

Least Squares Euclidean Multidimensional Scaling

Jan de Leeuw

Started October 02, 2020. Last update June 07, 2021

Contents

Note	25
Preface	29
1 Introduction	33
1.1 Brief History	33
1.1.1 Milestones	34
1.2 Basic MDS	35
1.2.1 Kruskal's stress	37
1.2.1.1 Square root	39
1.2.1.2 Weights	40
1.2.1.3 Normalization	41
1.2.2 Data Asymmetry	42
1.2.3 Local and Global	43
1.3 Generalizations	43
1.3.1 Non-metric	43
1.3.2 fstress	45
1.3.3 Constraints	45
1.3.4 Replications	45
1.3.5 Distance Asymmetry	45

2 Properties of Stress	47
2.1 Notation	47
2.1.1 Expanding	47
2.1.2 Matrix Expressions	48
2.2 Global Properties	50
2.2.1 Boundedness	50
2.2.2 Invariance	50
2.2.3 Continuity	51
2.2.4 Coercivity	51
2.3 Convexity	51
2.3.1 Distances	52
2.3.2 DC Functions	53
2.3.3 Subdifferentials	54
2.3.4 Negative Dissimilarities	54
2.4 Guttman Transform	55
2.5 Differentiability	57
2.5.1 Regularity	57
2.5.1.1 Distances	57
2.5.1.2 Rho and stress	58
2.5.2 Expansions	59
2.5.3 Directional Derivatives	60
2.5.3.1 First Order	60
2.5.3.2 Second Order	60
2.5.4 Special Expansions	62
2.5.5 Zero Distance Subspaces	62
2.5.6 Distance Smoothing	63
2.6 Stationary Points	66

CONTENTS	5
2.6.1 Local Maxima	66
2.6.2 Local Minima	66
2.7 Stress Envelopes	67
2.7.1 CS Majorization	68
2.7.2 AM/GM Minorization	68
2.7.3 Dualities	70
2.8 How Large is My Stress ?	71
3 Stress Spaces	73
3.1 Configuration Space	73
3.2 Coefficient Space	73
3.2.1 On the Planes	73
3.2.2 General	74
3.3 Sphere Space	76
3.4 Cross-Product Space	77
4 Pictures of Stress	79
4.1 Coefficient Space	80
4.2 Global Perspective	82
4.3 Global Contour	84
4.4 Stationary Points	84
4.4.1 First Minimum	84
4.4.2 Second Minimum	84
4.4.3 Third Minimum	88
4.4.4 First Saddle Point	88
4.4.5 Second Saddle Point	92
4.5 Another Look	92
4.6 A Final Look	94

4.7	Discuss	97
4.8	Coordinates	98
5	Classical Multidimensional Scaling	101
5.1	Algebra	102
5.1.1	Torgerson Transform	102
5.1.2	Schoenberg's Theorem	102
5.2	Approximation	103
5.2.1	Low Rank Approximation of the Torgersen Transform .	103
5.2.2	Minimization of Strain	104
5.2.3	Approximation from Below	104
6	Minimization of Stress	107
6.1	Gradient Methods	107
6.2	Initial Configurations	108
6.3	On MM Algorithms	108
6.4	Smacof Algorithm	109
6.4.1	Global Convergence	110
6.4.2	From the CS Inequality	111
6.4.3	From Majorization	111
6.4.4	Component Rotated Smacof	112
6.4.5	Local Convergence	112
6.4.5.1	Rotation to PC	112
6.4.6	Negative Dissimilarities	114
6.4.7	Unweighting	114
6.4.8	Smacof in Coefficient Space	116
6.5	Regions of Attraction	116
6.5.1	Smacof	116
6.5.2	Newton	117

7 Acceleration of Convergence	123
7.1 Simple Acceleration	123
7.2 One-Parameter Methods	128
7.3 SQUAREM	128
7.4 Vector extrapolation methods	128
8 Nonmetric MDS	129
8.1 Generalities	129
8.2 Single and Double Phase	130
8.2.1 Double Phase	130
8.2.2 Single Phase	132
8.3 Affine NMDS	133
8.4 Conic MDS	134
8.4.1 Normalization	134
8.4.2 Inner Iterations	137
8.4.3 Stress-1 and Stress-2	137
9 Interval MDS	139
9.1 The Additive Constant	139
9.1.1 Algebra	140
9.1.2 Examples	143
9.1.2.1 Small Example	143
9.1.2.2 De Gruijter Example	146
9.1.2.3 Ekman Example	147
9.1.3 A Variation	148

10 Polynomial and Splinical MDS	151
10.1 Polynomial MDS	151
10.1.1 Monotone Polynomials	151
10.2 Splinical MDS	151
10.3 Splines	151
10.3.1 B-splines	152
10.3.1.1 Boundaries	153
10.3.1.2 Normalization	153
10.3.1.3 Recursion	154
10.3.1.4 Illustrations	155
10.3.2 I-splines	155
10.3.2.1 Increasing Coefficients	158
10.3.2.2 Increasing Values	158
10.3.3 Time Series Example	160
10.3.3.1 B-splines	160
10.3.3.2 I-splines	161
10.3.3.3 B-Splines with monotone weights	162
10.3.3.4 B-Splines with monotone values	163
10.3.4 Monotone Splines	164
11 Ordinal MDS	165
11.1 Monotone Regression	165
11.1.1 Simple Monotone Regression	165
11.1.2 Weighted Monotone Regression	167
11.1.3 Normalized Cone Regression	167
11.2 Alternating Least Squares	167
11.3 Kruskal's Approach	168

11.4 Guttman's Approach	168
11.4.0.1 Rank Images	168
11.4.0.2 Single and Double Phase	170
11.4.0.3 Hard and Soft Squeeze	170
11.4.1 Smoothness of Ordinal Loss Functions	171
11.5 Scaling with Distance Bounds	171
11.6 Bounds on Stress	171
12 Unidimensional Scaling	173
12.1 An example	173
12.1.1 Perspective	174
12.1.2 Contour	175
12.2 Order Formulation	177
12.3 Permutation Formulation	178
12.4 Sign Matrix Formulation	178
12.5 Algorithms for UMDS	179
12.5.1 SMACOF	179
12.5.2 SMACOF (smoothed)	179
12.5.3 Branch-and-Bound	180
12.5.4 Dynamic Programming	180
12.5.5 Simulated Annealing	180
12.5.6 Penalty Methods	180
13 Full-dimensional Scaling	181
13.1 Convexity	181
13.2 Optimality	181
13.3 Iteration	181
13.4 Cross Product Space	181

13.5 Full-dimensional Scaling	182
13.6 Ekman example	185
14 Unfolding	189
14.1 Algebra	189
14.1.1 One-dimensional	191
14.2 Classical Unfolding	192
14.3 Nonmetric Unfolding	192
14.3.1 Degenerate Solutions	192
14.3.1.1 Which Stress	192
14.3.1.2 l'Hopital	192
14.3.1.3 Penalizing	193
14.3.1.4 Restricting Regression	193
15 Constrained Multidimensional Scaling	195
15.1 Basic Partitioning	195
15.2 Unweighting	196
15.3 Constraints on the Distances	197
15.3.1 Rectangles	197
15.4 Linear Constraints	197
15.4.1 Uniqueness	197
15.4.2 Combinations	198
15.4.3 Step Size	198
15.4.4 Single Design Matrix	198
15.4.5 Multiple Design Matrices	198
15.5 Circular MDS	198
15.5.1 Some History	199
15.5.2 Primal Methods	200

CONTENTS	11
15.5.3 Dual Methods	202
15.6 Elliptical MDS	204
15.6.1 Primal	204
15.6.2 Dual	205
15.7 Distance Bounds	208
15.8 Localized MDS	209
15.9 MDS as MVA	209
15.10 Horseshoes	209
16 Asymmetry in MDS	211
16.1 Conditional Rankings	211
16.2 Confusion Matrices	211
16.3 The Slide Vector	211
17 Individual Differences	213
17.1 Replications	214
17.2 INDSCAL/PARAFAC	214
17.3 IDIOSCAL/TUCKALS	214
18 Nominal MDS	215
18.1 Binary Dissimilarities	216
18.2 Indicator matrix	216
18.3 Unfolding Indicator Matrices	218
18.4 Linear Separation	222
18.5 Circular Separation	224
18.6 Multidimensional Scalogram Analysis	224

19 Sstress and strain	225
19.1 sstress	225
19.1.1 sstress and stress	226
19.1.2 Decomposition	226
19.1.3 Full-dimensional sstress	227
19.1.4 Minimizing sstress	228
19.1.4.1 ALSCAL	228
19.1.4.2 Majorization	231
19.1.4.3 SOS	232
19.1.4.4 Duality	232
19.1.4.5 ALS Unfolding	232
19.1.5 Bounds for sstress	233
19.2 strain	233
19.2.1 Unweighted	234
19.2.2 Using Additivity	235
20 fstress and rstress	237
20.1 fstress	237
20.1.1 Use of Weights	237
20.1.2 Convexity	238
20.2 rStress	238
20.2.1 Using Weights	238
20.2.2 Minimizing rstress	239
20.3 mstress	239
20.4 astress	241
20.5 pstress	241

CONTENTS	13
----------	----

21 Alternative Least Squares Loss	243
21.1 Sammon's MDS	243
21.2 McGee's Work	243
21.3 Shepard's Nonmetric MDS	243
21.4 Guttman's Nonmetric MDS	243
21.5 Positive Orthant Nonmetric MDS	243
21.6 Role Reversal	243
22 Inverse Multidimensional Scaling	245
22.1 Basic IMDS	246
22.2 [1]	250
22.3 [1,] -0.5	250
22.4 [2,] -0.5	250
22.5 [3,] 0.5	250
22.6 [4,] 0.5	250
22.7 [1] [2] [3] [4]	250
22.8 [1,] 1 -1 1 -1	250
22.9 [2,] -1 1 -1 1	250
22.10[3,] 1 -1 1 -1	250
22.11[4,] -1 1 -1 1	250
22.12[1] [2] [3] [4]	251
22.13[1,] 1 -1 -1 1	251
22.14[2,] -1 1 1 -1	251
22.15[3,] -1 1 1 -1	251
22.16[4,] 1 -1 -1 1	251
22.17[1] [2] [3] [4]	251
22.18[1,] -2 2 0 0	251

22.19[2,] 2 -2 0 0	251
22.20[3,] 0 0 2 -2	251
22.21[4,] 0 0 -2 2	251
22.22[,1] [,2]	252
22.23[1,] -0.5 -0.5	252
22.24[2,] -0.5 0.5	252
22.25[3,] 0.5 0.5	252
22.26[4,] 0.5 -0.5	252
22.271 2 3	252
22.282 1.000000	252
22.293 1.414214 1.000000	252
22.304 1.000000 1.414214 1.000000	252
22.31[1] -0.5 0.5 -0.5 0.5	252
22.32[,1]	254
22.33[1,] -0.6708204	254
22.34[2,] -0.2236068	254
22.35[3,] 0.2236068	254
22.36[4,] 0.6708204	254
22.371 2 3	254
22.382 3.464102	254
22.393 4.472136 2.828427	254
22.404 6.324555 4.472136 3.464102	254
22.411 2 3	255
22.422 4.330127	255
22.433 5.590170 2.121320	255
22.444 4.743416 5.590170 4.330127	255
22.451 2 3	255

CONTENTS 15

22.462 3.983717	255
22.473 3.801316 4.101219	255
22.484 6.640783 3.801316 3.983717	255
22.491 2 3	255
22.502 1.914908	255
22.513 5.472136 2.828427	255
22.524 6.324555 3.472136 5.013295	255
22.531 2 3	258
22.542 20.784610	258
22.553 0.000000 5.656854	258
22.564 0.000000 13.416408 0.000000	258
22.571 2 3	258
22.582 13.856406	258
22.593 4.472136 0.000000	258
22.604 0.000000 13.416408 0.000000	258
22.611 2 3	258
22.622 20.78461	258
22.633 0.00000 22.62742	258
22.644 0.00000 0.00000 20.78461	258
22.651 2 3	258
22.662 0.000000	258
22.673 13.416408 5.656854	258
22.684 0.000000 0.000000 20.784610	258
22.691 2 3	258
22.702 0.000000	258
22.713 13.416408 0.000000	258
22.724 0.000000 4.472136 13.856406	258

22.731 2 3	258
22.742 0.000000	258
22.753 4.472136 0.000000	258
22.764 8.432738 4.472136 0.000000	258
22.771 2 3	258
22.782 0.000000	258
22.793 0.000000 5.656854	258
22.804 12.649111 0.000000 0.000000	258
22.81[1] 460.00000 248.00000 1072.00000 460.00000 248.00000 36.44444 112.00000	259
22.82[1] 24.00000 24.00000 24.00000 21.75000 12.00000 13.33333 6.66667 13.33333	260
22.83[9] 17.33333 17.33333 19.68000 13.33333 6.66667 17.33333 6.66667 21.75000	260
22.84[17] 6.66667 60.00000 24.00000 21.75000 19.68000 17.33333 21.75000 13.33333	260
22.85[25] 19.68000 17.33333 17.33333 21.75000 17.33333 17.33333 13.33333 6.66667	260
22.86[33] 21.75000 17.33333 19.68000 13.33333 17.33333 19.68000 19.68000 17.33333	260
22.87[41] 6.66667 17.33333	260
22.88[,1] [,2]	261
22.89[1,] 0.0000000 0.0000000	261
22.90[2,] 0.0000000 0.0000000	261
22.91[3,] 0.0000000 -0.8164966	261
22.92[4,] 0.0000000 0.4082483	261
22.93[5,] 0.7071068 0.0000000	261
22.94[6,] -0.7071068 0.4082483	261
22.95[,1] [,2] [,3] [,4] [,5] [,6]	262

22.96[1,] 0 0 0 1 1 1	262
22.97[2,] 0 0 0 1 1 1	262
22.98[3,] 0 0 0 1 1 1	262
22.99[4,] 1 1 1 0 0 0	262
22.10[5,] 1 1 1 0 0 0	262
22.10[6,] 1 1 1 0 0 0	262
22.10[7,] 5 6	262
22.10[8,] 0.0000000 1.414214 1.632993	262
22.10[9,] 0.8164966 0.000000 0.000000	262
22.10[10,] 1.2247449 1.080123 1.414214	262
22.10[11,] 5 6	263
22.10[12,] 0.8164966 0.000000 0.000000	263
22.10[13,] 0.0000000 1.414214 1.632993	263
22.10[14,] 1.2247449 1.080123 1.414214	263
22.11[KVP PvdA VVD ARP CHU CPN PSP BP D66	265
22.11[KVP 0.00 5.63 5.27 4.60 4.80 7.54 6.73 7.18 6.17	265
22.11[PvdA 5.63 0.00 6.72 5.64 6.22 5.12 4.59 7.22 5.47	265
22.11[VVD 5.27 6.72 0.00 5.46 4.97 8.13 7.55 6.90 4.67	265
22.11[ARP 4.60 5.64 5.46 0.00 3.20 7.84 6.73 7.28 6.13	265
22.11[CHU 4.80 6.22 4.97 3.20 0.00 7.80 7.08 6.96 6.04	265
22.11[CPN 7.54 5.12 8.13 7.84 7.80 0.00 4.08 6.34 7.42	265
22.11[PSP 6.73 4.59 7.55 6.73 7.08 4.08 0.00 6.88 6.36	265
22.11[BP 7.18 7.22 6.90 7.28 6.96 6.34 6.88 0.00 7.36	265
22.11[D66 6.17 5.47 4.67 6.13 6.04 7.42 6.36 7.36 0.00	265
22.12[01] [,2]	266
22.12[11,] 1.78 3.579	266
22.12[22,] -1.46 2.297	266

22.12 ^B ,] 3.42 -2.776	266
22.12 ^H ,] 3.30 1.837	266
22.12 ^F ,] 3.84 0.308	266
22.12 ^G ,] -5.09 -0.044	266
22.12 ^I ,] -3.79 2.132	266
22.12 ^J ,] -2.86 -4.318	266
22.12 ^K ,] 0.86 -3.015	266
22.13 ^A KVP PvdA VVD ARP CHU CPN PSP BP D66	268
22.13 ^B KVP 0.0 32.6 9.7 9.3 24.6 24.3 11.7 18.9 15.2	268
22.13 ^C PvdA 32.6 0.0 25.7 36.9 20.7 34.1 36.1 29.7 22.2	268
22.13 ^D VVD 9.7 25.7 0.0 17.4 32.0 34.4 22.3 23.5 20.4	268
22.13 ^E ARP 9.3 36.9 17.4 0.0 28.0 33.7 25.6 28.9 23.0	268
22.13 ^F CHU 24.6 20.7 32.0 28.0 0.0 20.8 31.9 33.3 26.3	268
22.13 ^G CPN 24.3 34.1 34.4 33.7 20.8 0.0 24.9 30.7 31.8	268
22.13 ^H PSP 11.7 36.1 22.3 25.6 31.9 24.9 0.0 23.7 27.7	268
22.13 ^I BP 18.9 29.7 23.5 28.9 33.3 30.7 23.7 0.0 35.0	268
22.13 ^J D66 15.2 22.2 20.4 23.0 26.3 31.8 27.7 35.0 0.0	268
22.14 ^A KVP PvdA VVD ARP CHU CPN PSP BP D66	270
22.14 ^B KVP 0.00 3.48 0.00 0.00 2.34 0.00 0.00 0.00 0.00	270
22.14 ^C PvdA 3.48 0.00 0.00 0.00 0.00 0.93 0.00 0.00	270
22.14 ^D VVD 0.00 0.00 0.00 0.00 0.83 0.00 0.00 0.00 0.00	270
22.14 ^E ARP 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00	270
22.14 ^F CHU 2.34 0.00 0.83 0.00 0.00 0.00 0.00 0.00 0.00	270
22.14 ^G CPN 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00	270
22.14 ^H PSP 0.00 0.93 0.00 0.00 0.00 0.00 0.00 0.00 0.00	270
22.14 ^I BP 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00	270
22.14 ^J D66 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00	270

