Smacof at 50: A Manual
Part 2: Non-metric Smacof

Jan de Leeuw - University of California Los Angeles

Started March 30 2024, Version of April 07, 2024

Abstract

TBD

Contents

Note: This is a working manuscript which will be expanded/updated frequently. All suggestions for improvement are welcome. All Rmd, tex, html, pdf, R, and C files are in the public domain. Attribution will be appreciated, but is not required. The code files can be found at https://github.com/deleeuw/smacofCode, the manual files at https://github.com/deleeuw/smacofManual, and the example files at https://github.com/deleeuw/smacofExamples.

1 Introduction

pick and rank

2 Loss Function

$$\sigma(X,\Delta_1,\cdots,\Delta_s)=\frac{\sum_{r=1}^s\sum_{i,j}w_{ijr}(\delta_{ijr}-d_{ij}(X))^2}{\sum_{r=1}^s\sum_{i,j}w_{ijr}d_{ij}^2(X))}$$

which must be minimized over X and over $\delta_r\in\mathcal{K}_r$, with $\mathcal{K}_r$ pointed polyhedral convex cones, defined by a partial order $\leq_r$.

Minimize of X for given δ_{ijr}.

$$\sigma_R(X,\Delta_1,\cdots,\Delta_s)=\sum_{r=1}^s\sum_{i,j}w_{ijr}\delta_{ijr}^2-2\sum_{r=1}^s\sum_{i,j}w_{ijr}\delta_{ijr}d_{ij}(X)+\sum_{r=1}^s\sum_{i,j}w_{ijr}d_{ij}^2(X))$$

Nor use the basic smacof inequality

$$d_{ij}(X)\geq\frac{1}{d_{ij}(Y)}\text{tr }X'A_{ij}Y$$

so that

$$\sum_{r=1}^s\sum_{i,j}w_{ijr}\delta_{ijr}d_{ij}(X)\geq\text{tr }X'B(Y)Y$$

$$B(Y):=\sum_{r=1}^s\sum_{i,j}w_{ijr}\frac{\delta_{ijr}}{d_{ij}(Y)}A_{ij}$$

Also

$$V:=\sum_{r=1}^s\sum_{i,j}w_{ijr}A_{ij}$$

So that

$$\sigma_R(X)\leq K-2\text{tr }X'B(Y)Y+\text{tr }X'VX$$

and the smacof update over X with tr $X'VX=1$ is the same as in smacofRR.

3 Paired Comparisons

THe paired comparison method of data collection is the simplest and the most basic one of the Cartwheel methods.

Positive Orthant / Absolute Value / Pairwise

De Leeuw (1970) De Leeuw (2018) Hartmann (1979) Guttman (1969) Johnson (1973)

Suppose datum r says that that $(i,j) < (k,l)$. Then w_{ijr} and w_{klr} are non-zero and all other elements of W_r are zero. Thus

$$w_{ijr}(\delta_{ijr} - d_{ij})^2 + w_{klr}(\delta_{klr} - d_{kl})^2$$

Must be minimized over $\delta_{ijr} \leq \delta_{klr}$. If $d_{ij} \leq d_{kl}$ then $\hat{d}_{ijr} = d_{ij}$ and $\hat{d}_{klr} = d_{kl}$, and otherwise

$$\hat{d}_{ijr} = \hat{d}_{klr} = \frac{w_{ijr}d_{ij} + w_{klr}d_{kl}}{w_{ijr} + w_{klr}}$$

Thus

$$w_{ijr}(\hat{d}_{ijr} - d_{ij})^2 + w_{klr}(\hat{d}_{klr} - d_{kl})^2$$

is zero if the order of d_{ij} and d_{kl} is the same as the order in the data and

$$\frac{w_{ijr}w_{klr}}{w_{ijr} + w_{klr}}(d_{ij} - d_{kl})^2$$

So far we have only considered the forced-choice situation in which the subject has to choose one of the pairs. If we allow for the alternative that (i,j) and (k,l) are equally similar then we can choose between two approaches. In the *primary approach* we incur no loss for this pair, no matter what $d_{ij}(X)$ and $d_{kl}(X)$ are. In the *secondary approach* we require that $\delta_{ijr} = \delta_{klr}$ and consequently we add to the loss if $d_{ij}X) \neq d_{kl}(X)$.

