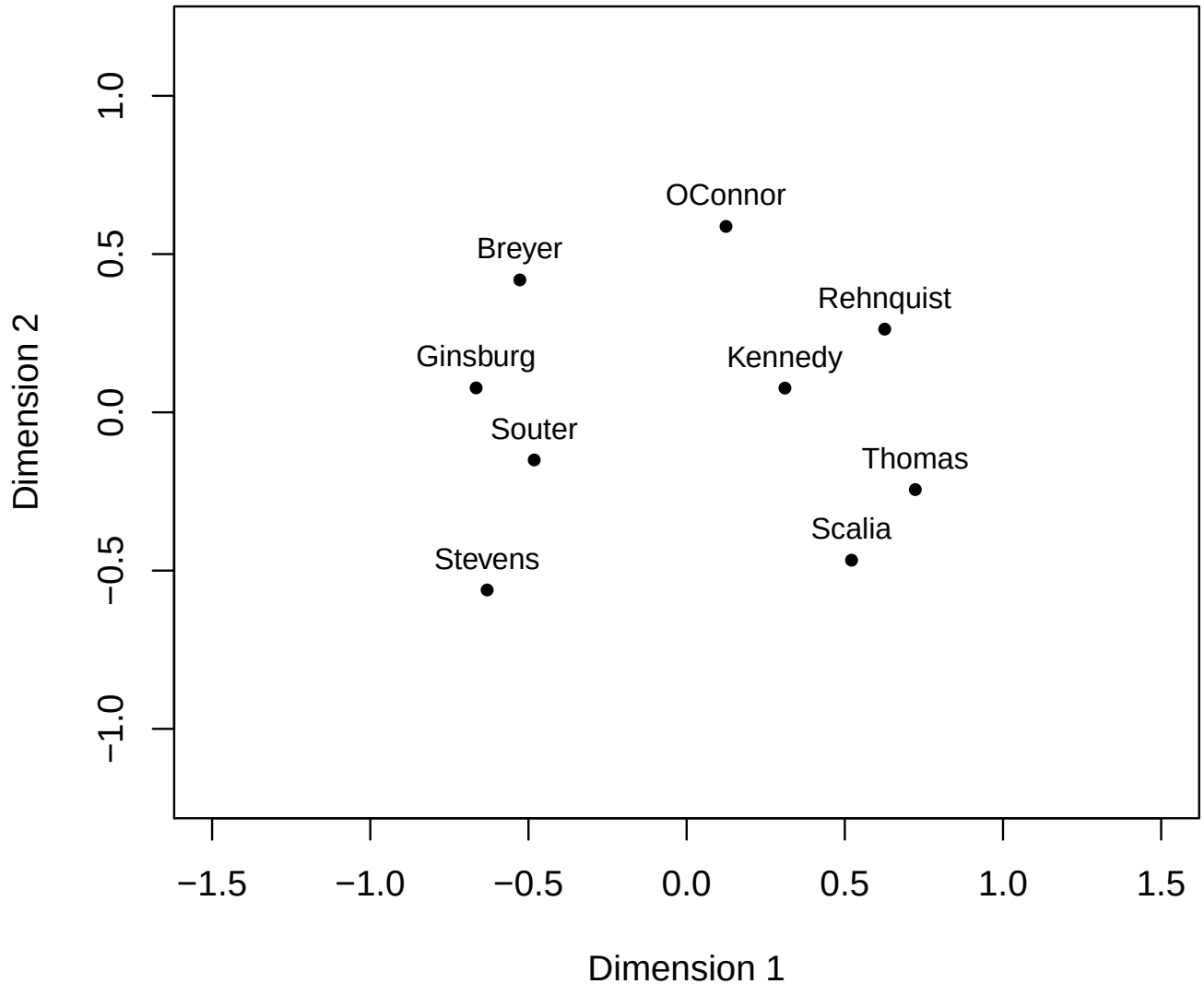
# Permutation Test Plots



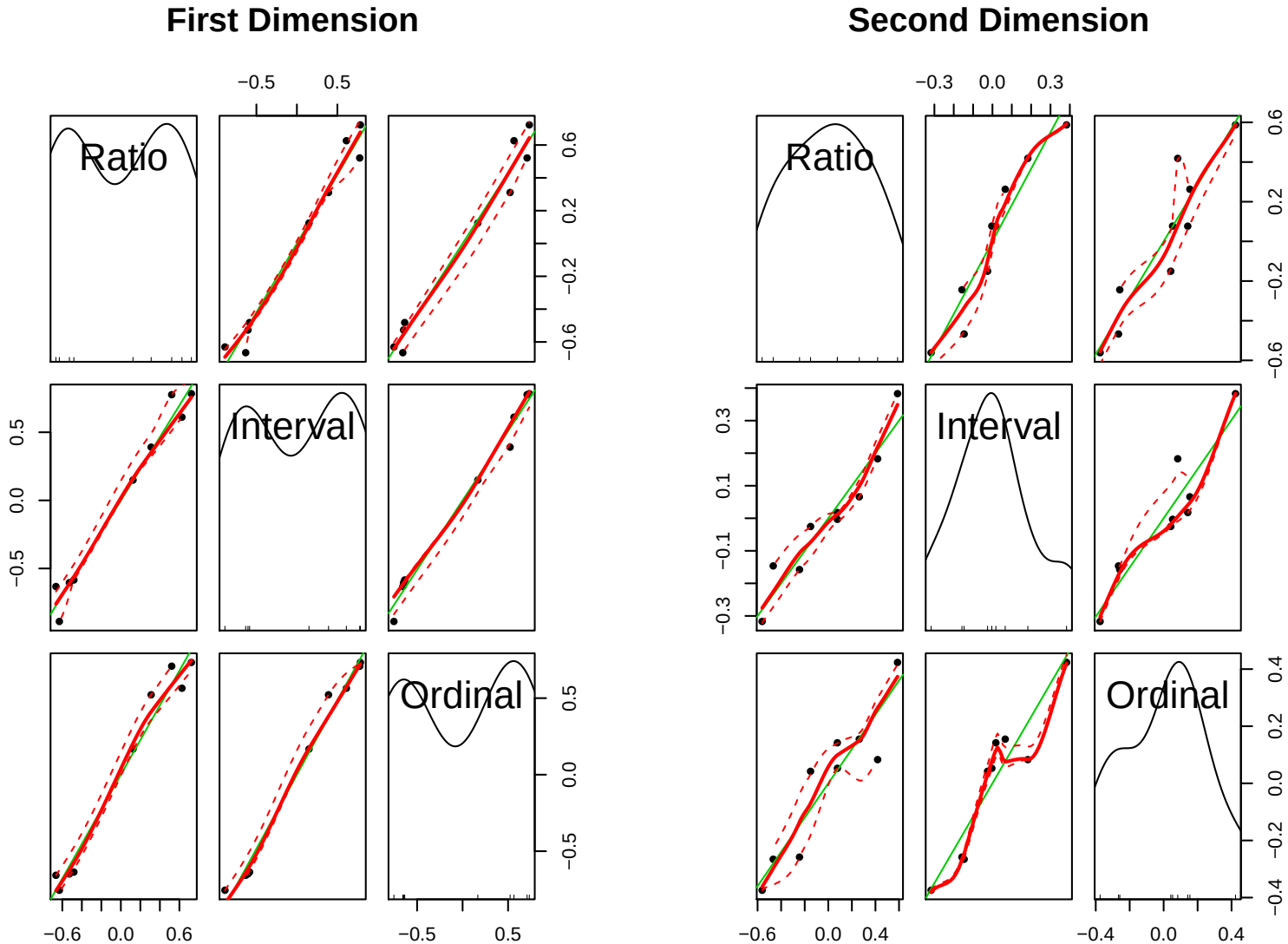
# SCOTUS Redux ($p = 2$)

```
> SCR <- mds(D1SCOTUS, ndim=2, type="ratio")
> SCR

Call:
mds(delta = D1SCOTUS, ndim = 2, type = "ratio")

Model: Symmetric SMACOF
Number of objects: 9
Stress-1 value: 0.184
Number of iterations: 73
```