22.15[1] 0.00000 0.00000 0.00000 -5.86238 0.00000 -2.96688 0.00000	272
22.15[8] -2.96688 -4.47894 -4.47894 -5.71312 -2.96688 0.00000 -5.00453272	
22.15[15] 0.00000 -5.86238 0.00000 -12.00000 0.00000 -5.86238 - 5.71312	272
22.15[22] -5.00453 -5.86238 -2.96688 -5.71312 -4.47894 -5.00453 - 5.86238	272
22.15[29] -5.00453 -5.00453 -2.96688 0.00000 -5.86238 -4.47894 - 5.71312	272
22.15[36] -2.96688 -5.00453 -5.71312 -5.71312 -4.47894 0.00000 - 4.47894	272
22.15Multiple Solutions	274
22.15Minimizing iStress	276
22.15Order three	277
22.15Order Four	282
23 Stability of MDS Solutions	283
23.1 Null Distribution	283
23.2 Pseudo-confidence Ellipsoids	283
23.3 A Pseudo-Bootstrap	283
24 In Search of Global Minima	285
24.1 Random Starts	286
24.1.1 Examples	287
24.1.1.1 Ekman	287
24.1.1.2 De Gruijter	290
24.2 Tunneling, Filling, Annealing, etc.	294
24.3 Cutting Planes	294
24.3.1 On the Circle	294
24.3.2 Cauchy Step Size	299

24.3.3	Balls	300
24.3.3.1	Outer Approximation	300
24.4	Distance Smoothing	300
24.5	Penalizing Dimensions	301
24.5.1	Local Minima	302
24.5.2	Algorithm	303
24.5.3	Examples	303
24.5.3.1	Chi Squares	304
24.5.3.2	Regular Simplex	305
24.5.3.3	Intelligence	308
24.5.3.4	Countries	310
24.5.3.5	Dutch Political Parties	311
24.5.3.6	Ekman	315
24.5.3.7	Morse in Two	317
24.5.3.8	Vegetables	318
24.5.3.9	Plato	319
24.5.3.10	Morse in One	323
25	Mathematical Addenda	327
25.1	Notation	327
25.1.1	Symbols	327
25.1.2	Summation	328
25.2	Matrices	328
25.2.1	Matrix Spaces	328
25.2.2	Constants	329
25.2.3	Diff Matrices	329
25.2.4	Sign Matrices	330

25.2.5 Sums and Products	330
25.2.6 Inverse	331
25.2.7 The Loewner order	331
25.2.8 Eigen Decomposition	331
25.2.9 Singular Value Decomposition	332
25.2.10 Moore-Penrose Inverse	332
25.2.11 Full-rank Decomposition	332
25.2.12 SDC matrices	333
25.3 Inequalities	333
25.3.1 Cauchy-Schwartz	333
25.3.2 Arithmetic/Geometric Mean	333
25.4 Calculus/Analysis	334
25.4.1 Functions	334
25.4.2 Differentiation	334
25.4.2.1 Derivative	334
25.4.2.2 Partial Derivatives	335
25.4.2.3 Jacobian	335
25.4.2.4 Gradient	335
25.4.2.5 Hessian	335
25.4.2.6 Differentials	336
25.4.3 Directional Derivatives	336
25.4.3.1 First Order	336
25.4.3.2 Second Order	336
25.4.4 Local Minima, Critical Points	337
25.4.5 Necessary Conditions	337
25.4.6 l'Hôpital's Rule	337
25.4.7 Implicit Functions	338

25.5 Convexity	338
25.5.1 Existence of Minima	338
25.5.2 Duality	338
25.5.3 Subdifferentials	338
25.5.4 DC Functions	339
25.5.5 Danskin-Berge Theorem	340
25.6 Fixed-Point Iterations	340
25.6.1 Point-to-set Maps	341
25.6.2 Zangwill's Theorem	341
25.6.3 Ostrowski's Theorem	341
25.7 Least Squares	341
25.7.1 Quadratic Programming	341
25.7.1.1 Example 1: Simple Monotone Regression . . .	344
25.7.1.2 Example 2: Monotone Regression with Ties .	346
25.7.1.3 Example 3: Weighted Rounding	349
25.7.1.4 Example 4: Monotone Polynomials	350
25.8 Quadratic penalties	354
A Code	357
A.1 R Code	357
A.1.1 utilities.R	357
A.1.2 commom/indexing.r	357
A.1.3 commom/io.r	359
A.1.4 commom/linear.r	361
A.1.5 commom/nextPC.r	363
A.1.6 commom/smacof.r	364
A.1.7 properties.R	368

CONTENTS	23
----------	----

A.1.8 pictures.R	371
A.1.9 classical.R	378
A.1.10 minimization.R	382
A.1.11 full.R	382
A.1.12 unfolding.R	385
A.1.13 constrained.R	389
A.1.14 nominal.R	395
A.1.15 sstress.R	397
A.1.16 inverse.R	400
A.1.17 global.R	410
A.1.18 mathadd.R	424
A.2 C code	431
A.2.1 deboor.c	431
A.2.2 cleanup.c	437
A.2.3 jacobi.c	437
A.2.4 jbkTies.c	441
A.2.5 matrix.c	447
A.2.6 jeffrey.c	448
A.2.7 mySort.c	449
A.2.8 nextPC.c	451
B Data	453
B.1 Small	453
B.2 De Gruijter	453
B.3 Ekman	454
B.4 Vegetables	454
C Unlicense	455

Note

This book will be expanded/updated frequently. The directory `deleeuw-pdx.net/pubfolders/stress` has a pdf version, the bib file, the complete Rmd file with the codechunks, and the R and C source code. All suggestions for improvement of text or code are welcome, and some would be really beneficial. For example, I only use base R graphics, nothing more fancy, because base graphics is all I know.

All text and code are in the public domain and can be copied, modified, and used by anybody in any way they see fit. Attribution will be appreciated, but is not required. For completeness we include a slightly modified version of the Unlicense as appendix C.

I number and label *all* displayed equations. Equations are displayed, instead of inlined, if and only if

- they are important, or
- they are referred to elsewhere in the text, or
- if not displaying them messes up the line spacing.

All code chunks in the text are named. Theorems, lemmas, chapters, sections, subsections and so on are also named and numbered, using bookdown/Rmarkdown.

I have been somewhat hesitant to use lemmas, theorems, and corollaries in this book. I am not a mathematician, and, given the level of almost all of the text, their use seems somewhat pretentious. But ultimately they do provide a useful, maybe even indispensable, organizational tool. As a compromise I use *result* for those results that are too trivial to be elevated to theorem

status. These results often do not even need a proof. If there is a proof of a lemma, theorem, corollary, or result, it ends with a ■.

Another idiosyncracy: if a line in multiline displayed equation ends with “=,” then the next line begins with “=.” If it ends with “+,” then the next line begin with “+,” and if it ends with “-” the next line begins with “+” as well. I’ll try to avoid ending a line with “+” or “-,” especially with “-,” but if it happens you are warned. A silly example is

$$\begin{aligned}
 & (x + y)^2 - && (1) \\
 & + 4x = && (2) \\
 & = x^2 + y^2 - 2x = && (3) \\
 & = (x - y)^2 \geq && (4) \\
 & \geq 0. && (5)
 \end{aligned}$$

Just as an aside: if I refer to something that has been mentioned “above” I mean something that comes earlier in the book and “below” refers to anything that comes later. This always confuses me, so I had to write it down.

The dilemma of whether to use “we” or “I” throughout the book is solved in the usual way. If I feel that a result is the work of a group (me, my co-workers, and the giants on whose shoulders we stand) then I use “we.” If it’s an individual decision, or something personal, then I use “I.” The default is “we,” as it always should be in scientific writing.

Chapter 25 has some of the necessary mathematical background material, both notation and results, sometimes with specific elaborations that seem useful for the book. Other mathematical material is worked into the individual chapters, if it is central to the development of MDS technique, or too bulky to be in a mathematical appendix. Examples are splines, majorization, unweighting, monotone regression, and the basic Zangwill and Ostrowski fixed point theorems we need. Other chapters just have references to chapter 25 when additional background is needed.

There is an appendix A with code, and an appendix B with data sets. These contain brief descriptions and links to the supplementary materials directories, which contain the actual code and data.

Something about code and R/C

I will use this note to thank Rstudio, in particular J. J. Allaire and Yihui Xi, for their contributions to the R universe, and for their promotion of open source software and open access publications. Not too long ago I was an ardent LaTeX user, firmly convinced I would never use anything else again in my lifetime. As I was convinced before that, by the way, that I would never use anything besides, in that order, FORTRAN, PL/I, APL, and (X)Lisp. But I lived too long. And then R, Rstudio, (R)Markdown, bookdown, and blogdown came along.



Figure 1: Forrest Young, Bepi Pinner, Jean-Marie Bouroche, Yoshio Takane, Jan de Leeuw at La Jolla, August 1975

Preface

This book is definitely *not* an impartial and balanced review of all of multidimensional scaling (MDS) theory and history. It emphasizes the computation, and the mathematics needed for computation. In addition, it is a summary of over 50 years of MDS work by me, either solo or together with my many excellent current or former co-workers and co-authors. It is heavily biased in favor of the **smacof** formulation of MDS (De Leeuw (1977a), De Leeuw and Heiser (1977), De Leeuw and Mair (2009)), and the corresponding majorization (or MM) algorithm. And, moreover, I am shamelessly squeezing in as many references to my published and unpublished work as possible, with links to the corresponding pdf's if they are available.

I have not organized the book along historical lines because most of the early techniques and results have been either drastically improved or completely abandoned. Nevertheless, some personal historical perspective may be useful. I will put most of it in this preface, so uninterested readers can easily skip it.

I got involved in MDS in 1968 when John van de Geer returned from a visit to Clyde Coombs in Michigan and started the Department of Data Theory in the Division of Social Sciences at Leiden University. I was John's first hire, although I was still a graduate student at the time.

Remember that Clyde Coombs was running the Michigan Mathematical Psychology Program, and he had just published his remarkable book "A Theory of Data" (Coombs (1964)). The name of the new department in Leiden was taken from the title of that book, and Coombs was one of the first visitors to give a guest lecture there.

This is maybe the place to clear up some possible misunderstandings about the name "Data Theory." Coombs was mainly interested in a taxonomy of data types, and in pointing out that "data" were not limited to a table or

data-frame of objects by variables. In addition, there were also similarity ratings, paired comparisons, and unfolding data. Coombs also emphasized that data were often non-metric, i.e. ordinal or categorical, and that it was possible to analyze these ordinal or categorical relationships directly, without first constructing numerical scales to which classical techniques could be applied. One of the new techniques discussed in Coombs (1964) was a non-metric form of MDS, in which not only the data but also the representation of the data in Euclidean space was non-metric.

John van de Geer had just published Van de Geer (1967). In that book, and in the subsequent book Van de Geer (1971), he developed his unique geometric approach to multivariate analysis. Relationship between variables, and between variables and individuals, were not just discussed using matrix algebra, but were also visualized in diagrams. This was related to the geometry in Coombs' Theory of Data, but it concentrated on numerical data in the form of rectangular matrices of objects by variables.

Looking back it is easy to see that both Van de Geer and Coombs influenced my approach to data analysis. I inherited the emphasis on non-metric data and on visualization. But, from the beginning, I interpreted "Data Theory" as "Data Analysis," with the emphasis shifting almost completely to techniques, loss functions, implementations, algorithms, optimization, computing, and programming. This is of interest because in 2020 my old Department of Statistics at UCLA, together with the Department of Mathematics, started a bachelor's program in Data Theory, in which "Emphasis is placed on the development and theoretical support of a statistical model or algorithmic approach. Alternatively, students may undertake research on the foundations of data science, studying advanced topics and writing a senior thesis." This sounds like a nice hybrid of Data Theory and Data Analysis, with a dash of computer science mixed in.

Computing and optimization were in the air in 1968, not so much because of Coombs, but mainly because of Roger Shepard, Joe Kruskal, and Doug Carroll at Bell Labs in Murray Hill. John's other student Eddie Roskam and I were fascinated by getting numerical representations from ordinal data by minimizing explicit least squares loss functions. Eddie wrote his dissertation in 1968 (Roskam (1968)). In 1973 I went to Bell Labs for a year, and Eddie went to Michigan around the same time to work with Jim Lingoes, resulting in Lingoes and Roskam (1973).

My first semi-publication was De Leeuw (1968c), quickly followed by a long sequence of other, rather rambling, internal reports. Despite this very informal form of publication the sheer volume of them got the attention of Joe Kruskal and Doug Carroll, and I was invited to spend the academic year 1973-1974 at Bell Laboratories. That visit somewhat modified my cavalier approach to publication, but I did not become half-serious in that respect until meeting with Forrest Young and Yoshio Takane at the August 1975 US-Japan seminar on MDS in La Jolla. Together we used the alternating least squares approach to algorithm construction that I had developed since 1968 into a quite formidable five-year publication machine, with at its zenith Takane, Young, and De Leeuw (1977).

In La Jolla I gave the first presentation of the majorization method for MDS, later known as smacof, with the first formal convergence proof. The canonical account of smacof was published in De Leeuw (1977a). Again I did not bother to get the results into a journal or into some other effective form of publication. The basic theory for what became known as smacof was also presented around the same time in the book chapter De Leeuw and Heiser (1977).

In 1978 I was invited to the Fifth International Symposium on Multivariate Analysis in Pittsburgh to present what became De Leeuw and Heiser (1980). There I met Nan Laird, one of the authors of the basic paper on the EM algorithm (Dempster, Laird, and Rubin (1977)). I remember enthusiastically telling her on the conference bus that EM and smacof were both special case of the general majorization approach to algorithm construction, which was consequently born around the same time. But that is a story for a companion volume, which currently only exists in a very preliminary stage (De Leeuw (2016a)).

My 1973 PhD thesis (De Leeuw (1973a), reprinted as De Leeuw (1984a)) was actually my second attempt at a dissertation. I had to get a PhD, any PhD, before going to Bell Labs, because of the discrepancy in the Dutch and American academic title and reward systems. I started writing a dissertation on MDS, in the spirit of what later became De Leeuw and Heiser (1982). But halfway through I lost interest and became impatient, and I decided to switch to nonlinear multivariate analysis. This second attempt did produce a finished dissertation, which blossomed, with the help of many, into Gifi (1990). But that again is a different history, which I will tell some other time

in yet another companion volume (De Leeuw (2016d)). For a long time I did not do much work on MDS, until the arrival of the R language and Patrick Mair led to a resurgence of my interest, and ultimately to De Leeuw and Mair (2009) and Mair, Groenen, and De Leeuw (2019).

I consider this MDS book to be a summary and extension of the basic papers De Leeuw (1977a) (reprinted as De Leeuw (2005a)), De Leeuw and Heiser (1977), De Leeuw and Heiser (1980), De Leeuw and Heiser (1982), and De Leeuw (1988) (published version of De Leeuw (1984b)), all written 30-40 years ago. Footprints in the sands of time. It can also be seen as an elaboration of the more mathematical sections of the excellent and comprehensive textbook of Borg and Groenen (2005). That book has much more information about the origins, the data, and the applications of MDS, as well as on the interpretation of MDS solutions. In this book we concentrate almost exclusively on the mathematical, computational, and programming aspects of MDS.

For those who cannot get enough of me, there is a data base of my publications since 1965, with links to pdf's, at http://deleeuw.pdx.net/janspubs/0_bib_material/. Unpublished reports written since my retirement in 2014 are at <http://deleeuw.pdx.net/publication/>.

There are many, many people I have to thank for my scientific education. Fifty-five years is a long time, and so many important teachers and researchers have crossed my path. Since I will join them in the not too distant future I will gratefully mention those researchers who had a major influence on my work and who are not with us any longer: Louis Guttman (died 1987), Clyde Coombs (died 1988), Warren Torgerson (died 1999), Forrest Young (died 2006), John van de Geer (died 2008), Joe Kruskal (died 2010), Doug Carroll (died 2011), and Rod McDonald (died 2012).

Chapter 1

Introduction

In this book we study the smacof family of *Multidimensional Scaling (MDS)* from now on) techniques. In MDS the data consist of some type of information about the *dissimilarities* between a number of *objects*. The information we have about these dissimilarities can be numerical, ordinal, or categorical. Thus we may have the actual values of some or all of the dissimilarities, we may know their rank order, or we may have a classification of them into a small number of qualitative bins.

MDS techniques map the objects into *points* in a metric space in such a way that the relation between the distances between the points mirrors or approximates the relation between dissimilarities of the objects. For numerical dissimilarities it is clear what “approximation” means, although it can be measured in many different ways. For ordinal and categorical dissimilarities the notion of approximation is less clear, and we have to develop more specialized techniques to measure fit.

1.1 Brief History

De Leeuw and Heiser (1980)

This is reviewed ably in the presidential address of Torgerson (1965).

As I mentioned in the preface, a complete overview of the state of the art until 2005 is Borg and Groenen (2005).

A more recent review paper, emphasizing smacof, is Groenen and Van de Velden (2016).

This section has a different emphasis. We limit ourselves to developments in Euclidean MDS, to contributions with direct computational consequences with a direct or indirect link to psychometrics, and to work before 1960.

Our history review takes the form of brief summaries of what we consider to be milestone papers or books.

1.1.1 Milestones

Torgerson (1952) Torgerson (1965)

Shepard (1962a) Shepard (1962b)

Kruskal (1964a) Kruskal (1964b)

Guttman (1968)

De Leeuw (1977a) De Leeuw and Heiser (1977)

There was some early work by Richardson, Messick, Abelson and Torgerson who combined Thurstonian scaling of similarities with the mathematical results of Schoenberg (1935) and G. Young and Householder (1938). Despite these early contributions it makes sense, certainly from the point of view of my personal history, but probably more generally, to think of MDS as starting as a widely discussed, used, and accepted technique since the book by Torgerson (1958). This was despite the fact that in the fifties and sixties computing eigenvalues and eigenvectors of a matrix of size 20 or 30 was still a challenge.