4 Triads

We have implemented three different versions of the method of triads, in which stimuli are presented three at a time, at the corners of an equilateral triangle, as in …

In the first one we present all $\binom{n}{3} = \frac{1}{6}n(n-1)(n-2) \approx \frac{1}{6}n^3$ triples of stimuli and we ask the subject to rank the three similarities between them. More precisely we ask for the two pairs with the largest and smallest similarity, and we interpret the responses as giving us a rank order. Coombs (1954) calls this the *method of similarities*, and Torgerson (1958) calls it the *complete method of triads*.

The second method was first proposed by Richardson (1938), as the *method of triadic cobinations*. Every triad is presented three times using a slightly different layout. … We ask the subject which one of the top stimuli is most similar to the bottom stimulus. This requires $n\binom{n-1}{2} = \frac{1}{2}n(n-1)(n-2) \approx \frac{1}{2}n^3$ presentations for a complete set. Since there is only one comparision involved, this is a special case of the paired comparisons method, in which the pairs always have excetly one stimulus in common. Coombs (1954) call this the *method of propellors* because we only draw lines from the bottom stimulus (the "hub") to the two stimuli at the top.

5 Richardson Hub Method

each triple presented three times, with a different hub each time which one of the two is maximally similar to the hub stimulus. Thus the data is the single inequality $(i, k) < (j, k)$. Coombs calls this the …

6 Conditional Rank Orders – Klingberg

7 Full Rank Orders

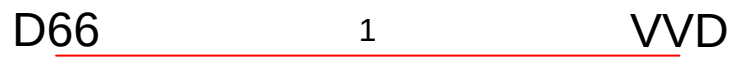
8 A Paired Comparison Example

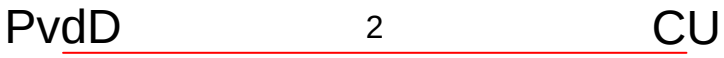
The objects that we want to scale are 10 Dutch political parties.

```
parties
```

```
##  [1] "GL"   "PvdA" "VVD"  "D66"  "CDA"  "SP"   "PvdD" "CU"   "FVD"  "SGP"
```

There is only one subject, and it is me. I used the program `smacofMakeRandomPairs()` to generate and present to me 50 random pairs of pairs (i, j) and (k, l). One such pair looks like






```
## (D66,VVD) and (PvdD,CU)
## most similar pair:
```

In Rstudio the graphics are in the plot window, the text is in the console. If you run R from the terminal the text will be in the terminal, and the graphics will be in the R graphics device for the session.

It took me about 5 minutes to make the 50 binary choices, duly recorded by the program. I left The Netherlands almost 40 years ago, so I am far from an expert on Dutch politics, so my choices may be far off the mark.