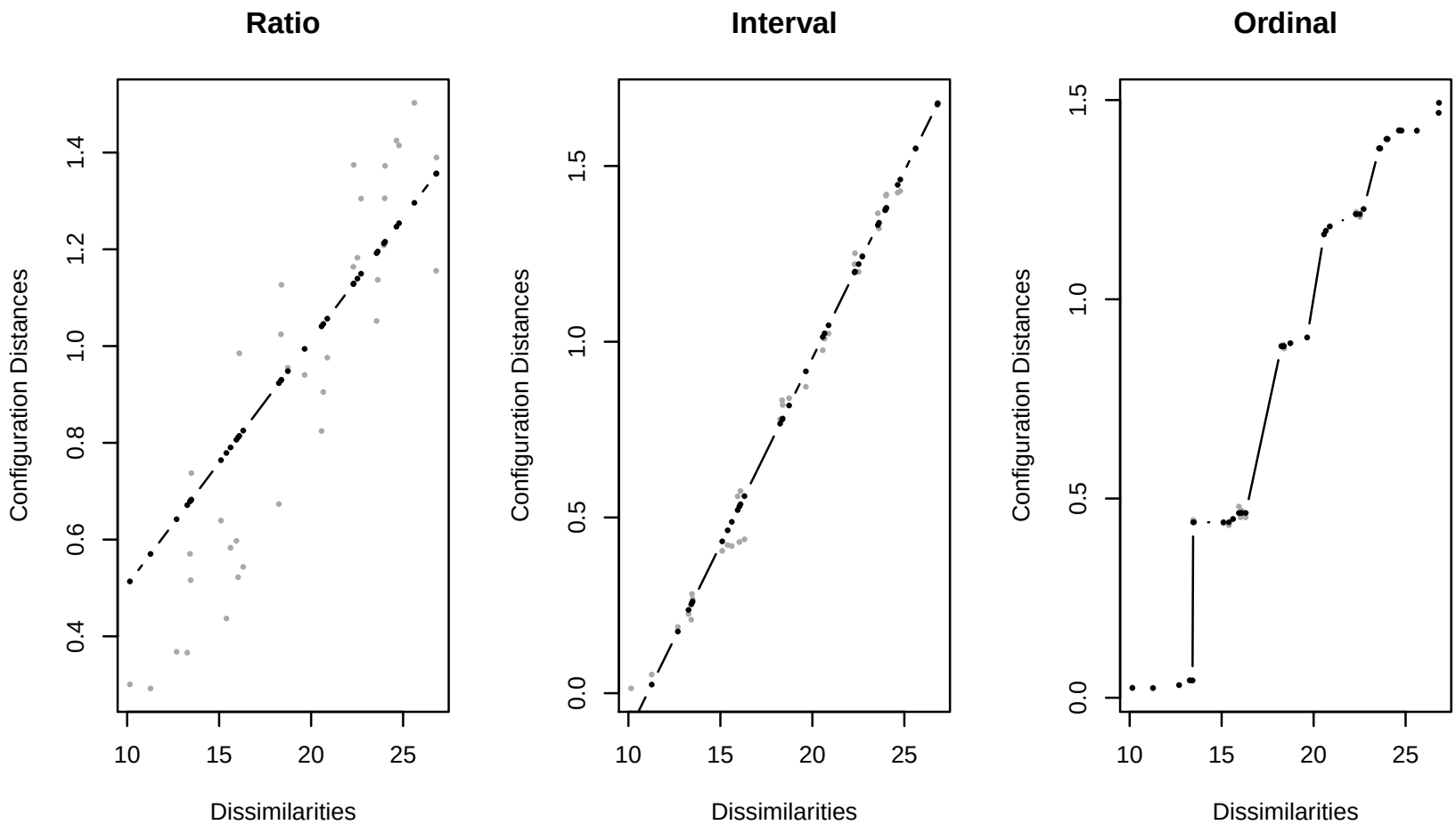
# SCOTUS MDS Plot



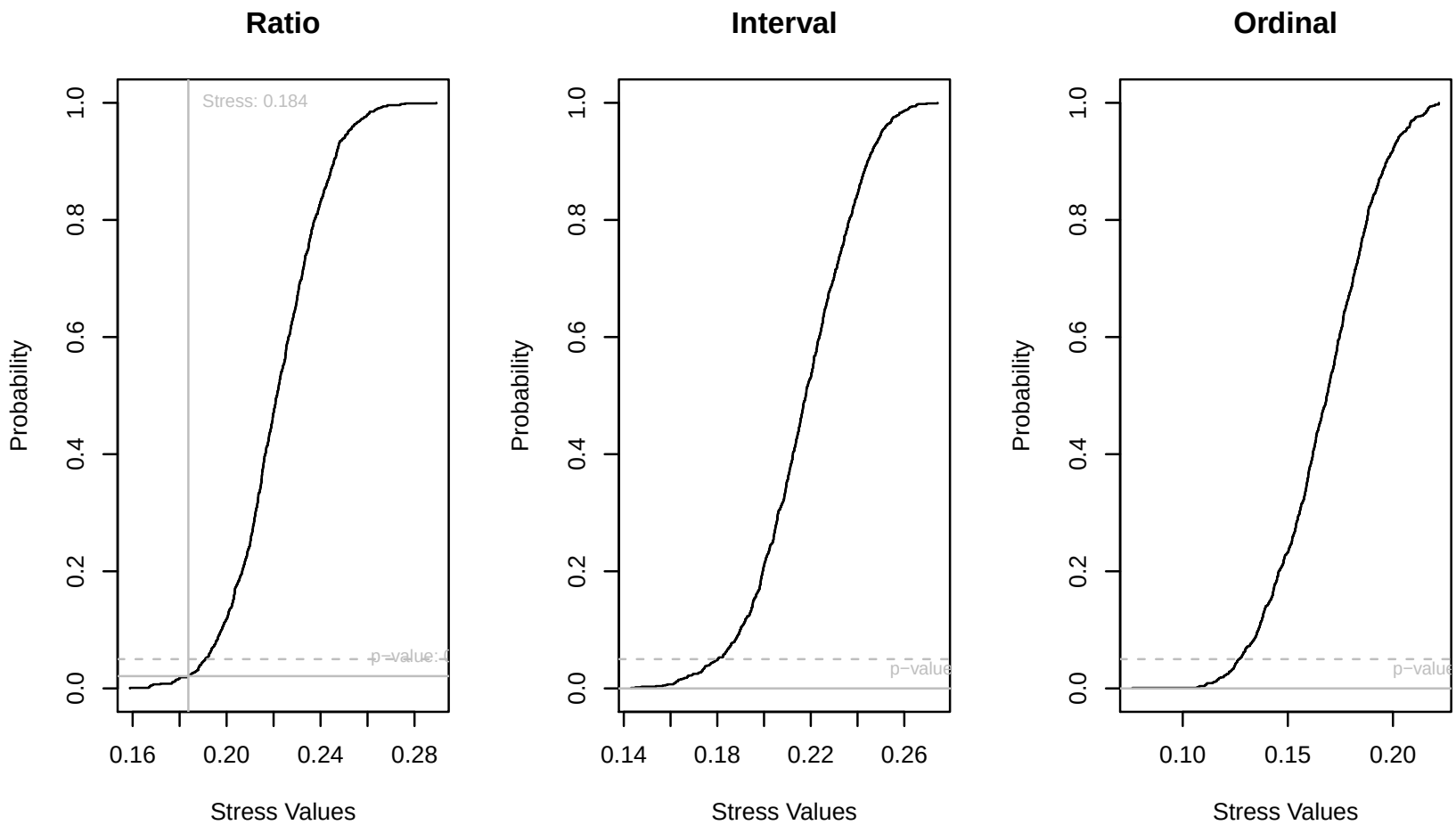
# Ratio-Interval-Ordinal Comparison



# SCOTUS Diagnostics: Shepard Diagrams



# SCOTUS Permutation Test Plots



# Useful References


- Kruskal, J.B., and M. Wish 1978. *Multidimensional Scaling.* Sage.
- McIver, John, and Edward C. and Carmines. 1981. *Unidimensional Scaling*. Sage Publications.
- Davison, Mark L. *Multidimensional Scaling*. 1983. New York: Wiley.
- Cox, Trevor F. and Michael A. A. Cox. 2000. *Multidimensional Scaling*, 2nd Ed. New York: Chapman & Hall.
- Borg, Ingwer, and Groenen, Patrick. 2005. *Modern Multidimensional Scaling: Theory and Applications*, 2nd Ed. Berlin: Springer-Verlag.
- Borg, Ingwer, Patrick Groenen, and Patrick Mair. 2013. *Applied Multidimensional Scaling*. Berlin: Springer-Verlag.

# Useful R Packages and Routines

Distances, Proximities, etc.

- `dist` function (base R)
- `distances` package
- `proxy` package

Scaling

- `stats::cmdscale` (classical MDS, in base R)
- `smacof` (state-of-the-art MDS package)
- `vegan` (ecology package; has some good MDS routines)
- Others...

# Useful Links

- `smacof` documentation.
- Seven ways to do MDS in R.
- Jan De Leeuw's website.