A few years later the popularity of MDS got a large boost by developments centered at Bell Telephone Laboratories in Murray Hill, New Jersey, the magnificent precursor of Silicon Valley. First there was nonmetric MDS by Shepard (1962a), Shepard (1962b) and Kruskal (1964a), Kruskal (1964b), And later another major development was the introduction of individual difference scaling by Carroll and Chang (1970) and Harshman (1970). Perhaps even more important was the development of computer implementations of these new techniques. Some of the early history of nonmetric MDS is in De Leeuw (2017f).

Around the same time there were interesting theoretical contributions in Coombs (1964), which however did not really influence the practice of MDS. And several relatively minor variations of the Bell Laboratories approach were proposed by Guttman (1968), but Guttman's influence on further MDS implementations turned out to be marginal.

The main development after the Bell Laboratories surge was probably smacof. Initially, in De Leeuw (1977a), this stood for *Scaling by Maximizing a Convex Function*. Later it was also used to mean *Scaling by Majorizing a Complicated Function*. Whatever. In this book smacof just stands for smacof. No capitals.

The first smacof programs were written in 1977 in FORTRAN at the Department of Data Theory in Leiden (Heiser and De Leeuw (1977)). Eventually they migrated to SPSS (for example, J. M. Meulman and Heiser (2012)) and to R (De Leeuw and Mair (2009)). The SPSS branch and the R branch have diverged somewhat, and continue to be developed independently.

1.2 Basic MDS

Following Kruskal and Shepard we measure the fit of distances to dissimilarities using an explicit real-valued *loss function* (a.k.a. *badness-of-fit measure*), which is minimized over the possible maps of the objects into the metric space. This is a very general definition of MDS, covering all kinds of variations in the target metric space and in the way fit is measured. Obviously we can not discuss all possible forms and implementations, which also includes various techniques more properly discussed as cluster analysis, classification, or discrimination.

To outline our scope we define *basic MDS*, which is short for *Least Squares Euclidean Metric MDS* (LSEM-MDS). It is defined as MDS with the following characteristics.

1. The space is a finite dimensional linear space.
2. The metric is Euclidean.
3. The dissimilarities are numerical, symmetric, and non-negative.
4. The loss function is a weighted sum of squares of the *residuals*, which are the differences between dissimilarities and distances.
5. Weights are numerical, symmetric, and non-negative.

6. Self-dissimilarities are zero and the corresponding terms in the loss function have weight zero.

The *loss function* we use is called *stress*. It was first explicitly introduced in MDS by Kruskal (1964a) and Kruskal (1964b). We define stress in a slightly different way, because we want to be consistent over the whole range of the smacof versions and implementations. In smacof stress is the real-valued function σ , defined on the space $\mathbb{R}^{n \times p}$ of configurations, as

$$\sigma(X) := \frac{\sum \sum_{1 \leq i < j \leq n} w_{ij}(\delta_{ij} - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} w_{ij}\delta_{ij}^2}. \quad (1.1)$$

We use the notation $\sum \sum_{1 \leq i < j \leq n}$ for summation over the upper-diagonal elements of a matrix. Note also that we use $:=$ for definitions, i.e. for concepts and symbols that are not standard mathematical usage, when they occur for the first time in this book.

In definition (1.1) we use the following objects and symbols.

1. $W = \{w_{ij}\}$ is a symmetric, non-negative, and hollow matrix of *weights*, where ‘hollow’ means zero diagonal.
2. $\Delta = \{\delta_{ij}\}$ is a symmetric, non-negative, and hollow matrix of *dissimilarities*.
3. X is an $n \times p$ *configuration*, containing coordinates of n *points* in p dimensions.
4. $D(X) = \{d_{ij}(X)\}$ is a symmetric, non-negative, and hollow matrix of *Euclidean distances* between the n points in X . Thus $d_{ij}(X) := \sqrt{\sum_{s=1}^p (x_{is} - x_{js})^2}$.

Observe that we distinguish the linear space $\mathbb{R}^{n \times p}$ of $n \times p$ matrices from the linear space \mathbb{R}^{np} of np element vectors. The two spaces are isomorphic, and connected by the vec operator and its inverse. Some formulas in MDS are more easily expressed in \mathbb{R}^{np} , but we prefer to work in the more natural space $\mathbb{R}^{n \times p}$ of configurations.

The function D , which computes the distance matrix $D(X)$ from a configuration X , is matrix-valued. It maps the $n \times p$ -dimensional linear space $\mathbb{R}^{n \times p}$

of configuration matrices into the set $D(\mathbb{R}^{n \times p})$ of Euclidean distance matrices between n points in \mathbb{R}^p , which is a subset of the convex cone of hollow, symmetric, non-negative matrices in the linear space $\mathbb{R}^{n \times n}$.

In basic MDS the weights and dissimilarities are given numbers, and we minimize stress over all $n \times p$ configurations X . Note that the *dimensionality* p is also supposed to be known beforehand, and that MDS in p dimensions is different from MDS in $q \neq p$ dimensions. We sometimes emphasize this by writing $pMDS$, which shows that we will map the points into p -dimensional space.

Two boundary cases that will interest us are *Unidimensional Scaling* or *UDS*, where $p = 1$, and *Full-dimensional Scaling* or *FDS*, where $p = n$. Thus UDS is 1MDS and FDS is nMDS. Most actual MDS applications in the sciences use 1MDS, 2MDS or 3MDS, because configurations in one, two, or three dimensions can easily be plotted with standard graphics tools. Thus MDS is not primarily a tool to test hypotheses about dimensionality and the find meaningful dimensions. It is a mostly a mapping tool for data reduction, to graphically find interesting aspects of dissimilarity matrices. The projections on the dimensions are very much ignored, it is the configuration of points that is the interesting outcome. This distinguishes MDS from, for example, factor analysis. Exceptions are applications of MDS in the conformation of molecules, in genetic mapping along the chromosome, in archeological seriation, and in geographic applications. There the dimensionality and general structure of the configuration are given by prior knowledge, we just do not the precise location and distances of the points. For more discussion of the different uses of MDS we refer to De Leeuw and Heiser (1982).

1.2.1 Kruskal's stress

Definition (1.1) differs from Kruskal's original stress in at least three ways: in Kruskal's use of the square root, in our use of weights, and in our different approach to normalization.

Remember that Kruskal's definition was intended for ordinal multidimensional scaling only. He first defined *raw stress* as

$$\sigma^*(X) := \sum_{1 \leq i < j \leq n} (\hat{d}_{ij} - d_{ij}(X))^2, \quad (1.2)$$

where the \hat{d}_{ij} is a set of numbers monotone with the dissimilarities.

To simplify the discussion, we delay the precise definition of \hat{d} , for a little while. (Kruskal (1964a), p. 8)

Kruskal then mentions that raw stress satisfies $\sigma^*(\alpha X) = \alpha^2 \sigma^*(X)$, which is clearly undesirable because the size of the configuration should not influence the quality of the fit.

An obvious way to cure this defect in the raw stress is to divide it by a $>$ scaling factor, that is, a quantity which has the same quadratic dependence on the scale of the configuration that raw stress does. (Kruskal (1964a), p. 8).

By the way, although the precise definition of \hat{D} has been delayed, the uniform stretching result already assumes that if we multiply D by α then \hat{D} also gets multiplied by α . Thus it sort of gives away that \hat{D} is also a function of X , at least of the scale of X .

$$\sigma^*(X) := \sum_{1 \leq i < j \leq n} (\hat{d}_{ij} - d_{ij}(X))^2,$$

For the normalization of raw stress Kruskal chooses

$$\tau^*(X) := \sum_{1 \leq i < j \leq n} d_{ij}^2(X), \quad (1.3)$$

Finally, it is desirable to use the square root of this expression, which is analogous to choosing the standard deviation in place of the variance. (Kruskal (1964a), p. 9)

Thus Kruskal's normalized loss function for ordinal MDS becomes

$$\sigma(X) := \sqrt{\frac{\sigma^*(X)}{\tau^*(X)}} = \sqrt{\frac{\sum \sum_{1 \leq i < j \leq n} (\hat{d}_{ij} - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} d_{ij}^2(X)}}. \quad (1.4)$$

At this point in Kruskal (1964a) the definition of \hat{D} still hangs in the air, although we know that the \hat{D} are monotone with Δ , and that multiplying X by a constant will multiply both $D(X)$ and \hat{D} by the same constant. Matters are clarified right after the definition of stress.

Now it is easy to define the \hat{d}_{ij} . They are the numbers which minimize σ (or equivalently, σ^* subject to the monotonicity constraints. (Kruskal (1964a), p. 9)

Thus, actually, raw stress is the minimum over the pseudo-distance matrices Ω in \mathfrak{D} , the set of all monotone transformations of the dissimilarities.

$$\sigma^*(X) := \min_{\Omega \in \mathfrak{D}} \sum_{1 \leq i < j \leq n} (\omega_{ij} - d_{ij}(X))^2, \quad (1.5)$$

and \hat{D} is the minimizer, which is now clearly a function of X ,

$$\hat{D}(X) := \operatorname{argmin}_{\Omega \in \mathfrak{D}} \sum_{1 \leq i < j \leq n} (\omega_{ij} - d_{ij}(X))^2. \quad (1.6)$$

So, finally,

$$\sigma(X) := \min_{\Omega \in \mathfrak{D}} \sqrt{\frac{\sigma^*(X)}{\tau^*(X)}} = \sqrt{\frac{\sum \sum_{1 \leq i < j \leq n} (\omega_{ij} - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} d_{ij}^2(X)}}. \quad (1.7)$$

We have paid so much attention to Kruskal's original definition, because the choices made there will play a role in the normalization discussion in the ordinal scaling chapter (section 8.4.1), in the comparison of Kruskal's and Guttman's approach to OMDS (sections 11.3 and 11.4), and in our discussions about the differences between Kruskal's stress (1.7) and smacof's stress (1.1) in the next three sections of this chapter.

1.2.1.1 Square root

Let's discuss the square root first. Using it or not using it does not make a difference for the minimization. Using the square root does give a more

sensible root-mean-square scale, in which stress is homogeneous of degree one, instead of two. But I do not want to compute all those unnecessary square roots in my algorithms, and I do not want to drag them along through my derivations. Moreover it potentially causes problems with differentiability at those X where $\sigma(X)$ is zero. Thus, throughout the book, we do not use the square root in our formulas and derivations. In fact, we do not even use it in our computer programs, except at the very last moment when we return the final stress after the algorithm has completed.

1.2.1.2 Weights

There were no weights $W = \{w_{ij}\}$ in the original definition of stress by Kruskal (1964a), and neither are they there in most of the basic later contributions to MDS by Guttman, Lingoes, Roskam, Ramsay, or Young. We will use weights throughout the book, because they have various interesting applications within basic MDS, without unduly complicating the derivations and computations. In Groenen and Van de Velden (2016), section 6, the various uses of weights in the stress loss function are enumerated. They generously, but correctly, attribute the consistent use of weights in MDS to me. My turn. I quote from their paper:

1. Handling missing data is done by specifying $w_{ij} = 0$ for missings and 1 otherwise thereby ignoring the error corresponding to the missing dissimilarities.
2. Correcting for nonuniform distributions of the dissimilarities to avoid dominance of the most frequently occurring dissimilarities.
3. Mimicking alternative fit functions for MDS by minimizing Stress with w_{ij} being a function of the dissimilarities.
4. Using a power of the dissimilarities to emphasize the fitting of either large or small dissimilarities.
5. Special patterns of weights for specific models.
6. Using a specific choice of weights to avoid nonuniqueness.

In some situations, for example for huge data sets, it is computationally convenient, or even necessary, to minimize the influence of the weights on the computations. We can use *majorization* to turn the problem from a weighted

least squares problem to an iterative unweighted least squares problem. The technique is discussed in detail in chapter 25, section 6.4.7.

1.2.1.3 Normalization

This section deals with a rather trivial problem, which has however caused problems in various stages of smacof's 45-year development history. Because the problem is trivial, and the choices that must be made are to a large extent arbitrary, it has been overlooked and somewhat neglected.

Multiplying all weights by a constant does not change the value of stress, and consequently the minimization problem remains exactly the same. No matter how we scale W the numerator of stress is the weighted mean square of the *residuals* $\delta_{ij} - d_{ij}(X)$ and the denominator is the weighted mean square of the dissimilarities. So there is no need to normalize the weights, and we leave them just as the user inputs them (or as they are chosen by default).

We do scale the dissimilarities, however. It is clear that if we multiply the dissimilarities by a constant, then the optimal approximating distances $D(X)$ and the optimal configuration X will be multiplied by the same constant. Consequently we always scale dissimilarities by $\sum \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}^2 = 1$. This simplifies our formulas and makes them look better. It presupposes, of course, that $w_{ij} \delta_{ij} \neq 0$ for at least one $i \neq j$, which we will happily assume in the sequel. Using normalized dissimilarities amounts to the same as minimizing stress defined as

$$\sigma(X) = \frac{\sum \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij}^2 - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}^2}. \quad (1.8)$$

This is useful to remember when we discuss the various normalizations of non-metric MDS in chapter 8, section 8.4.1.

In *output* using the scaling $\sum \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}^2 = 1$ has some disadvantages. There are $\binom{n}{2}$ weights, and, say, M non-zero weights. The summation in the numerator and denominator of #ref(eq:stressall) is really over M terms only. If n is at all large the scaled dissimilarities, and consequently the distances and configuration, will become very small. Thus, in actual computation, we scale our dissimilarities as $\sum \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}^2 = \binom{n}{2}$. So, we scale our

dissimilarities to one in formulas and to $\binom{n}{2}$ in computations. Thus the computed stress will be

$$\begin{aligned}\sigma(X) &= \frac{1}{\binom{n}{2}} \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij}^2 - d_{ij}(X))^2 = \\ &= 1 - 2 \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} d_{ij}(X) + \sum_{1 \leq i < j \leq n} w_{ij} d_{ij}^2(X).\end{aligned}\tag{1.9}$$

... we shall find ourselves doing arithmetic with dissimilarities. This we must not do, because we are committed to using only the rank ordering of the dissimilarities. (a, p 6-7)

1.2.2 Data Asymmetry

The non-basic situation in which there are asymmetric weights and/or dissimilarities in basic MDS is analyzed in De Leeuw (1977a), although it is just standard linear least squares projection theory.

For all $i \neq j$ let $\underline{w}_{ij} = \frac{1}{2}(w_{ij} + w_{ji})$ and $\underline{\delta}_{ij} = (w_{ij}\delta_{ij} + w_{ji}\delta_{ji})/(w_{ij} + w_{ji})$. Then

$$\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} d_{ij}^2(X) = \sum_{1 \leq i < j \leq n} \underline{w}_{ij} d_{ij}^2(X),\tag{1.10}$$

and

$$\sum_{i=1}^n \sum_{j=1}^n w_{ij} \delta_{ij} d_{ij}(X) = 2 \sum_{1 \leq i < j \leq n} w_{ij} \underline{\delta}_{ij} d_{ij}(X).\tag{1.11}$$

Thus

$$\begin{aligned}\sigma(X) &= 1 - 2 \sum_{1 \leq j < i \leq n} \underline{w}_{ij} \underline{\delta}_{ij} d_{ij}(X) + \sum_{1 \leq j < i \leq n} \underline{w}_{ij} d_{ij}^2(X) \\ &= (1 - \sum_{1 \leq j < i \leq n} \underline{w}_{ij} \delta_{ij}^2) + \sum_{1 \leq j < i \leq n} \underline{w}_{ij} (\underline{\delta}_{ij} - d_{ij}(X))^2\end{aligned}\tag{1.12}$$

1.2.3 Local and Global

In this book will study both the properties of the stress loss function and a some of its generalizations, and the various ways to minimize these loss functions over configurations (and sometimes over transformations of the dissimilarities as well).

Emphasis local minima

1.3 Generalizations

In basic MDS our goal is to compute both $\min_X \sigma(X)$ and $\text{Argmin}_X \sigma(X)$, where $\sigma(X)$ is defined as (1.1), and where we minimize over all configurations in $\mathbb{R}^{n \times p}$. Note we use the notation $\text{Argmin}_{x \in X} f(x)$ for the set of minimizers of f over X . Thus $z \in \text{Argmin}_{x \in X} f(x)$ means z minimizes f over X . If it is clear from theory that the minimum is necessarily unique, we use argmin instead of Argmin .

The most important generalizations of basic MDS we will study in later chapters of this book are discussed briefly in the following sections.

1.3.1 Non-metric

Basic MDS studies *Metric Multidimensional Scaling* or *MMDS*, in which dissimilarities are either known or missing. In chapter 8 we relax this assumption. Dissimilarities may be partly known, for example we may know they are in some interval, we may only know their order, or we may know them up to some smooth transformation. MDS with partly known dissimilarities is *Non-metric Multidimensional Scaling* or *NMDS*. Completely unknown (missing) is an exception, because we can just handle this in basic MDS by setting the corresponding weights equal to zero.

In NMDS we minimize stress over all configurations, but also over the unknown dissimilarities. What we know about them (the interval they are in, the transformations that are allowed, the order they are in) defines a subset of the space of non-negative, hollow, and symmetric matrices. Any matrix in that subset is a matrix of what Takane, Young, and De Leeuw (1977) call

disparities, i.e. imputed dissimilarities. The imputation provides the missing information and transforms the non-numerical information we have about the dissimilarities into a numerical matrix of disparities. Clearly this is an *optimistic imputation*, in the sense that it chooses from the set of admissible disparities to minimize stress (for a given configuration).