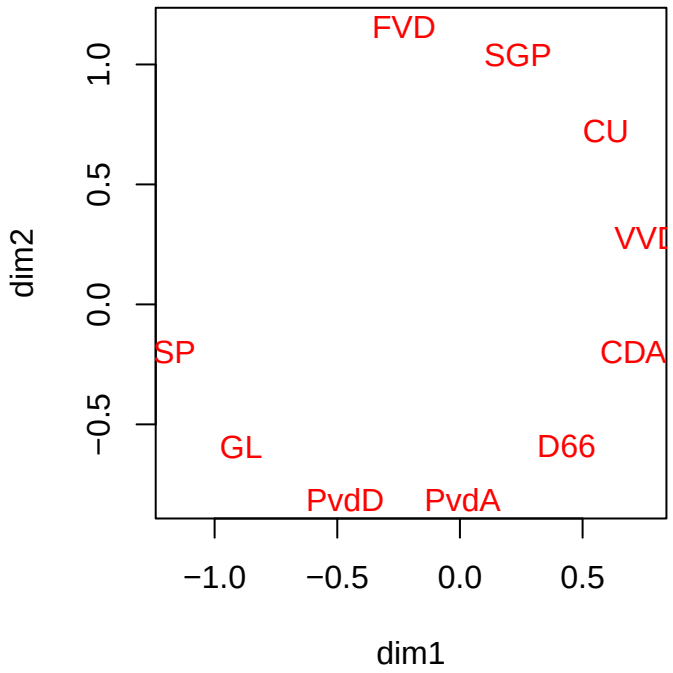
The data, with the four indices first and then the choice, are

```
##     i  j  k  l
## 1   8  5  2  9 1
## 2   4  1  1  2 2
## 3   1  3  2  7 2
## 4   5  2  2  7 2
## 5   2  9  9  8 2
## 6   7  5  9 10 1
## 7   1 10 10  8 2
## 8   9 10  9  4 2
## 9   5  2  5  9 2
## 10  4  1 10  7 1
## 11  1  3  5 10 2
```

```
## 12  4 10  1 10 1
## 13  5  4  8  5 2
## 14  5  1  4  2 2
## 15  4 10  5  2 2
## 16  1  3  9 10 2
## 17  4  7  4  3 2
## 18  4  1 10  7 1
## 19  6  4  5  4 2
## 20  6  1  9  3 2
## 21  8  3  4  3 1
## 22 10  7  6  1 2
## 23  4  1  9  8 1
## 24 10  7  2  6 2
## 25  8  6  5  4 2
## 26  8  7  4  1 2
## 27  6  7  1  2 2
## 28  9  7  4  3 2
## 29  4  8  6  4 1
## 30  2  7  9  7 1
## 31  2  8  9  4 1
## 32  4  7  6  4 1
## 33  5  9  9 10 1
## 34  1  8  4  2 2
## 35  1  3  9  7 1
## 36 10  8  3  6 1
## 37  1  3  6  7 1
## 38  1 10  9  8 2
## 39  1  2 10  2 1
## 40  2  9  9  8 2
## 41  6 10  9  4 2
## 42 10  2  6  5 2
## 43  1  7  7  5 1
## 44  3  5 10  8 1
## 45  5  9  2  8 1
## 46  9  8  9 10 1
## 47  9  8  6  4 1
## 48  9  4  6  1 1
## 49  2  8  4  7 2
## 50  4  1  9  4 1
```

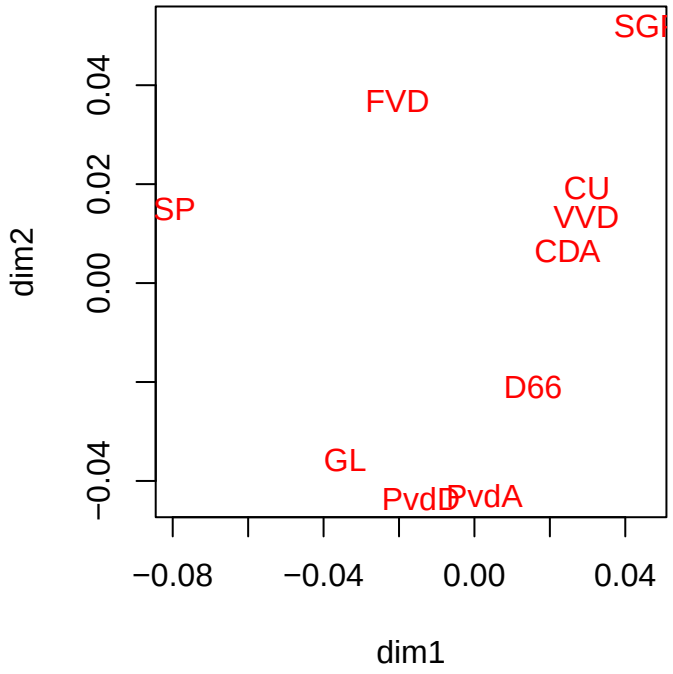
Thus in the first row I say that the pair $(8, 5)$ is more similar than $(2, 9)$, in other words (CU, CDA) is more similar than (PvdA, FVD).