One more terminological point. Often *non-metric* is reserved for ordinal MDS, in which only know the (partial or complete) order of the dissimilarities. Allowing linear or polynomial transformations of the dissimilarities, or estimating an additive constant, is not supposed to be non-metric. There is something to be said for that. Maybe it makes sense to distinguish non-metric *in the wide sense* (in which stress must be minimized over both X and Δ) and *non-metric in the narrow sense* in which the set of admissible disparities is defined by linear inequalities. Nonmetric in the narrow sense will also be called *ordinal MDS* or *OMDS*.

It is perhaps useful to remember that Kruskal (1964a) introduced explicit loss functions in MDS to put the somewhat heuristic NMDS techniques of Shepard (1962a) onto a firm mathematical and computational foundation. Thus, more or less from the beginning of iterative least squares MDS, there was a focus on non-metric rather than metric MDS, and this actually contributed a great deal to the magic and success of the technique. In this book most of the results are derived for basic MDS, which is metric MDS, with non-metric MDS a relatively straightforward extension not discussed until chapter 8. So, at least initially, we take the numerical values of the dissimilarities seriously, as do Torgerson (1958) and Shepard (1962a), Shepard (1962b). It may be the case that in the social and behavioural sciences only the ordinal information in the dissimilarities is reliable and useful. But, since 1964, MDS has also been applied in molecular conformation, chemometrics, genetic sequencing, archaeological seriation, and in network design and location analysis. In these areas the numerical information in the dissimilarities is usually meaningful and should not be thrown out right away. Also, the use of the Shepard plot, with dissimilarities on the horizontal axis and fitted distances on the vertical axis, suggests there is more to dissimilarities than just their rank order.

1.3.2 fstress

Instead of defining the residuals in the least squares loss function as $\delta_{ij} - d_{ij}(X)$ chapter 20 discusses the more general cases where the residuals are $f(\delta_{ij}) - f(d_{ij}(X))$ for some known non-negative increasing function f . This defines the *fstress* loss function.

If $f(x) = x^r$ with $r > 0$ then *fstress* is called *rstress*. Thus stress is *rstress* with $r = 1$, also written as *1stress* or σ_1 . In more detail we

1.3.3 Constraints

Instead of minimizing stress over all X in $\mathbb{R}^{n \times p}$ we will look in chapter 15 at various generalizations where minimization is over a subset \otimes of $\mathbb{R}^{n \times p}$. This is often called *Constrained Multidimensional Scaling* or *CMDS*.

exp vs conf FA

1.3.4 Replications

ind diff {#chindif}

1.3.5 Distance Asymmetry

We have seen in section 1.2.2 of this chapter that in basic MDS the assumption that W and Δ are symmetric and hollow can be made without loss of generality. The simple partitioning which proved this was based on the fact that $D(X)$ is symmetric and hollow. By the way, the assumption that W and D are non-negative cannot be made without loss of generality, as we will see below.

In 16 we relax the assumption that $D(X)$ is symmetric (still requiring it to be non-negative and hollow). This could be called *Asymmetric MDS*, or *AMDS*. I was reluctant at first to include this chapter, because asymmetric distances do not exist. And certainly are not Euclidean distances, so they are not covered by the title of this book. But as long as we stay close to Euclidean distances, least squares, and the smacof approach, I now feel reasonably confident the chapter is not too much of a foreign body.

When Kruskal introduced gradient based methods to minimize stress he also discussed the possibility to use Minkovski metrics other than the Euclidean metric. This certainly was part of the appeal of the new methods, in fact it seemed as if the gradient methods made it possible to use any distance function at all. This initial feeling of empowerment was somewhat naive, because it ignored the seriousness of the local minimum problem, the combinatorial nature of one-dimensional scaling, the problems with nonmetric unfolding, and the problematic nature of gradient methods if the distances are not everywhere differentiable. All these complications will be discussed later in this book. But it made me decide to ignore Minkovski distances (and hyperbolic and elliptic non-Euclidean distances), because life with stress is complicated and challenging enough as it is.

Chapter 2

Properties of Stress

2.1 Notation

The notation used in the smacof approach to MDS first appeared in De Leeuw (1977a), and was subsequently used in several of the related smacof references, such as De Leeuw and Heiser (1982), De Leeuw (1988), chapter 8 of Borg and Groenen (2005), and De Leeuw and Mair (2009).

2.1.1 Expanding

We expand stress by writing out the squares of the residuals and then summing. Define

$$\eta_\delta^2 := \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}^2, \quad (2.1)$$

$$\rho(X) := \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} d_{ij}(X), \quad (2.2)$$

$$\eta^2(X) := \sum_{1 \leq i < j \leq n} w_{ij} d_{ij}^2(X). \quad (2.3)$$

More precisely, using conditional summation,

$$\rho(X) := \sum_{1 \leq i < j \leq n} \{w_{ij}\delta_{ij}d_{ij}(X) \mid w_{ij}\delta_{ij} > 0\}, \quad (2.4)$$

$$\eta^2(X) := \sum_{1 \leq i < j \leq n} \{w_{ij}d_{ij}^2(X) \mid w_{ij} > 0\}. \quad (2.5)$$

Remember that we have normalized by $\eta_\delta^2 = 1$. With our newly defined functions ρ and η^2 we can write stress as

$$\sigma(X) = 1 - 2\rho(X) + \eta^2(X) = (1 - \rho(X))^2 + (\eta^2(X) - \rho^2(X)). \quad (2.6)$$

The CS inequality implies that for all X

$$\rho(X) = \sum_{1 \leq i < j \leq n} w_{ij}\delta_{ij}d_{ij}(X) \leq \eta_\delta\eta(X) = \eta(X), \quad (2.7)$$

and thus, from (2.6),

$$\sigma(X) \geq (1 - \eta(X))^2, \quad (2.8)$$

$$\sigma(X) \geq (1 - \rho(X))^2. \quad (2.9)$$

2.1.2 Matrix Expressions

Using matrix notation allows us to arrive at compact expressions, which suggest various mathematical and computational shortcuts. In order to use matrix notation for distances we use the difference matrices A_{ij} , discussed in section 25.2.3.

We start with $d_{ij}^2(X) = \text{tr } X' A_{ij} X$. Define

$$V := \sum_{1 \leq i < j \leq n} \sum w_{ij} A_{ij}, \quad (2.10)$$

so that

$$\eta^2(X) = \text{tr } X'VX. \quad (2.11)$$

The matrix V has off-diagonal elements equal to $-w_{ij}$ and diagonal elements $v_{ii} = \sum_{j \neq i} w_{ij}$. It is symmetric, positive semi-definite, and doubly-centered. Thus it is singular, because $Ve = 0$.

Section 25.2.12 discusses the rank of V in more detail and defines reducibility of W . If W is reducible the MDS problem separates into a number of smaller independent MDS problems. We will assume in the sequel, without loss of generality, that this does not occur, and that consequently W is irreducible.

The Moore-Penrose inverse, defined in section 25.2.10 of chapter 25, of V is

$$V^{-1} = (V + \frac{ee'}{n})^{-1} - \frac{ee'}{n}. \quad (2.12)$$

If all weights are one, then $V = nJ$ and $V^{-1} = \frac{1}{n}J$, with J the centering matrix $I - \frac{1}{n}ee'$ from section 25.2.2.

Writing out $\rho(X)$ from (2.2) in matrix form is a bit more complicated. Define

$$r_{ij}(X) := \begin{cases} 0 & \text{if } d_{ij}(X) = 0, \\ \frac{\delta_{ij}}{d_{ij}(X)} & \text{if } d_{ij}(X) > 0, \end{cases} \quad (2.13)$$

and

$$B(X) := \sum_{1 \leq i < j \leq n} w_{ij} r_{ij}(X) A_{ij}. \quad (2.14)$$

Then we have

$$\rho(X) = \text{tr } X'B(X)X. \quad (2.15)$$

Just like V , the matrix-valued function B is symmetric, positive-semidefinite, and doubly-centered. If all dissimilarities and distances are positive then irreducibility of W implies that the rank of $B(X)$ is equal to $n - 1$. Note that if $\delta_{ij} = d_{ij}(X) > 0$ for all i, j (perfect fit), then the r_{ij} from (2.13) are all equal to one, and $B(X) = V$.

In (2.13) we have set $r_{ij}(X) = 0$ if $d_{ij}(X) = 0$. This is arbitrary. Since $b_{ij}(X) = r_{ij}(X)$ if $d_{ij}(X) = 0$ we get a different matrix $B(X)$ if we choose to set, say, $r_{ij}(X) = 1$ or $r_{ij}(X) = \delta_{ij}$ whenever $d_{ij}(X) = 0$. But

$$B(X)X = \sum_{1 \leq i < j \leq n} w_{ij} r_{ij}(X)(e_i - e_j)(x_i - x_j)' \quad (2.16)$$

is the same, no matter how we choose $r_{ij}(X)$ if $d_{ij}(X) = 0$. And, consequently, so is $\rho(X) = \text{tr } X'B(X)X$.

We now see, from equation (2.6), that

$$\sigma(X) = 1 - 2 \text{ tr } X^T B(X)X + \text{tr } X^T V X. \quad (2.17)$$

2.2 Global Properties

2.2.1 Boundedness

Stress is a sum of squares, and thus it is non-negative, i.e. bounded below by zero. Because $\sigma(\lambda X) = 1 - \lambda\rho(X) + \frac{1}{2}\lambda^2\eta^2(X)$ we see that σ is unbounded above. In fact, it is an unbounded convex quadratic on each ray through the origin.

2.2.2 Invariance

Stress only depends on the distances between the points in the configuration, and thus it is invariant under rigid geometrical transformations (rotations, reflections, and translations). Thus $\sigma(XK) = \sigma(X)$ for all K with $K'K = KK' = I$. Also $\sigma(X + eu') = \sigma(X)$ for all $u \in \mathbb{R}^p$. And stress is even, i.e. $\sigma(-X) = \sigma(X)$.

It follows directly that the minimizer of stress, if it exists, cannot possibly be unique. Whatever the value at a minimum, it is shared by all rigid transformations of the configuration.

It also follows from translational invariance that we can minimize stress over the $p(n - 1)$ dimensional subspace of $\mathbb{R}^{n \times p}$ of all $n \times p$ matrices which are

centered, i.e. have $e'X = 0$. Or over all matrices which have the first row x_1 equal to zero. Rotational invariance implies we can also require without loss of generality that X is orthogonal, i.e. that $X'X$ is diagonal.

2.2.3 Continuity

Both d_{ij} and d_{ij}^2 are continuous on $\mathbb{R}^{n \times p}$. This follows easily from the formula for Euclidean distance, together with the composition rules for continuous functions and the continuity of the square and the square root.

Continuity also follows from the continuity of the norm, because $d_{ij}(X) = \|X'(e_i - e_j)\|$, and it follows from the general result that in any metric space $\mathcal{X} = \langle X, d \rangle$ the metric d is continuous on $\mathcal{X} \otimes \mathcal{X}$ with the product topology.

Convexity, discussed in section 2.3 of this chapter, also implies continuity.

2.2.4 Coercivity

Stress is not a convex function of the configuration. But it is bowl shaped around the origin, in a way we are going to make more precise. First a definition: A function is *coercive* if for every sequence $\{X_k\}$ with $\lim_{k \rightarrow \infty} \|X_k\| = \infty$ we also have $\lim_{k \rightarrow \infty} \sigma(X_k) = +\infty$.

It is easy to see that σ is coercive. We know that $\sigma(X) = 1 - 2\rho(X) + \eta^2(X)$. Now the CS inequality gives $\rho(X) \leq \eta(X)$, and thus $\sigma(X) \geq (1 - \eta(X))^2$, which shows that stress is coercive.

It follows from coercivity (Ortega and Rheinboldt (1970), section 4.3) that all level sets of stress $\mathcal{L}_s := \{X \mid \sigma(X) = s\}$ are compact, and that there is at least one configuration for which the global minimum of stress is attained.

2.3 Convexity

Stress is definitely not a convex function of the configuration. Although if it actually was convex there would not be as much motivation for writing this book. Nevertheless convexity still play an important part in our analysis of MDS, ever since De Leeuw (1977a).

2.3.1 Distances

The convexity in the MDS problem comes from the convexity of the distance and the squared distance.

By the triangle inequality

$$d_{ij}(\lambda X + (1-\lambda)Y) = \|(\lambda X + (1-\lambda)Y)'(e_i - e_j)\| = \|\lambda X'(e_i - e_j) + (1-\lambda)Y'(e_i - e_j)\| \leq \lambda d_{ij}(X) + (1-\lambda)d_{ij}(Y)$$

If g is convex and f is convex and increasing then $f \circ g$ is convex.

Because g is convex $g(\lambda x + (1 - \lambda)y) \leq \lambda g(x) + (1 - \lambda)g(y)$. Because f is increasing and convex

$$f(g(\lambda x + (1 - \lambda)y)) \leq f(\lambda g(x) + (1 - \lambda)g(y)) \leq \lambda f(g(x)) + (1 - \lambda)f(g(y)).$$

First, for $0 \leq \lambda \leq 1$,

$$d_{ij}^2(\lambda X + (1 - \lambda)Y) = \lambda^2 d_{ij}^2(X) + (1 - \lambda)^2 d_{ij}^2(Y) + 2\lambda(1 - \lambda)(x_i - x_j)'(y_i - y_j) \quad (2.18)$$

Thus, using 25.1,

$$2\lambda(1 - \lambda)(x_i - x_j)'(y_i - y_j) \leq \lambda(1 - \lambda)(d_{ij}^2(X) + d_{ij}^2(Y)). \quad (2.19)$$

Combining (2.18) and (2.19) proves convexity of the squared distance.

Now use equation (2.18) and the CS inequality in the form $(x_i - x_j)'(y_i - y_j) \leq d_{ij}(X)d_{ij}(Y)$. This gives

$$d_{ij}^2(\lambda X + (1 - \lambda)Y) \leq (\lambda d_{ij}(X) + (1 - \lambda)d_{ij}(Y))^2. \quad (2.20)$$

Taking square roots on both sides of equation (2.20) proves convexity of the distance.

2.3.2 DC Functions

In basic MDS

1. ρ is a non-negative convex function, homogeneous of degree one.
2. η^2 is a non-negative convex quadratic form, homogeneous of degree two.
3. σ is a non-negative difference of two convex functions.

This follows because η^2 is a weighted sum of squared distances and ρ is a weighted sum of distances, both with non-negative coefficients, and thus they are both convex.

Real-valued functions that are differences of two convex functions are also known as a *DC functions* or *delta-convex function*. DC functions are important in optimization, especially in non-convex and global optimization. For excellent reviews of the various properties of DC functions, see Hiriart-Urruty (1988), Vesely and Zajicek (1989), Tuy (1998), chapter 3, and Bacak and Borwein (2011).

It follows from the general properties of convex and DC functions that σ is both uniformly continuous and locally Lipschitz, in fact Lipschitz on each compact subset of $\mathbb{R}^{n \times p}$ (Rockafellar (1970), theorem 10.4). The fact that σ is only locally Lipschitz is due entirely to the quadratic part η^2 , because ρ is globally Lipschitz.

all twice differentiable are DC

Theorem 2.1. ρ is Lipschitz, with Lipschitz constant $\sum \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}$.

In the first place

$$|\rho(X) - \rho(Y)| = \left| \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} (d_{ij}(X) - d_{ij}(Y)) \right| \leq \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} |(d_{ij}(X) - d_{ij}(Y))|,$$

and, using the reverse triangle inequality,

$$|d_{ij}(X) - d_{ij}(Y)| = ||\|X^T(e_i - e_j)\| - \|Y^T(e_i - e_j)\|| \leq \|(X - Y)^T(e_i - e_j)\| \leq \|X - Y\|.$$

On the other hand η^2 is only locally Lipschitz.

2.3.3 Subdifferentials

To compute the subdifferential (defined in chapter 25 section 25.5.3) of ρ we represent distances as maximum functions over the unit ball \mathfrak{S}_p in \mathbb{R}^p .

$$d_{ij}(X) = \max_{\|z\| \leq 1} \operatorname{tr} z' X' (e_i - e_j). \quad (2.21)$$

If $d_{ij}(X) > 0$ the maximum is attained at the unique z equal to $(x_i - x_j)/d_{ij}(X)$. Thus $\partial d_{ij}(X)$ is the singleton containing this z . If $d_{ij}(X) = 0$ we find

$$\partial d_{ij}(X) = \{Y \in \mathbb{R}^{n \times p} | Y = (e_i - e_j)z', \|z\| \leq 1\} \quad (2.22)$$

Thus $Y \in \partial d_{ij}(X)$ if and only if Y is zero, except for $y_i = z$ and $y_j = -z$, where z is any vector in \mathfrak{S}_p . By the sum rule for subdifferentials

$$\partial \rho(X) = \sum_{d_{ij}(X) > 0} \sum w_{ij} \delta_{ij} \frac{(e_i - e_j)(x_i - x_j)'}{d_{ij}(X)} + \sum_{d_{ij}(X) = 0} \sum w_{ij} \delta_{ij} (e_i - e_j) z'_{ij}, \quad (2.23)$$

where again the z_{ij} are arbitrary vectors in \mathfrak{S}_p .

2.3.4 Negative Dissimilarities

There are perverse situations in which some weights and/or dissimilarities are negative (Heiser (1991)). Define $w_{ij}^+ := \max(w_{ij}, 0)$ and $w_{ij}^- := -\min(w_{ij}, 0)$. Thus both w_{ij}^+ and w_{ij}^- are non-negative, and $w_{ij} = w_{ij}^+ - w_{ij}^-$. Make the same decomposition of the δ_{ij} .

Then

$$\rho(X) = \sum (w_{ij}^+ \delta_{ij}^+ + w_{ij}^- \delta_{ij}^-) d_{ij}(X) - \sum (w_{ij}^+ \delta_{ij}^- + w_{ij}^- \delta_{ij}^+) d_{ij}(X), \quad (2.24)$$

and

$$\eta^2(X) = \sum w_{ij}^+ d_{ij}^2(X) - \sum w_{ij}^- d_{ij}^2(X). \quad (2.25)$$

Note that both ρ and η^2 are no longer convex, but both are DC, and consequently so is σ .

A bit more

2.4 Guttman Transform

The Guttman Transform is named to honor the contribution of Louis Guttman to non-metric MDS (mainly, but by no means exclusively, in Guttman (1968)). Guttman introduced the transform in a slightly different way, and partly for different reasons. In chapter 6 we shall see that the Guttman Transform plays a major role in defining and understanding the smacof algorithm.

The *Guttman Transform* of a configuration X (for given weights and dissimilarities) is defined as

$$\mathfrak{G}(X) := V^{-1}B(X)X, \quad (2.26)$$

which is simply equal to $\mathfrak{G}(X) = n^{-1}B(X)X$ if all weights are equal.

What we have called $B(X)$ is what Guttman calls the *correction matrix* or C-matrix (see De Leeuw and Heiser (1977) for a comparison).

Completing the square in equation (2.17) gives

$$\sigma(X) = 1 + \eta^2(X - \mathfrak{G}(X)) - \eta^2(\mathfrak{G}(X)), \quad (2.27)$$

which shows that

$$1 - \eta^2(\mathfrak{G}(X)) \leq \sigma(X) \leq 1 + \eta^2(X - \mathfrak{G}(X)). \quad (2.28)$$

Note that it follows from (2.28) that $\sigma(X) \geq 1 - \eta^2(\mathfrak{G}(X))$, with equality if and only if $X = \mathfrak{G}(X)$.

The Guttman transform is *self-scaling* (a.k.a. homogeneous of degree zero) because $\mathfrak{G}(\alpha X) = \mathfrak{G}(X)$ for all $-\infty < \alpha < +\infty$. With our definition (2.14) of $B(X)$ we also have $\mathfrak{G}(0) = 0$. The Guttman transform is also *self-centering*, because $\mathfrak{G}(X + e\mu') = \mathfrak{G}(X)$ for all $\mu \in \mathbb{R}^p$.

We already know, from the CS inequality, that

$$\rho(X) \leq \eta(X). \quad (2.29)$$

With the Guttman transform in hand we can apply CS once more, and find

$$\rho(X) = \text{tr } X' B(X) X = \text{tr } X' V \mathfrak{G}(X) \leq \eta(X) \eta(\mathfrak{G}(X)) \quad (2.30)$$

Note that the Guttman transform is bounded. In fact, using the Euclidean norm throughout,

$$\mathfrak{G}(X) \leq \|V^+\| \|B(X)X\|$$

Now

$$B(X)X = \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} \frac{x_i - x_j}{d_{ij}(X)} (e_i - e_j),$$

and thus

$$\|B(X)X\| \leq \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} \left\| \frac{x_i - x_j}{d_{ij}(X)} \right\| \|e_i - e_j\| = \sqrt{2} \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij},$$

and

$$\|\mathfrak{G}(X)\| \leq \sqrt{2} \|V^+\| \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}.$$

In fact

$$B(X)X - B(Y)Y = \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} \left\{ \frac{x_i - x_j}{d_{ij}(X)} - \frac{y_i - y_j}{d_{ij}(Y)} \right\} (e_i - e_j),$$

and thus

$$\|\mathfrak{G}(X) - \mathfrak{G}(Y)\| \leq 2\|V^+\| \sum_{1 \leq i < j \leq n} w_{ij}\delta_{ij},$$

and thus the Guttman transform is Lipschitz.

2.5 Differentiability

The fact that $d_{ij}(X)$ can be zero creates some problems with differentiability. Clearly $d_{ij}^2(X) = \text{tr } X' A_{ij} X$ is quadratic, and thus infinitely many times differentiable. But $d_{ij}(X) = \sqrt{\text{tr } X' A_{ij} X}$ is not differentiable at points where $x_i = x_j$ and thus $d_{ij}(X) = 0$, because the square root is not differentiable at zero. For stress this means that $\eta^2(X)$ is differentiable but $\rho(X)$ is not if $d_{ij}(X) = 0$ for some i and j with $w_{ij}\delta_{ij} > 0$.

These problems have been largely ignored in the MDS literature, and there are indeed reasons why they are not of great practical importance (see section 2.6.2 of this chapter), at least not in basic MDS. But for reasons of completeness we discuss zero distances in some detail. Historically the complications caused by zero distances were one of the reasons why I switched from differentiability to convexity in De Leeuw (1977a) and from derivatives to directional derivatives in De Leeuw (1984c). It turned out that the important characteristics of the smacof algorithm, and several important aspects of stress surfaces, were better described by inequalities than by equations.

2.5.1 Regularity

A configuration X is *regular* if $d_{ij}(X) > 0$ for all $i \neq j$ with $w_{ij}\delta_{ij} > 0$. I apologize for yet another instance of overloading the word “regular.”

2.5.1.1 Distances

If X is regular then all d_{ij} are differentiable at X , in fact infinitely many times differentiable. Remembering that $d_{ij}(X) = \sqrt{\text{tr } X' A_{ij} X}$ we see that

$$\mathcal{D}d_{ij}(X) = \frac{1}{d_{ij}(X)} A_{ij} X = (e_i - e_j) \frac{(x_i - x_j)'}{d_{ij}(X)}. \quad (2.31)$$

If we collect the partials in an $n \times p$ matrix, then row i matrix of this matrix is $(x_i - x_j)/\|x_i - x_j\|$ and row j is $(x_j - x_i)/\|x_i - x_j\|$. The rest of the matrix are zeroes, and the matrix is column-centered. Both non-zero rows have norm one. It also follows that the sum of squares of the elements of $\mathcal{D}d_{ij}(X)$ is equal to two, and thus all partial derivatives are less than or equal to $\sqrt{2}$.

For the quadratic form defined by the second derivatives we find

$$\mathcal{D}^2 d_{ij}(X)(Y, Y) = \frac{1}{d_{ij}(X)} \left\{ \text{tr } Y' A_{ij} Y - \frac{(\text{tr } Y' A_{ij} X)^2}{d_{ij}^2(X)} \right\}. \quad (2.32)$$

From the CS inequality we see that $\mathcal{D}^2 d_{ij}(X)(Y, Y) \geq 0$, as we would expect from any decent twice-differentiable convex function.

Of course $\mathcal{D}d_{ij}^2(X) = 2A_{ij}X$, and $\mathcal{D}^2 d_{ij}^2(X)(Y, Y) = 2\text{tr } Y' A_{ij} Y$.

2.5.1.2 Rho and stress

At a regular configuration ρ is twice differentiable, with $\mathcal{D}\rho(X) = B(X)X$, and

$$\mathcal{D}^2 \rho(X)(Y, Z) = \sum_{1 \leq i < j \leq n} \sum_{1 \leq s < t \leq p} w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left\{ \text{tr } Y' A_{ij} Y - \frac{\text{tr } Y' A_{ij} X \text{tr } Y' A_{ij} X \text{tr } Z' A_{ij} X}{d_{ij}^2(X)} \right\} \quad (2.33)$$

By substituting $Y = e_i e_s'$ and $Z = e_j e_t'$ we get the order np Hessian of ρ at X .

$$\{\nabla^2 \rho(X)\}_{is,jt} = \sum_{1 \leq i < j \leq n} \sum_{1 \leq s < t \leq p} w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left\{ \delta^{st} A_{ij} - \frac{(x_{is} - x_{js})(x_{it} - x_{jt})}{d_{ij}^2(X)} \right\}.$$

We reserve the symbol \mathfrak{H} for this matrix valued function. Note that $0 \lesssim \mathfrak{H}(X) \lesssim I_p \otimes B(X)$ for all X .

Thus, again assuming regularity,

$$\mathcal{D}\sigma(X) = 2(V - B(X))X. \quad (2.34)$$

For the second derivatives of stress at a regular configuration we obtain

$$\mathcal{D}^2\sigma(X)(Y, Z) = 2\text{tr } Y'(V - B(X))Z + 2 \sum_{1 \leq i < j \leq n} \sum w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left\{ \frac{(\text{tr } Y'A_{ij}X)(\text{tr } Z'A_{ij}X)}{d_{ij}^2(X)} \right\}. \quad (2.35)$$

Formula (2.35) was first given by De Leeuw (1988). It was used, for various purposes, in De Leeuw (1993), De Leeuw and Groenen (1997), and De Leeuw (2017e). Most of these uses will also be detailed in this book.

2.5.2 Expansions

We now take a step back from regularity and expand stress at X in the direction Y , i.e. we look at $\sigma(X + \epsilon Y)$ for small positive ϵ .

If $d_{ij}(X) > 0$ then

$$\begin{aligned} d_{ij}(X + \epsilon Y) &= d_{ij}(X) + \epsilon \frac{\text{tr } X'A_{ij}Y}{d_{ij}(X)} + \\ &\quad + \frac{1}{2}\epsilon^2 \frac{1}{d_{ij}(X)} \left\{ d_{ij}^2(Y) - \frac{(\text{tr } X'A_{ij}Y)^2}{d_{ij}^2(X)} \right\} + o(\epsilon^2). \end{aligned} \quad (2.36)$$

Remember that $o(\epsilon^2)$ is any function f of ϵ with $\lim_{\epsilon \rightarrow 0} f(\epsilon)/\epsilon^2 = 0$. If $d_{ij}(X) = 0$ then of course $d_{ij}(X + \epsilon Y) = \epsilon d_{ij}(Y)$. Thus

$$\begin{aligned} \rho(X + \epsilon Y) &= \rho(X) + \\ &\quad + \epsilon \left(\text{tr } X'B(X)Y + \sum_{1 \leq i < j \leq n} \{w_{ij}\delta_{ij}d_{ij}(Y) \mid w_{ij}\delta_{ij} > 0 \text{ but } d_{ij}(X) = 0\} \right) + \\ &\quad + \frac{1}{2}\epsilon^2 \sum_{1 \leq i < j \leq n} \left\{ w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left(d_{ij}^2(Y) - \frac{(\text{tr } X'A_{ij}Y)^2}{d_{ij}^2(X)} \right) \mid w_{ij}\delta_{ij}d_{ij}(X) > 0 \right\} + o(\epsilon^2). \end{aligned} \quad (2.37)$$

2.5.3 Directional Derivatives

This section computes first and second directional derivatives of stress. For definitions of the directional derivatives we refer to chapter 25, section 25.4.3.

2.5.3.1 First Order

If $d_{ij}(X) > 0$ then d_{ij} is differentiable at X and thus, not surprisingly,

$$\mathcal{D}_+ d_{ij}(X; Y) = \lim_{\epsilon \downarrow 0} \frac{d_{ij}(X + \epsilon Y) - d_{ij}(X)}{\epsilon} = \frac{\operatorname{tr} X' A_{ij} Y}{d_{ij}(X)} = \operatorname{tr} Y' \mathcal{D} d_{ij}(X). \quad (2.38)$$

If $d_{ij}(X) = 0$ then

$$\mathcal{D}_+ d_{ij}(X; Y) = \lim_{\epsilon \downarrow 0} \frac{d_{ij}(X + \epsilon Y) - d_{ij}(X)}{\epsilon} = d_{ij}(Y). \quad (2.39)$$

Thus

$$\mathcal{D}_+ \sigma(X; Y) = 2 \operatorname{tr} Y' (V - B(X)) X - 2 \sum_{d_{ij}(X)=0} \sum w_{ij} \delta_{ij} d_{ij}(Y) \quad (2.40)$$

2.5.3.2 Second Order

$$\mathcal{D}_+^2 d_{ij}(X; Y) = \lim_{\epsilon \downarrow 0} \frac{d_{ij}(X + \epsilon Y) - d_{ij}(X) - \epsilon \mathcal{D}_+ d_{ij}(X; Y)}{\frac{1}{2} \epsilon^2} \quad (2.41)$$

Since if $d_{ij}(X) > 0$

$$d_{ij}(X + \epsilon Y) = d_{ij}(X) + \epsilon \frac{\operatorname{tr} X' A_{ij} Y}{d_{ij}(X)} + \frac{1}{2} \epsilon^2 \left\{ \frac{d_{ij}^2(Y)}{d_{ij}(X)} - \frac{(\operatorname{tr} X' A_{ij} Y)^2}{d_{ij}^3(X)} \right\} + o(\epsilon^2), \quad (2.42)$$

we see that

$$\mathcal{D}_+^2 d_{ij}(X; Y) = \frac{1}{d_{ij}(X)} \left\{ d_{ij}^2(Y) - \frac{(\text{tr } X' A_{ij} Y)^2}{d_{ij}^2(X)} \right\} = \mathcal{D}^2 d_{ij}(X)(Y, Y). \quad (2.43)$$

If $d_{ij}(X) = 0$ then $\mathcal{D}_+^2 d_{ij}(X; Y) = 0$. Thus

$$\mathcal{D}_+^2 \sigma(X; Y) = \sum_{\substack{1 \leq i < j \leq n \\ d_{ij}(X) > 0}} w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left\{ d_{ij}^2(Y) - \frac{(\text{tr } X' A_{ij} Y)^2}{d_{ij}^2(X)} \right\}$$

For the Ben-Tal and Zowe second directional derivative, assuming $d_{ij}(X) > 0$, we need the expansion

$$\begin{aligned} d_{ij}(X + \epsilon Y + \epsilon^2 Z) &= \\ &= d_{ij}(X) + \epsilon \frac{\text{tr } X' A_{ij} Y}{d_{ij}(X)} + \frac{1}{2} \epsilon^2 \left\{ \frac{d_{ij}^2(Y)}{d_{ij}(X)} + 2 \frac{\text{tr } X' A_{ij} Z}{d_{ij}(X)} - \frac{(\text{tr } X' A_{ij} Y)^2}{d_{ij}^3(X)} \right\} + o(\epsilon^2), \end{aligned} \quad (2.44)$$

and thus

$$\mathcal{D}_{BZ}^2 d_{ij}(X; Y, Z) = \frac{1}{d_{ij}(X)} \left\{ d_{ij}^2(Y) - \frac{(\text{tr } X' A_{ij} Y)^2}{d_{ij}^2(X)} \right\} + 2 \frac{\text{tr } X' A_{ij} Z}{d_{ij}(X)}. \quad (2.45)$$

If $d_{ij}(X) = 0$ and $d_{ij}(Y) > 0$ then

$$d_{ij}(X + \epsilon Y + \epsilon^2 Z) = \epsilon d_{ij}(Y) + \epsilon^2 \frac{\text{tr } Y' A_{ij} Z}{d_{ij}(Y)} + o(\epsilon^2)$$

and thus

$$\mathcal{D}_{BZ}^2 d_{ij}(X; Y, Z) = \frac{\text{tr } Y' A_{ij} Z}{d_{ij}(Y)} = \text{tr } Z' \mathcal{D} d_{ij}(Y).$$

If both $d_{ij}(X) = 0$ and $d_{ij}(Y) = 0$ then

$$d_{ij}(X + \epsilon Y + \epsilon^2 Z) = \epsilon^2 d_{ij}(Z)$$

and thus

$$\mathcal{D}_{BZ}^2 d_{ij}(X; Y, Z) = d_{ij}(Z).$$

2.5.4 Special Expansions

Example 2.1. Suppose $Y = XT$, with T antisymmetric, so that $X + \epsilon Y = X(I + \epsilon T)$. Then

$$\sigma(X + \epsilon Y) = \sigma(X) + \frac{1}{2}\epsilon^2 \operatorname{tr} Y'(V - B(X))Y + o(\epsilon^2) \quad (2.46)$$

Example 2.2. Suppose $\underline{X} = [X \mid 0]$ and $\underline{Y} = [0 \mid Y]$ so that $\underline{X} + \epsilon \underline{Y} = [X \mid \epsilon Y]$. Here \underline{X} and \underline{Y} are $n \times p$, X is $n \times r$, with $r < p$, and Y is $n \times (p - r)$. Then

$$\sigma(\underline{X} + \epsilon \underline{Y}) = \sigma(X) - \epsilon \sum_{d_{ij}(X)=0} w_{ij} \delta_{ij} d_{ij}(Y) + \frac{1}{2}\epsilon^2 \operatorname{tr} Y'(V - B(X))Y + o(\epsilon^2) \quad (2.47)$$

Example 2.3. Suppose $\underline{X} = [X \mid 0]$ and $\underline{Y} = [Z \mid Y]$ so that $\underline{X} + \epsilon \underline{Y} = [X + \epsilon Z \mid \epsilon Y]$. Here \underline{X} and \underline{Y} are $n \times p$, X is $n \times r$, with $r < p$, and Y is $n \times (p - r)$. Then

$$\sigma(\underline{X} + \epsilon \underline{Y}) = \sigma(X) - \epsilon \sum_{d_{ij}(X)=0} w_{ij} \delta_{ij} d_{ij}(Y) + \frac{1}{2}\epsilon^2 \operatorname{tr} Y'(V - B(X))Y + o(\epsilon^2) \quad (2.48)$$

2.5.5 Zero Distance Subspaces

A *zero-distance subspace* is a subspace of configuration space in which one or more distances are zero. That a zero distance indeed defines a subspace follows from the fact that $d_{ij}(X) = 0$ is equivalent to $x_i = x_j$.

How many zero-distance subspaces are there? The same as the number of set partitions of n objects, which is the Bell number B_n . Bell numbers are defined by the recursion

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k, (\$eq : bellnums) \quad (2.49)$$

with $B_0 = 1$. The next ten Bell numbers B_1, \dots, B_{10} are 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 11597. So there are lots of zero-distance subspaces.

As we know in basic MDS zero distances cause problems with differentiability. But if we know ahead of time where we want the zero distances (i.e. which points we want to coincide) then we can work around this.