In order to start the iterative process we need an initial configuration. Insipred by De Gruijter (1967) I used the infamous left-right horseshoe.




```
source("../../smacofCode/smacofNM/smacofNMforPairs.R")
partiesResult <- smacofNMforPairs(partiesData, partiesXold,
                                  eps = 1e-15, itmax = 1000, verbose = FALSE)
```

We have nice smooth monotone convergence. The convergence criterion is reached in 344 iterations and the stress is $1.3573576 \times 10^{-14}$, i.e. practically zero. Thus we have actually found a solution to the system of 50 nonlinear inequalities defined by the data. This means two things. In the first place I am consistent in my choices, and in the second place the solution is undoubtedly far from unique. The map is



We are still reasonably close to the horseshoe, showing the influence of the initial configuration. Of course the main reason for the non-uniqueness is not the consistency of my choices, but the fact that we have used only 50 pairs from the `choose(choose(10, 2), 2)` = 990 possible ones. The solution will become much tighter if there are more pairs, either from a single subject, or from a number of subjects (in which case we would certainly prefer MDS with parameters for individual differences).

9 A Rank Order Example

```
library(MASS)

source("../../smacofCode/smacofNM/smacofConvert.R")
source("../../smacofCode/smacofNM/smacofMakeData.R")
source("../../smacofCode/smacofNM/smacofMonotoneRegression.R")
source("../../smacofCode/smacofNM/smacofPlots.R")

data(ekman, package = "smacof")
ekman <- 1 - ekman
ekmanData <- smacofMakeRankOrderData(ekman)
ekmanXold <- matrix(c(-0.1576320, -0.54429773,
                      -0.2169837, -0.50815312,
                      -0.4778581, -0.31203061,
                      -0.5131424, -0.23393512,
                      -0.5338027,  0.09175252,
                      -0.4368053,  0.36071649,
                      -0.2650675,  0.52174944,
                      -0.1374662, 0.57443857,
                      0.3162169,  0.46015778,
                      0.4575480,  0.24454905,
                      0.5210238,  0.03357499,
                      0.5128727, -0.13074190,
                      0.4824539, -0.22250509,
                      0.4486425, -0.33527525), 14, 2, byrow = TRUE)
```