Suppose the n objects are partitioned into r subsets. Define the $n \times r$ indicator matrix G , which codes which subset each object belongs to. Then we minimize stress over X with the constraint $X = GZ$, where Z is $r \times p$, i.e. we really minimize σ over Z . $B(Z) := G'B(X)G$ and $V = G'VG$

$$G'B(X)GZ = G'VGZ$$

$$G'(B(X)X - VX)$$

Stability

2.5.6 Distance Smoothing

In sections 2.3 and 2.6 we show the lack of differentiability in basic MDS is not a serious problem in the actual computation of local minima. There is another rather straightforward way to circumvent the differentiability issue, which actually may have additional benefits. The idea is to use an approximation of the Euclidean distance that is as close as possible on the positive real axis but smooth at zero. This was first applied in unidimensional MDS by Pliner (Pliner (1986), Pliner (1996)) and later taken up and generalized to pMDS for arbitrary p , and even for arbitrary Minkovski metrics, by Groenen, Heiser, and Meulman (1998) and Groenen, Heiser, and Meulman (1999). They coined the term *distance smoothing* for this variation of the smacof framework for MDS.

Pliner uses a smooth approximation of the sign function, while Groenen et al. use a smooth Huber approximation of the absolute value function. We use a classical and efficient approximation $|x| \approx \sqrt{x^2 + \epsilon^2}$ to the absolute value function, used in image analysis, location analysis, and computational geometry (De Leeuw (2018b), Ramirez et al. (2014)). In our context that

becomes $d_{ij}(X) \approx d_{ij}(X, \epsilon) := \sqrt{d_{ij}^2(X) + \epsilon^2}$. Note that on the non-negative reals

$$\max(\epsilon, d_{ij}(X)) \leq d_{ij}(X, \epsilon) \leq d_{ij}(X) + \epsilon. \quad (2.50)$$

Figures 2.1 and 2.2 show the absolute value function and its derivative are approximated for ϵ equal to 0, 0.01, 0.05, 0.1, 0.5.

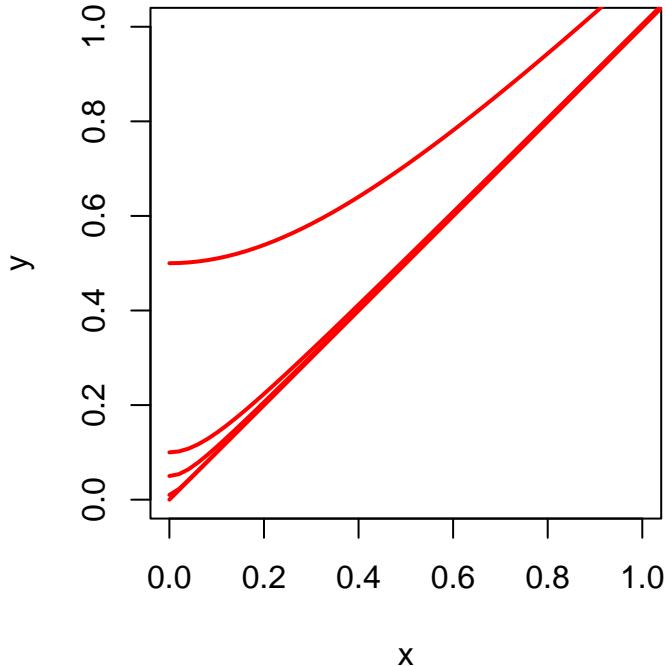


Figure 2.1: Function for Various Epsilon

The distance smoother we use fits nicely into smacof. Define $X_\epsilon := [X \mid \epsilon I]$. Then $d_{ij}(X_\epsilon) = \sqrt{d_{ij}^2(X) + \epsilon^2}$. Thus we can define

$$\sigma_\epsilon(X) := \sigma(X_\epsilon) = \sum_{1 \leq i < j \leq n} \sum w_{ij} (\delta_{ij} - d_{ij}(X_\epsilon))^2, \quad (2.51)$$

with ρ_ϵ and η_ϵ^2 defined in the same way.

For a fixed $\epsilon > 0$ now $d_{ij}(X_\epsilon)$, and thus stress, is (infinitely many times) differentiable n all of $\mathbb{R}^{n \times p}$. Moreover $d_{ij}(X, \epsilon)$ is convex in X for fixed ϵ and jointly convex in X and ϵ , and as a consequence so are ρ_ϵ and η_ϵ^2 .

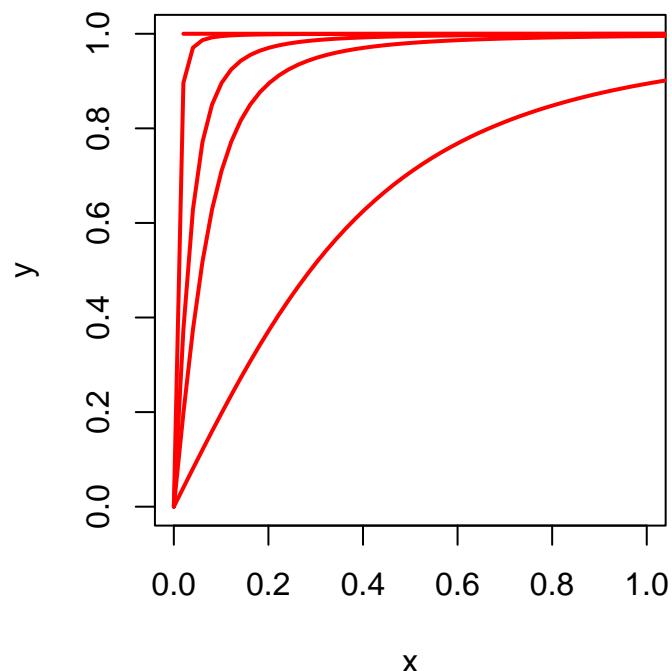


Figure 2.2: Derivative for Various Epsilon

2.6 Stationary Points

Theorem 2.2. *At a stationary point of stress we have $\eta(X) \leq 1$.*

If X is stationary we have $VX = B(X)X$ and thus $\rho(X) = \eta^2(X)$. Consequently $\sigma(X) = 1 - 2\rho(X) + \eta^2(X) = 1 - \eta^2(X)$ and because $\sigma(X) \geq 0$ we see that X must be in the ellipse $\{Z \in \mathbb{R}^{n \times p} \mid \eta^2(Z) \leq 1\}$.

Theorem 2.2 is important, because it means that we can require without loss of generality that X is in the ellipsoidal disk $\eta(X) \leq 1$, which is a compact convex set.

2.6.1 Local Maxima

Theorem 2.3. *stress has a single local maximum at $X = 0$ with value 1.*

Proof. At $X = 0$ we have for the one-sided directional derivative

$$\mathbb{D}_+ \sigma(0, Y) = \lim_{\alpha \downarrow 0} \frac{\sigma(0 + \alpha Y) - \sigma(0)}{\alpha} = -2\rho(Y) \leq 0, \quad (2.52)$$

which implies that stress has a local maximum at zero.

By contradiction: suppose that there is a local maximum at $X \neq 0$. Then on the line through zero and X there should be a local maximum at X as well. But

$$\sigma(\alpha X) = 1 - 2\alpha\rho(X) + \alpha^2\eta^2(X), \quad (2.53)$$

is a convex quadratic, which consequently cannot have a local maximum at X . \square

2.6.2 Local Minima

The main result on local minima of stress is due to De Leeuw (1984c). We give a slight strengthening of the result, along the lines of De Leeuw (2018a), and a slightly simplified proof.

Theorem 2.4. *stress is differentiable at local minima.*

Proof. It suffices to prove that if stress has a local minimum at X we have $d_{ij}(X) > 0$ for all $w_{ij}\delta_{ij} > 0$, in other words that ρ in equation (2.4) is differentiable at X . Since η^2 is differentiable everywhere this will show stress is differentiable at X .

From equation (2.40) we see

Because □

Note: suppose $\delta_{12} = 0$ and $\delta_{13} = \delta_{23} = \frac{1}{2}\sqrt{2}$ and all weights are one. Thus

$$\sigma(X) = 1 + d_{12}^2(X) + \left(\frac{1}{2}\sqrt{2} - d_{13}(X)\right)^2 + \left(\frac{1}{2}\sqrt{2} - d_{23}(X)\right)^2$$

If in addition $w_{12} = 0$ then

$$\sigma(X) = 1 + \left(\frac{1}{2}\sqrt{2} - d_{13}(X)\right)^2 + \left(\frac{1}{2}\sqrt{2} - d_{23}(X)\right)^2$$

Saddle Points {#propsaddle}

At a saddle point X stress is differentiable and $\mathcal{D}\sigma(X) = 0$. But there are directions of decrease and increase from X , and thus stress does not have a local minimum there.

If $VX = B(X)X$ and $d_{ij} = 0$ then saddle point ? No.

If $VX = B(X)X$ then $\mathbb{D}_+\sigma(X, Y) \leq 0$

Theo Suppose $VX = B(X)X$ then $V(X | 0) = B(X|0)(X|0)$

Corr Suppose $VX = B(X)X$ and X is of rank $r < p$. Then $XL = (Z|0)$ and thus $VZ = B(Z)Z$.

If $VX = B(X)X$ and X is singular then X is a saddle point.

2.7 Stress Envelopes

intro

2.7.1 CS Majorization

Theorem 2.5. σ is the lower envelop of an infinite number of convex quadratics.

Proof. By the CS inequality

$$d_{ij}(X) = \max_Y \frac{\text{tr } X^T A_{ij} Y}{d_{ij}(Y)}, \quad (2.54)$$

which implies

$$\sigma(X) = \min_Y \left(1 - \text{tr } X^T B(Y) Y + \frac{1}{2} \text{tr } X^T V X \right), \quad (2.55)$$

which is what we set out to prove. \square

We can use the lower envelop of a finite number of the quadratics from theorem 2.5 to approximate stress. This is illustrated graphically, using a small example in which the configuration is a convex combination of two fixed configurations. Thus in the example stress is a function of the single parameter $0 \leq \lambda \leq 1$ defining the convex combination. In figure 2.3 stress is in red, and we have used the three quadratics corresponding with λ equal to 0.25, 0.5, 0.75. The maximum of the three quadratics is in blue, and the approximation is really good, in fact almost perfect in the areas where the blue is not even visible. As an aside, we also see three points in the figure where stress is not differentiable. The minimum of the three quadratics is also not differentiable at a point, but that point is different from the points where stress is non-smooth.

Note that by definition stress and the lower envelop of the quadratics are equal at the three points where λ is 0.25, 0.5, 0.75, i.e at the three vertical lines in the plot.

2.7.2 AM/GM Minorization

Instead of approximating stress from above, we can also approximate it from below.

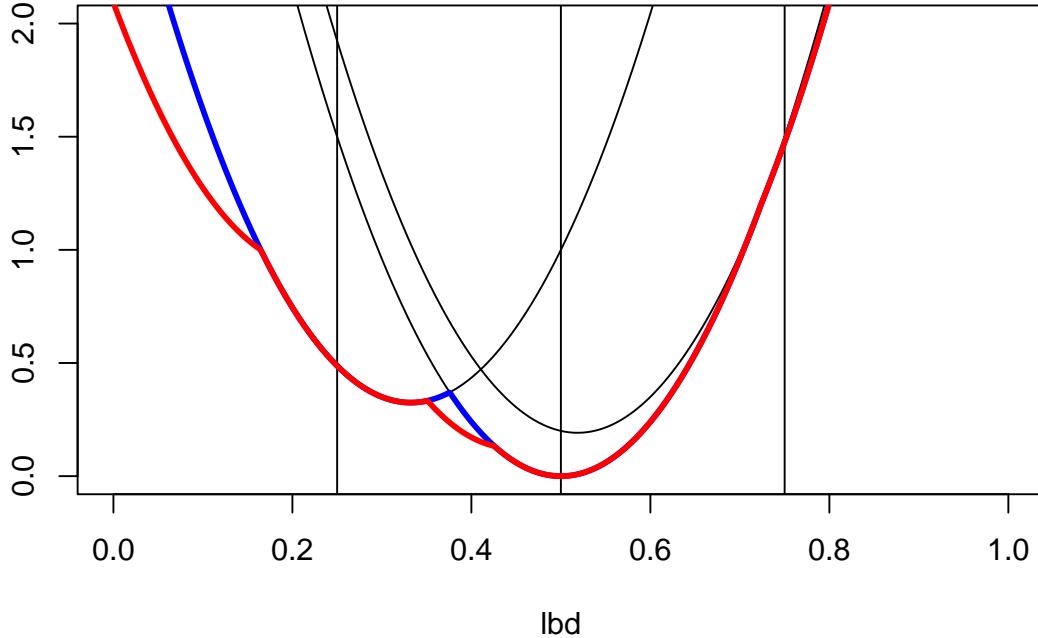


Figure 2.3: Piecewise Quadratic Upper Approximation

Theorem 2.6. σ is the upper envelop of an infinite number of quadratics.

Proof. By AM/GM

$$d_{ij}(X) \leq \min \frac{1}{2} \frac{1}{d_{ij}(Y)} \{ d_{ij}^2(X) + d_{ij}^2(Y) \} \quad (2.56)$$

Thus

$$\sigma(X) = \max_Y \left(1 - \frac{1}{2} \rho(Y) + \frac{1}{2} \text{tr } X'(V - B(Y))X \right) \quad (2.57)$$

□

Again we illustrate this result using a finite number of quadratics. In figure 2.4 we choose λ equal to 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. Although we now use 11 quadratics, and thus force the envelop to be equal to the function at the 11 points on the vertical lines in the plot, the approximation is poor. This seems to be mainly because the convex-like function stress must be approximated from below by quadratics which are often concave.

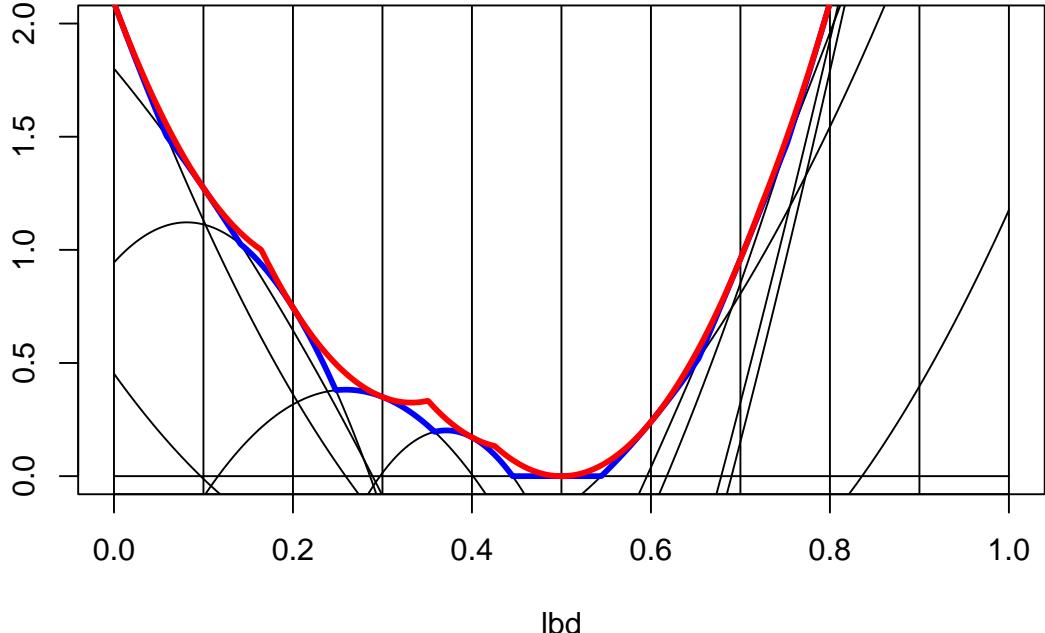


Figure 2.4: Piecewise Quadratic Lower Approximation

2.7.3 Dualities

$$\begin{aligned} \min_X \sigma(X) &= \min_Y \left(1 - \frac{1}{2} \text{tr } Y' B(Y) V^+ B(Y) Y \right) = \\ &= 1 - \frac{1}{2} \max_Y \text{tr } Y' B(Y) V^+ B(Y) Y. \end{aligned} \quad (2.58)$$

Thus minimizing stress is equivalent to maximizing $\eta^2(V^+ B(X) X)$.

$$\min_X \sigma(X) \geq \max_{B(Y) \lesssim V} (1 - \rho(Y))$$

By the minimax inequality $\min_X \sigma(X) = \min_X \max_Y \theta(X, Y) \geq \max_Y \min_X \theta(X, Y)$. Now $\min_X \theta(X, Y)$ is $-\infty$, unless $B(Y) \lesssim V$, in which case $\min_X \theta(X, Y) = 0$. Thus

$$\max_Y \min_X \theta(X, Y) = \max_{B(Y) \lesssim V} (1 - \rho(Y))$$

2.8 How Large is My Stress ?

Kruskal

Thurstonian

Ramsay

Monte Carlo

$$\mathbb{E}(\chi_p) = \sqrt{2} \frac{\Gamma(\frac{p+1}{2})}{\Gamma(\frac{p}{2})}$$

Chapter 3

Stress Spaces

intro

Much of this chapter is a modified, and in some places expanded, version of De Leeuw (2016e).

3.1 Configuration Space

So far we have defined stress on $\mathbb{R}^{n \times p}$, the space of all matrices with n rows and p columns. We call this *configuration space*.

Even for n as small as four and p as small as two the dimension of the space of centered configurations is six, and there is no natural way to visualize a function of six variables.

coordinates

3.2 Coefficient Space

3.2.1 On the Planes

What we can do is plot stress on two-dimensional subspaces, either as a contour plot or as a perspective plot. Our two-dimensional subspaces are of the form $\alpha X + \beta Y$, where X and Y are fixed configurations.

3.2.2 General

Now let Y_1, Y_2, \dots, Y_r be linearly independent configurations in $\mathbb{R}^{n \times p}$, and consider minimizing stress over all linear combinations X of the form $X = \sum_{s=1}^r \theta_s Y_s$.