10 Code

10.1 Make the cartwheels and ask for choices

```
smacofMakeAllTriads <- function(names, complete = TRUE) {
  outfile <- file("./output.txt", open = "w")
  n <- length(names)
  m <- choose(n, 3)
  z <- t(combn(n, 3))[sample(1:m, m), ]
  z <- t(apply(z, 1, function(x)
    sample(x, length(x))))
  y <- 8 - 3 * sqrt(3)
  for (i in 1:m) {
    x <- z[i, ]
    plot(
      1:10,
      axes = FALSE,
      type = "n",
      xlab = "",
      ylab = ""
    )
    lines(c(2, 8), c(8, 8), col = "RED")
    lines(c(2, 5), c(8, y), col = "RED")
    lines(c(8, 5), c(8, y), col = "RED")
    text(c(2, 8, 5), c(8.2, 8.2, y - .2),
         c(names[x[1]], names[x[2]], names[x[3]]), cex = 1.5)
    text(5, 8.5, "1")
    text(3, 5.40, "2")
    text(7, 5.40, "3")
    print(c(noquote(names[x[1]]), noquote(names[x[2]]), noquote(names[x[3]])))
    u <- readline("most similar pair: ")
    if (complete) {
      v <- readline("least similar pair: ")
      write(c(x, u, v), ncolumns = 5, file = outfile)
    } else {
      write(c(x, u), ncolumns = 4, file = outfile)
    }
  }
  close(outfile)
}

smacofMakeRandomTriads <-
  function(names, nrandom, complete = TRUE) {
    outfile <- file("./output.txt", open = "w")
```

```
    n <- length(names)
    y <- 8 - 3 * sqrt(3)
    for (i in 1:nrandom) {
      x <- sample(1:n, 3)
      plot(
        1:10,
        axes = FALSE,
        type = "n",
        xlab = "",
        ylab = ""
      )
      lines(c(2, 8), c(8, 8), col = "RED")
      lines(c(2, 5), c(8, y), col = "RED")
      lines(c(8, 5), c(8, y), col = "RED")
      text(c(2, 8, 5), c(8.2, 8.2, y - .2),
           c(names[x[1]], names[x[2]], names[x[3]]), cex = 1.5)
      text(5, 8.5, "1")
      text(3, 5.40, "2")
      text(7, 5.40, "3")
      print(c(noquote(names[x[1]]), noquote(names[x[2]]), noquote(names[x[3]])))
      u <- readline("most similar pair: ")
      if (complete) {
        v <- readline("least similar pair: ")
        write(c(x, u, v), ncolumns = 5, file = outfile)
      } else {
        write(c(x, u), ncolumns = 4, file = outfile)
      }
    }
    close(outfile)
  }

smacofMakeAllPairs <- function(names) {
  outfile <- file("./output.txt", open = "w")
  n <- length(names)
  l <- choose(n, 2)
  u <- combn(n, 2)
  u <- apply(u, 2, function(x)
    sample(x, length(x)))
  m <- choose(l, 2)
  v <- combn(l, 2)[, sample(1:m, m)]
  v <- apply(v, 2, function(x)
    sample(x, length(x)))
  for (i in 1:m) {
    x <- c(u[, v[1, i]], u[, v[2, i]])
```

```
    plot(
      1:10,
      axes = FALSE,
      type = "n",
      xlab = "",
      ylab = ""
    )
    lines(c(2, 8), c(8, 8), col = "RED")
    lines(c(2, 8), c(4, 4), col = "RED")
    text(c(2, 8, 2, 8),
         c(8.5, 8.5, 4.5, 4.5),
         c(names[x[1]], names[x[2]], names[x[3]], names[x[4]]),
         cex = 1.5)
    text(5, 8.5, "1")
    text(5, 4.5, "2")
    cat("(",
        names[x[1]],
        ",",
        names[x[2]],
        ") and (",
        names[x[3]],
        ",",
        names[x[4]],
        ")\n",
        sep = "")
    r <- readline("most similar pair: ")
    write(c(x, r), ncolumns = 5, file = outfile)
  }
  close(outfile)
}

smacofMakeRandomPairs <- function(names, nrandom) {
  outfile <- file("./output.txt", open = "w")
  n <- length(names)
  l <- choose(n, 2)
  u <- combn(n, 2)
  u <- apply(u, 2, function(x)
    sample(x, length(x)))
  for (i in 1:nrandom) {
    k <- sample(l, 2)
    x <- c(u[, k[1]], u[, k[2]])
    plot(
      1:10,
      axes = FALSE,
```

```
      type = "n",
      xlab = "",
      ylab = ""
    )
    lines(c(2, 8), c(8, 8), col = "RED")
    lines(c(2, 8), c(4, 4), col = "RED")
    text(c(2, 8, 2, 8),
         c(8.5, 8.5, 4.5, 4.5),
         c(names[x[1]], names[x[2]], names[x[3]], names[x[4]]),
         cex = 1.5)
    text(5, 8.5, "1")
    text(5, 4.5, "2")
    cat("(",
        names[x[1]],
        ",",
        names[x[2]],
        ") and (",
        names[x[3]],
        ",",
        names[x[4]],
        ")\n",
        sep = "")
    r <- readline("most similar pair: ")
    write(c(x, r), ncolumns = 5, file = outfile)
  }
  close(outfile)
}

smacofMakeRankOrderData <- function(delta, tieblocks = TRUE) {
  if (any(class(delta) == "dist")) {
    n <- attr(delta, "Size")
    delta <- smacofDistToRMVector(delta)
  }
  if (is.matrix(delta)) {
    delta <- as.dist(delta)
    n <- attr(delta, "Size")
    delta <- smacofDistToRMVector(delta)
  }
  delta <- rank(delta)
  x <- matrix(0, 0, 3)
  k <- 1
  for (i in 2:n) {
    for (j in 1:(i - 1)) {
      x <- rbind(x, c(i, j, delta[k]))
```

```
      k <- k + 1
    }
  }
  r <- order(delta)
  x <- x[r, ]
  return(x)
}

smacofMakeConditionalRankOrderData <-
  function(delta, nr, nc, tieblocks = TRUE) {
    x <- matrix(0, 0, 3)
    for (i in 1:nr) {
      if (is.matrix(delta)) {
        d <- delta[i, ]
      } else {
        d <- delta[(i - 1) * nc + (1:nc)]
      }
      d <- rank(d)
      u <- order(d)
      v <- unname(cbind(i, 1:nc, d)[u,])
      x <- rbind(x, v)
    }
    return(x)
  }
```

10.2 Smacof for Pairs

```
smacofNMforPairs <-
  function(data,
           xold,
           itmax = 10,
           eps = 1e-10,
           verbose = TRUE) {
    n <- nrow(xold)
    m <- nrow(data)
    itel <- 1
    dold <- as.matrix(dist(xold))
    w <- matrix(0, n, n)
    for (r in 1:m) {
      i <- data[r, 1]
      j <- data[r, 2]
      k <- data[r, 3]
      l <- data[r, 4]
```

```
    w[i, j] <- w[i, j] + 1
    w[k, l] <- w[k, l] + 1
    w[j, i] <- w[i, j]
    w[l, k] <- w[k, l]
  }
  ssqd <- sum(w * (dold ^ 2))
  dold <- dold / sqrt(ssqd)
  xold <- xold / sqrt(ssqd)
  v <- -w
  diag(v) <- -rowSums(v)
  vinv <- ginv(v)
  sold <- Inf
  repeat {
    bold <- matrix(0, n, n)
    snew <- 0
    for (r in 1:m) {
      i <- data[r, 1]
      j <- data[r, 2]
      k <- data[r, 3]
      l <- data[r, 4]
      x <- data[r, 5]
      dij <- dold[i, j]
      dkl <- dold[k, l]
      if (x == 1) {
        if (dij <= dkl) {
          dhatij <- dij
          dhatkl <- dkl
        } else {
          ave <- (dij + dkl) / 2
          dhatij <- ave
          dhatkl <- ave
          snew <- snew + ((dij - dkl) ^ 2) / 2
        }
      }
      if (x == 2) {
        if (dkl <= dij) {
          dhatij <- dij
          dhatkl <- dkl
        } else {
          ave <- (dij + dkl) / 2
          dhatij <- ave
          dhatkl <- ave
          snew <- snew + ((dij - dkl) ^ 2) / 2
        }
```

```
      }
      bold[i, j] <- bold[i, j] + dhatij / dij
      bold[k, l] <- bold[k, l] + dhatkl / dkl
      bold[j, i] <- bold[i, j]
      bold[l, k] <- bold[k, l]
    }
    bold <- -bold
    diag(bold) <- -rowSums(bold)
    xnew <- vinv %*% bold %*% xold
    dnew <- as.matrix(dist(xnew))
    ssqd <- sum(w * (dnew ^ 2))
    xnew <- xnew / sqrt(ssqd)
    dnew <- dnew / sqrt(ssqd)
    if (verbose) {
      cat(
        "itel = ",
        formatC(itel, format = "d"),
        "sold = ",
        formatC(sold, digits = 10, format = "f"),
        "snew = ",
        formatC(snew, digits = 10, format = "f"),
        "\n"
      )
    }
    if ((itel == itmax) || ((sold - snew) < eps)) {
      break
    }
    xold <- xnew
    dold <- dnew
    sold <- snew
    itel <- itel + 1
  }
  return(list(
    b = bold,
    v = v,
    x = xnew,
    loss = snew,
    itel = itel
  ))
}
```