Each linear combination can be identified with a unique vector $\theta \in \mathbb{R}^r$, the coefficients of the linear combination. Thus we can also formulate our problem as minimizing stress over *coefficient space*, which is simply \mathbb{R}^r . We write $d_{ij}(\theta)$ for $d_{ij}(X)$ and $\sigma(\theta)$ for $\sigma(X)$. Note that $d_{ij}(\theta) = \sqrt{\theta' C_{ij} \theta}$, where C_{ij} has elements $\{C_{ij}\}_{st} := \text{tr } Y_s' A_{ij} Y_t$.

If the Y_t are actually a basis for configuration space (i.e. if $r = np$) then minimizing over configuration space and coordinate space is the same thing. For the Y_t we could choose all rank one matrices, for example, of the form $a_i b_s'$ where the a_i are a basis for \mathbb{R}^n and the b_s are a basis for \mathbb{R}^p . And, in particular, the a_i and b_s can be chosen as unit vectors of length n and p , respectively. That case we have $C_{ij} = I_p \otimes A_{ij}$, i.e. the direct sum of p copies of A_{ij} . Also if $\theta = \text{vec}(X)$ then $d_{ij}(X) = \sqrt{\theta' (I_p \otimes A_{ij}) \theta}$

If $r < np$ then coefficient space defines a proper subspace of configuration space. If it happens to be the $(n - 1)p$ dimensional subspace of all column-centered matrices, then the two approaches still define the same minimization problem. But in general $r < (n - 1)p$ with the Y_s column-centered defines a *constrained MDS problem*, which we analyze in more detail in chapter 15.

Coefficient space is also a convenient place to deal with rotational indeterminacy in basic MDS. It follows from QR decomposition that any configuration matrix can be rotated in such a way that its upper diagonal elements (the x_{ij} with $i < j$) are zero (define X_p to be the first p rows of X , compute $X_p' = QR$ with Q square orthonormal and R upper triangular, thus $X_p = R' Q'$ and $X_p Q = R'$, which is lower triangular). The column-centered upper triangular configurations are a subspace of dimension $p(n - 1) - p(p - 1)/2$, and we can choose the Y_s as a basis for this subspace. In this way we eliminate rotational indeterminacy in a relatively inexpensive way.

If $X = \sum_{s=1}^r \theta_s Y_s$ then we define the symmetric positive definite matrix $B(\theta)$ of order r with elements

$$b_{st}(\theta) := \text{tr } Y_s' B(X) Y_t, \quad (3.1)$$

where $B(X)$ is the usual B-matrix of order n in configuration space, defined in equation (2.14). Also define V of order r by

$$v_{st} := \text{tr } Y_s' V Y_t, \quad (3.2)$$

where the second V , of order n , is given by equation (2.10). Then

$$\sigma(\theta) = 1 - 2 \theta' B(\theta) \theta + \theta' V \theta. \quad (3.3)$$

The relationship between the stationary points in configuration space and coefficient space is fairly straightforward.

Theorem 3.1. *Suppose θ is in coefficient space and $X = \sum_{s=1}^r \theta_s Y_s$ is the corresponding point in configuration space.*

1. *If X is a stationary point in configuration space then θ is a stationary point in coefficient space.*
2. *If θ is a stationary point in coefficient space then X is a stationary point in configuration space if and only if $\text{rank}(Y_1 \mid \dots \mid Y_r) \geq n - 1$. (THIS IS WRONG)*

Proof. We have $B(X)X = VX$, i.e.

$$\sum_{s=1}^r \theta_s B(X) Y_s = \sum_{s=1}^r \theta_s V Y_s. \quad (3.4)$$

Premultiplying both sides by Y_t^T and taking the trace gives $B(\theta)\theta = V\theta$. This proves the first part.

For the second part, suppose $B(\theta)\theta = V\theta$ and define $X = \sum_{s=1}^r \theta_s Y_s$. Then

$$\sum_{t=1}^r \text{tr } Y_t^T (B(X) - V) X = 0. \quad (3.5)$$

Thus $B(X)X = VX$ if and only if $Y_s^T (B(X) - V) X = 0$ for all s , which translates to the rank condition in the theorem (this is WRONG, correct).

□

The advantage of working in coefficient space is that formulas tend to become more simple. Functions are defined on \mathbb{R}^r , and not a space of matrices, in which some coordinates belong to the same point (row) and others to other points (rows), and some are on the same dimension (column), while others are on different dimensions (columns).

Note that expressions such as (3.4) and (3.4) simplify if the Y_s are V -orthonormal, i.e. if $\text{tr } Y_s' V Y_t = \delta^{st}$ and thus $V = I$. It is easy to generate such an orthonormal set from the original Y_s by using the Gram-Schmidt process. The R function `gramy()` in `utilities.R` does exactly that. Coefficient space, which is the span of the Y_s , is not changed by the orthogonalization process.

For a V -orthonormal set Y we have the stationary equations $B(\theta)\theta = \theta$, which says that θ is an eigenvector of $B(\theta)$ with eigenvalue 1.

The Hessian is

$$\mathcal{D}^2\sigma(\theta) = I - H(\theta), \quad (3.6)$$

with

$$H(\theta) := \mathcal{D}^2\rho(\theta) = \sum_{1 \leq i < j \leq n} w_{ij} \frac{\delta_{ij}}{d_{ij}(\theta)} \left\{ C_{ij} - \frac{C_{ij}\theta\theta' C_{ij}}{\theta' C_{ij}\theta} \right\}. \quad (3.7)$$

We have $0 \lesssim H(\theta) \lesssim B(\theta)$ and thus $I - B(\theta) \lesssim \mathcal{D}^2\sigma(\theta) \lesssim I$.

Hessian in coef and conf space

3.3 Sphere Space

$$\min_X \sigma(X) = \min_{\lambda \geq 0} \min_{\eta^2(X)=1} \sigma(\lambda X) = \min_{\eta^2(X)=1} \min_{\lambda \geq 0} \sigma(\lambda X) = \min_{\eta^2(X)=1} 1 - \rho^2(X). \quad (3.8)$$

We see that basic MDS can also be formulated as maximization of ρ over the ellipsoid $\{X \mid \eta^2(X) = 1\}$ or, equivalently, over the convex ellipsoidal disk $\{X \mid \eta^2(X) \leq 1\}$. A similar formulation is available in coefficient space.

This shows that the MDS problem can be seen as a rather special nonlinear eigenvalue problem. Guttman (1968) also discussed the similarities of MDS and eigenvalue problems, in particular as they related to the power method. In linear eigenvalue problems we maximize a convex quadratic form, in the MDS problem we maximize the homogeneous convex function ρ , in both cases over an ellipsoidal disk. The sublevel sets of ρ defined as $\mathcal{L}_r := \{X \mid \rho(X) \leq r\}$ are nested convex sets containing the origin. The largest of these sublevel sets that still intersects the ellipsoid $\eta^2(X) = 1$ corresponds to the global minimum of stress (see section 4.5).

$\alpha X + \beta Y$ in sphere space one-dimensional 4.6

3.4 Cross-Product Space

We can write

$$d_{ij}^2(X) = \text{tr } X' A_{ij} X = \text{tr } A_{ij} C,$$

with $C = XX'$. This shows minimizing stress can also be formulated as minimizing

$$\sigma(C) = 1 + \text{tr } VC - 2 \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} \sqrt{\text{tr } A_{ij} C} \quad (3.9)$$

over all positive semi-definite C of rank $r \leq p$. A fundamental result, which forms the basis of chapter 13 in this book, is that σ is convex on *Cross-Product Space*, i.e. the closed convex cone of all $C \gtrsim 0$.

Chapter 4

Pictures of Stress

Even for n as small as four and p as small as two the dimension of the space of centered configurations is six, and there is no natural way to visualize a function of six variables. What we can do is plot stress on two-dimensional subspaces, either as a contour plot or as a perspective plot. Our two-dimensional subspaces are of the form $\alpha X + \beta Y$, where X and Y are fixed configurations. Much of this chapter is a modified, and in some places expanded, version of De Leeuw (2016e).

Throughout we use a small example of order $n = 4$ which all dissimilarities equal. The same example has been analyzed by De Leeuw (1988), De Leeuw (1993), Trosset and Mathar (1997), and Zilinskas and Poslipskyte (2003). For this example a global minimum in two dimensions has its four points in the corners of a square. That is our X , which has stress 0.0285955. Our Y is another stationary point, which has three points in the corners of an equilateral triangle and the fourth point in the center of the triangle. Its stress is 0.0669873. We column-center the configurations and scale them so that they are actually stationary points, i.e. so that $\eta^2(X) = \rho(X)$ and $\eta^2(Y) = \rho(Y)$. The example is chosen in such a way that there are non-zero α and β such that $d_{12}(\alpha X + \beta Y) = 0$. In fact d_{12} is the only distance that can be made zero by a non-trivial linear combination.

Another way of looking at the two configurations is that X are four points equally spaced on a circle, and Y are three points equally spaced on a circle with the fourth point in the center of the circle. De Leeuw (1988) erroneously claims that Y is a non-isolated local minimum of stress, but Trosset and

Mathar (1997) have shown there exists a descent direction at Y , and thus Y is actually a saddle point. Of course the stationary points defined by X and Y are far from unique, because we can permute the four points over the corners of the square and the triangle in many ways.

4.1 Coefficient Space

Configurations as a linear combination of a number of given configurations have already been discussed in general in chapter 2, section 3 as the transformation from configuration space to coefficient space. Since we are dealing here with the special case of linear combinations of only two configurations we specialize some of these general results.

We start with $d_{ij}^(\theta) = \theta' T_{ij} \theta$, where θ has elements α and β , and where T is the 2×2 matrix with elements

$$t_{ij} := \begin{bmatrix} \text{tr } X' A_{ij} X & \text{tr } X' A_{ij} Y \\ \text{tr } Y' A_{ij} X & \text{tr } Y' A_{ij} Y \end{bmatrix} \quad (4.1)$$

Then

$$\tilde{\sigma}(\theta) := 1 - 2 \theta' C(\theta) \theta + \theta' U \theta, \quad (4.2)$$

where, using $Z(\theta) = \alpha X + \beta Y$,

$$C(\theta) := \begin{bmatrix} \text{tr } X' B(Z(\theta)) X & \text{tr } X' B(Z(\theta)) Y \\ \text{tr } Y' B(Z(\theta)) X & \text{tr } Y' B(Z(\theta)) Y \end{bmatrix}, \quad (4.3)$$

and

$$U := \begin{bmatrix} \text{tr } X' V X & \text{tr } X' V Y \\ \text{tr } Y' V X & \text{tr } Y' V Y \end{bmatrix}. \quad (4.4)$$

We have used $\tilde{\sigma}$ in equation (4.2) to distinguish stress on the two-dimensional space of coefficients from stress on the eight-dimensional space of 4×2 configurations. Thus $\tilde{\sigma}(\alpha, \beta) = \sigma(\alpha X + \beta Z)$.

The gradient at θ is

$$\nabla \tilde{\sigma}(\theta) = U\theta - C(\theta)\theta, \quad (4.5)$$

and the Hessian is

$$\nabla^2 \tilde{\sigma}(\theta) = U - \sum_{1 \leq i < j \leq n} w_{ij} \frac{\delta_{ij}}{d_{ij}(\theta)} \left\{ T_{ij} - \frac{T_{ij}\theta\theta' T_{ij}}{\theta' T_{ij}\theta} \right\}. \quad (4.6)$$

Theorem 4.1. *If $B(X)X = VX$ and $\theta = [1 \ 0]$ then $C(\theta)\theta = U\theta$.*

Proof. If $B(X)X = VX$ and $\theta = [1 \ 0]$ then, by equations (4.3) and (4.4),

$$U - C(\theta) = \begin{bmatrix} 0 & 0 \\ 0 & \text{tr } Y'(V - B(X))Y \end{bmatrix}. \quad (4.7)$$

Thus $(U - C(\theta))\theta = 0$. □

Thus each stationary point of σ gives a stationary point of $\tilde{\sigma}$. The other way around, however, we are not so lucky.

Theorem 4.2. *If $C(\theta)\theta = U\theta$ and if the $n \times 2p$ matrix $[X \ Y]$ has rank $n - 1$ then $B(Z)Z = VZ$.*

Proof. If $C(\theta)\theta = U\theta$ then both $\text{tr } X'(V - B(Z)Z) = 0$ and $\text{tr } Y'(V - B(Z)Z) = 0$. If the $n \times 2p$ matrix $[X \ Y]$ has rank $n - 1$ then this implies $(V - B(Z))Z = 0$. □

In our example the singular values of $\begin{bmatrix} X & Y \end{bmatrix}$ are 0.4879059, 0.4333287, 0.2242285, $1.4867245 \times 10^{-17}$ and thus there is a one-one correspondence between stationary points of σ and $\tilde{\sigma}$.

Theorem 4.3. *If $B(X)X = VX$ and $\theta = \begin{bmatrix} 1 & 0 \end{bmatrix}$ then*

1. *If $\text{tr } Y'(V - B(X))Y > 0$ then σ has a local minimum at theta.*
 2. *If σ has a saddle point at θ then $\text{tr } Y'(V - B(X))Y < 0$.*
-

Proof. Suppose $B(X)X = VX$ and $\theta = \begin{bmatrix} 1 & 0 \end{bmatrix}$. Then, from (4.6) and (4.7),

$$\nabla^2\sigma(\theta) = \begin{bmatrix} 0 & 0 \\ 0 & \text{tr } Y'(V - B(X))Y \end{bmatrix} + \sum_{1 \leq i < j \leq n} \sum w_{ij} \frac{\delta_{ij}}{d_{ij}(\theta)} \frac{T_{ij}\theta\theta' T_{ij}}{\theta' T_{ij}\theta}.$$

□

In our example $\text{tr } X'(V - B(Y))X$ is -0.1502768 and $\text{tr } Y'(V - B(X))Y$ is -0.0533599.

4.2 Global Perspective

We first make a global perspective plot, over the range $(-2.5, +2.5)$.

We see the symmetry, following from the fact that stress is even. We also see the local maximum at the origin, where stress is not differentiable. Also note the ridge, where $d_{12}(\theta) = 0$ and where stress is not differentiable either. The ridge shows nicely that on rays emanating from the origin stress is a convex quadratic. Also, far away from the origin, stress globally behaves very much like a convex quadratic (except for the ridge). Clearly local minima must be found in the valleys surrounding the small mountain at the origin, all within the sphere with radius $\sqrt{2}$.

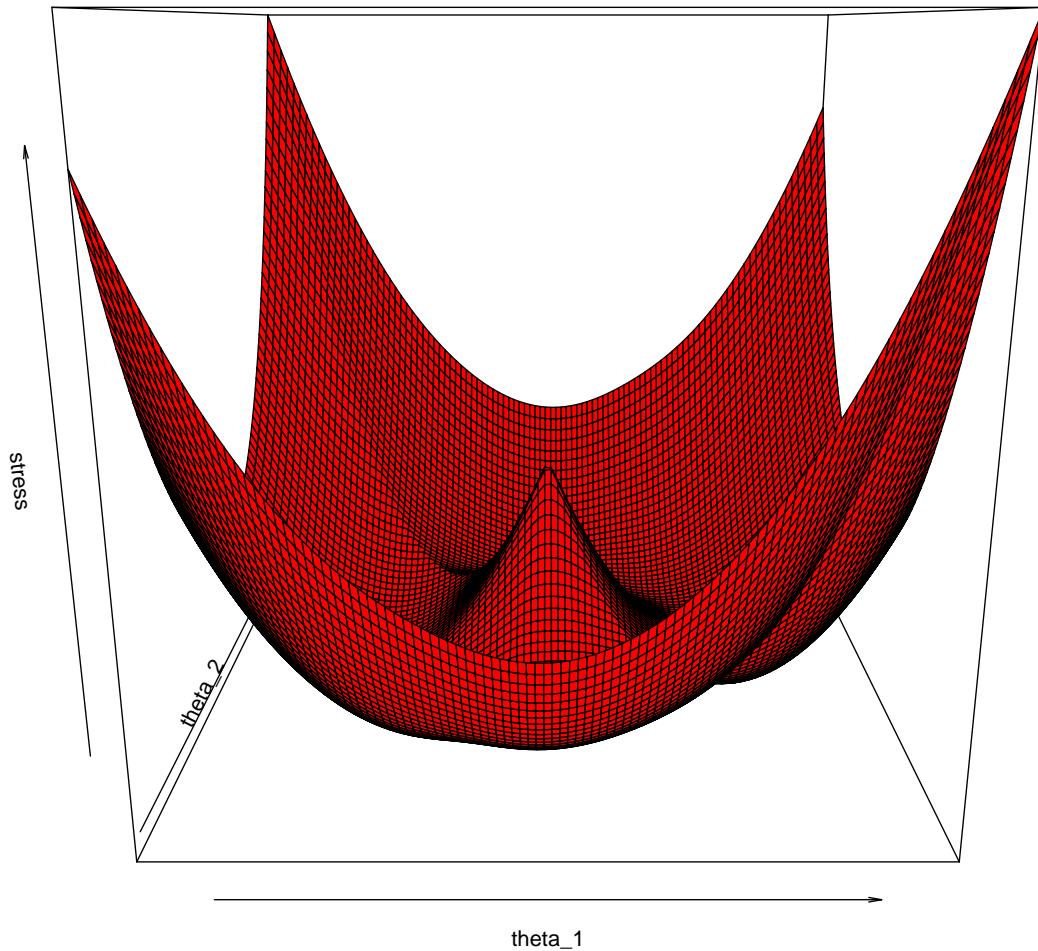


Figure 4.1: Global Perspective

4.3 Global Contour

Figure 4.1 is a contour plot of stress over $(-2, +2) \otimes (-2, +2)$. The red line is $\{\theta \mid d_{12}(\theta) = 0\}$. The blue line has the minimum of the convex quadratic on each of the rays through the origin. Thus all local minima, and in fact all stationary points, are on the blue line. Note that the plot uses θ to define the coordinate axes, not $\gamma = (\alpha, \beta)$. Thus there are no stationary points at $(0, 1)$ and $(1, 0)$, but at the corresponding points $(1.3938469, 0)$ and $(1.0406404, 0.8849253)$ in the θ coordinates (and, of course, at their mirror images).