10.3 Smacof for Rank Orders

```
library(MASS)


smacofNMforRankOrder <-
  function(data,
           xold,
           ties = 1,
           itmax = 1000,
           eps = 1e-10,
           verbose = TRUE) {
    n <- nrow(xold)
    m <- nrow(data)
    itel <- 1
    # put the w in a matrix
    w <- matrix(0, n, n)
    wvec <- data[, 4]
    for (i in 1:m) {
      w[data[i, 1], data[i, 2]] <- wvec[i]
      w[data[i, 2], data[i, 1]] <- wvec[i]
    }
    v <- -w
    diag(v) <- -rowSums(v)
    vinv <- ginv(v)
    dold <- as.matrix(dist(xold))
    ssqd <- sum(w * (dold ^ 2))
    dold <- dold / sqrt(ssqd)
    xold <- xold / sqrt(ssqd)
    # put dold in the correct order in a vector
    dord <- rep(0, m)
    for (i in 1:m) {
      dord[i] <- dold[data[i, 1], data[i, 2]]
    }
    if (ties == 1) {
      dprim <- smacofPrimaryMonotoneRegression(data, dord)
      dord <- dprim$result
      data <- dprim$data
    }
    if (ties == 2) {
      dord <- smacofSecondaryMonotoneRegression(data, dord)
    }
    # put the dhat in a matrix
    dhat <- matrix(0, n, n)
```

```
  for (i in 1:m) {
    dhat[data[i, 1], data[i, 2]] <- dord[i]
    dhat[data[i, 2], data[i, 1]] <- dord[i]
  }
  sold <- sum(w * (dhat - dold) ^ 2)
  repeat {
    bold <- dhat / (dold + diag(n))
    bold <- -bold
    diag(bold) <- -rowSums(bold)
    xnew <- vinv %*% bold %*% xold
    dnew <- as.matrix(dist(xnew))
    ssqd <- sum(w * (dnew ^ 2))
    xnew <- xnew / sqrt(ssqd)
    dnew <- dnew / sqrt(ssqd)
    smid <- sum(w * (dhat - dnew) ^ 2)
    # put dold in the correct order in a vector
    dord <- rep(0, m)
    for (i in 1:m) {
      dord[i] <- dold[data[i, 1], data[i, 2]]
    }
    if (ties == 1) {
      dpri <- smacofPrimaryMonotoneRegression(data, dord)
      dord <- dpri$result
      data <- dpri$data
    }
    if (ties == 2) {
      dord <- smacofSecondaryMonotoneRegression(data, dord)
    }
    # put the dhat in a matrix
    # put the dhat in a matrix
    dhat <- matrix(0, n, n)
    for (i in 1:m) {
      dhat[data[i, 1], data[i, 2]] <- dord[i]
      dhat[data[i, 2], data[i, 1]] <- dord[i]
    }
    snew <- sum(w * (dhat - dnew) ^ 2)
    if (verbose) {
      cat(
        "itel = ",
        formatC(itel, format = "d"),
        "sold = ",
        formatC(sold, digits = 10, format = "f"),
        "smid = ",
        formatC(smid, digits = 10, format = "f"),
```

```
        "snew = ",
        formatC(snew, digits = 10, format = "f"),
        "\n"
      )
    }
    if ((itel == itmax) || ((sold - snew) < eps)) {
      break
    }
    xold <- xnew
    dold <- dnew
    sold <- snew
    itel <- itel + 1
  }
  return(list(
    b = bold,
    v = v,
    ranks = data[,3],
    dvec = dord,
    dnew = as.dist(dnew),
    dhat = as.dist(dhat),
    xnew = xnew,
    loss = snew,
    itel = itel
  ))
}
```

References

Coombs, C. H. 1954. “A Method for the Study of Interstimulus Similarity.” *Psychometrika* 19: 183–94.

De Gruijter, D. N. M. 1967. “The Cognitive Structure of Dutch Political Parties in 1966.” Report E019-67. Psychological Institute, University of Leiden.

De Leeuw, J. 1970. “The Positive Orthant Method for Nonmetric Multidimensional Scaling.” Research Report 001-70. Leiden, The Netherlands: Department of Data Theory FSW/RUL.

———. 2018. “The Positive Orthant Method .” 2018.

Guttman, L. 1969. “Smallest Space Analysis by the Absolute Value Principle.” In *Proceedings of the XIX International Congress of Psychology, London*.

Hartmann, W. 1979. *Geometrische Modelle zur Analyse empirischer Data*. Akademie Verlag.

Johnson, R. M. 1973. “Pairwise Nonmetric Multidimensional Scaling.” *Psychometrika* 38 (12–18).

Richardson, M. W. 1938. “Multidimensional Psychophysics.” *Psychological Bulletin* 35: 659–60.

Torgerson, W. S. 1958. *Theory and Methods of Scaling*. New York: Wiley.