Besides the single local maximum at the origin, it turns out that in this example there are five pairs of stationary points. Or, more precisely, I have not been able to find more than five. Each stationary point θ has a mirror image $-\theta$. Three of the five are local minima, two are saddle points. Local minima are plotted as blue points, saddle points as red points.

4.4 Stationary Points

4.4.1 First Minimum

We zoom in on the first local minimum at $(1.0406404, 0.8849253)$. Its stress is 0.0669873, and the corresponding configuration has three points in the corners of an equilateral triangle and the fourth point in its centroid. Note that this local minimum is a saddle point in configuration space $\mathbb{R}^{4 \times 2}$ (Trossset and Mathar (1997)). The eigenvalues of $B(\theta)$ are $(1.3686346, 1)$ and those of the Hessian $I - H(\theta)$ are $(1, 0.0817218)$. The area of the contour plot around the stationary value is in figure 4.3.

4.4.2 Second Minimum

The second local minimum (which is the global minimum) at $(1.3938469, 0)$ has stress 0.0285955. The configuration are the four points at the corners of a square. The eigenvalues of $B(\theta)$ are $(1.1362799, 1)$ and those of the Hessian $I - H(\theta)$ are $(1, 0.3743105)$. The area of the contour plot around the stationary value is in figure 4.4.

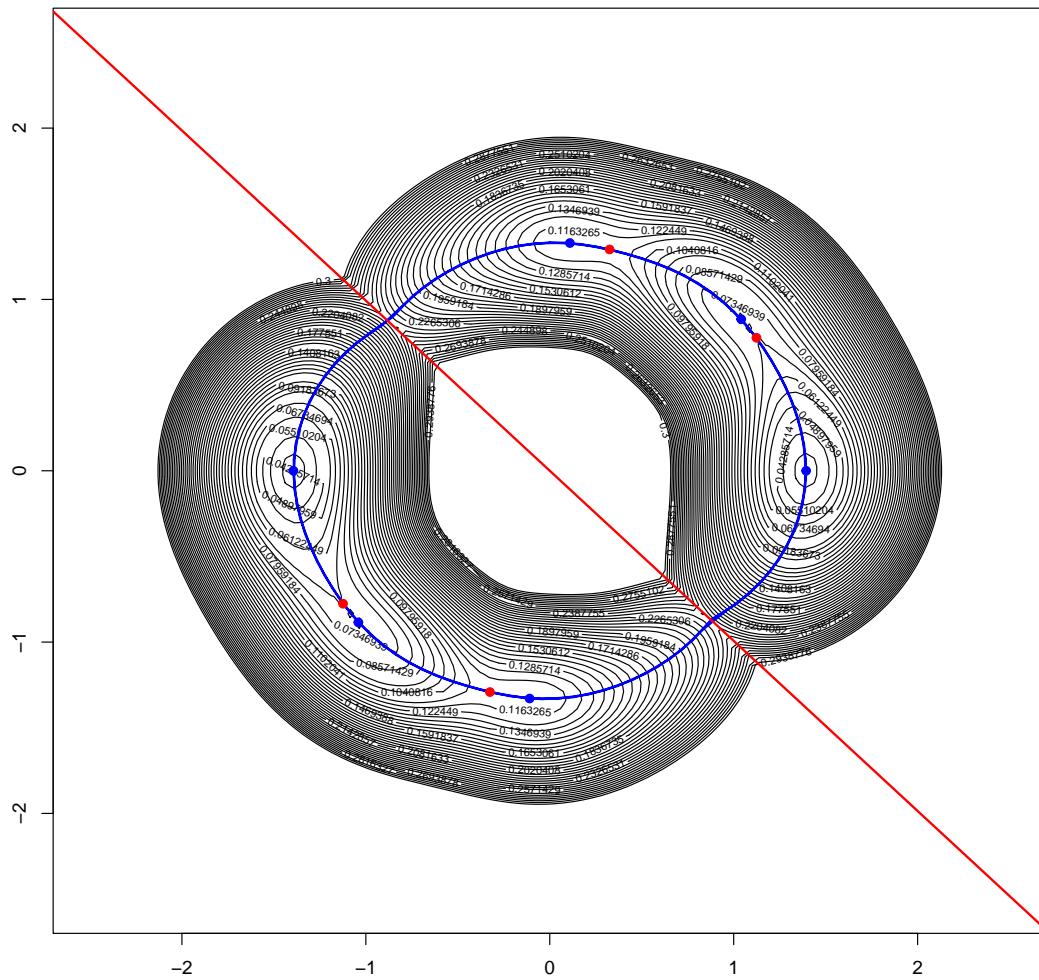


Figure 4.2: Global Contour

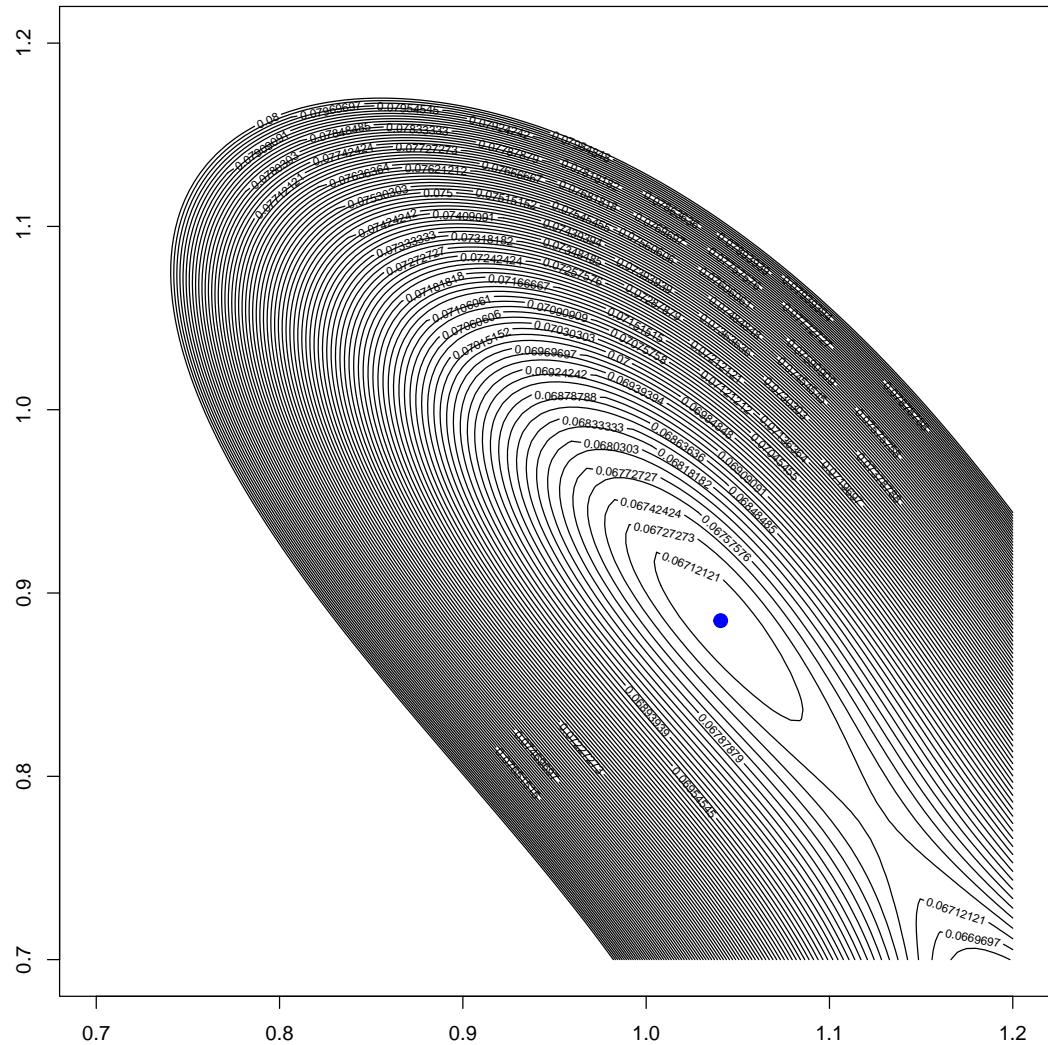


Figure 4.3: Contour Plot First Minimum

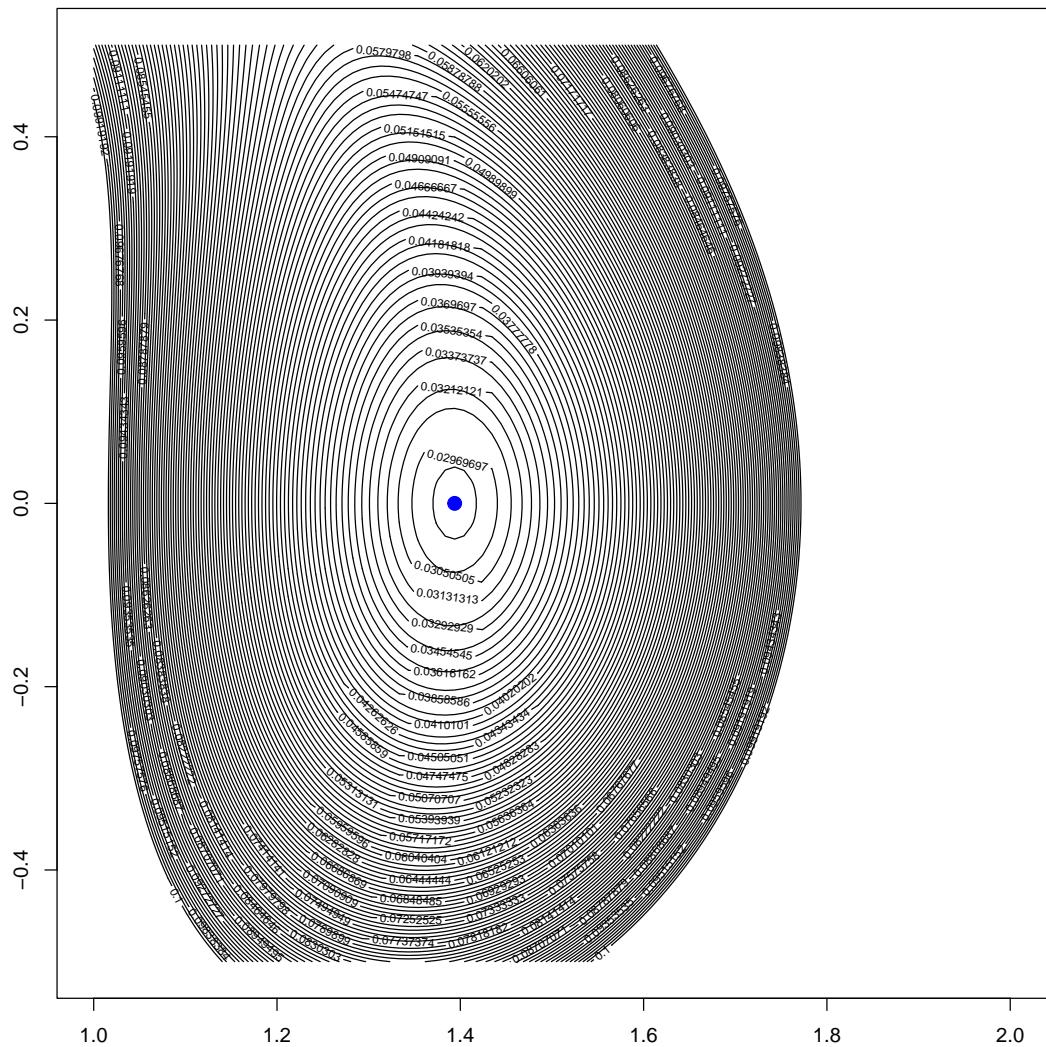


Figure 4.4: Contour Plot Second Minimum

4.4.3 Third Minimum

The third local minimum at $(0.1096253, 1.3291942)$ has stress 0.1106125, and the corresponding configuration is in figure 4.5.

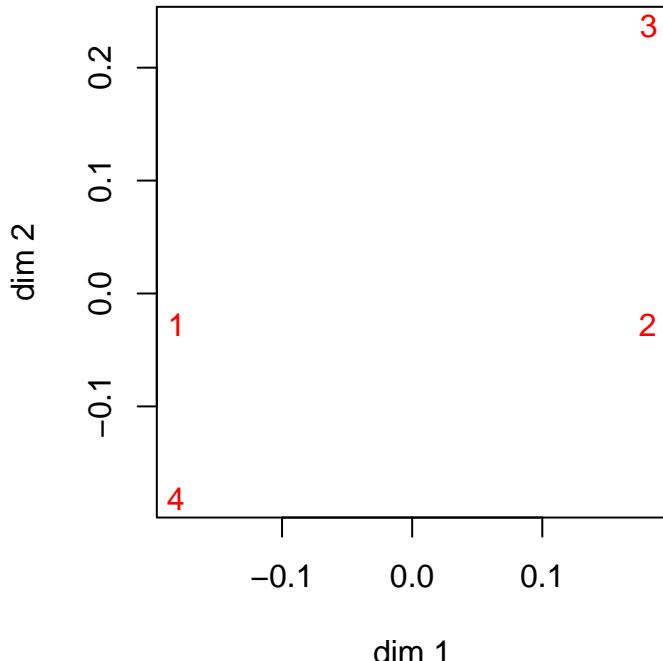


Figure 4.5: Configuration Third Minimum

The eigenvalues of $B(\theta)$ are $(1.5279386, 1)$ and those of the Hessian $I - H(\theta)$ are $(1, 0.2362079)$. The area of the contour plot around the stationary value is in figure 4.6

4.4.4 First Saddle Point

The saddle point at $(0.3253284, 1.2916758)$ has stress 0.1128675, and the corresponding configuration is in figure 4.7.

The eigenvalues of $B(\theta)$ are $(1.7778549, 1)$ and those of the Hessian $I - H(\theta)$ are $(1, -0.311088)$. The area of the contour plot around the stationary value is in figure 4.8

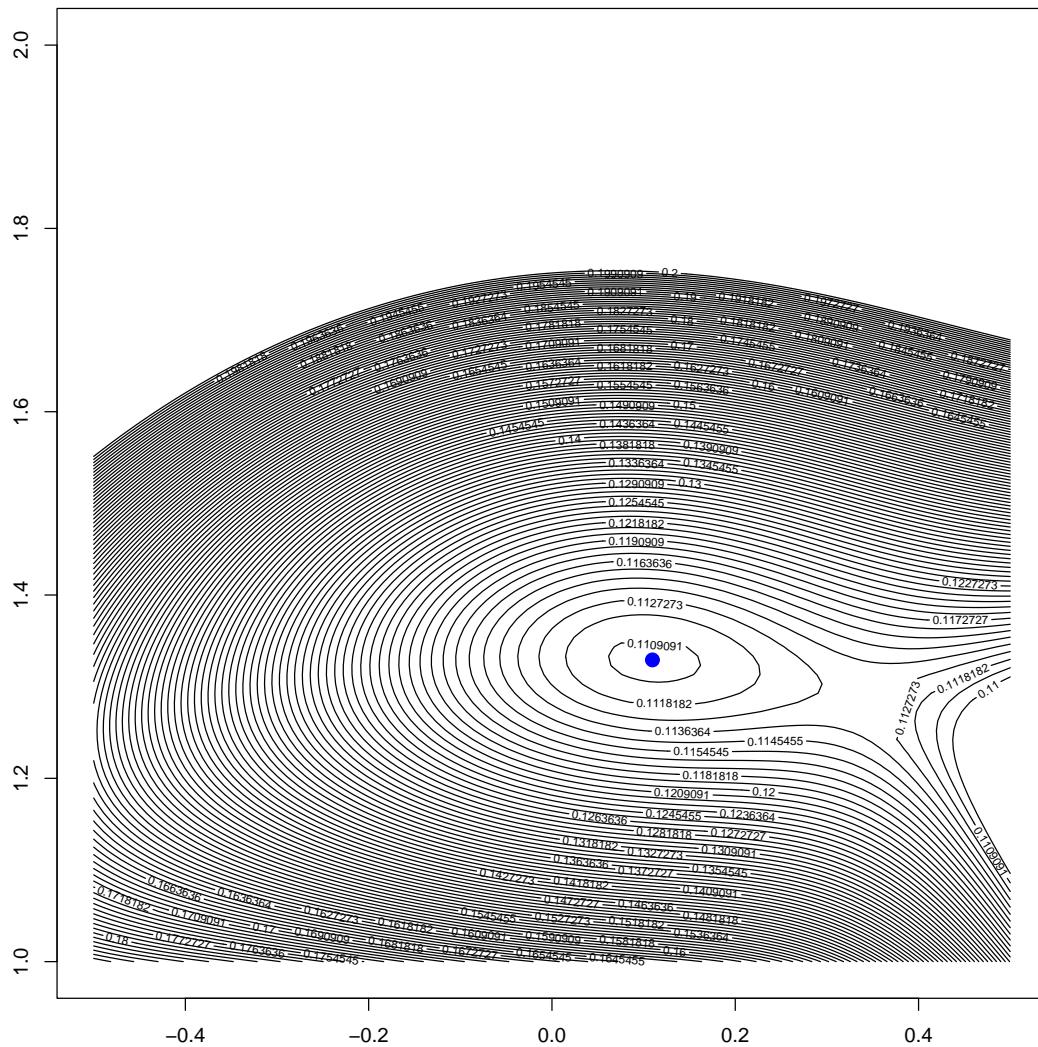


Figure 4.6: Contour Plot Third Minimum

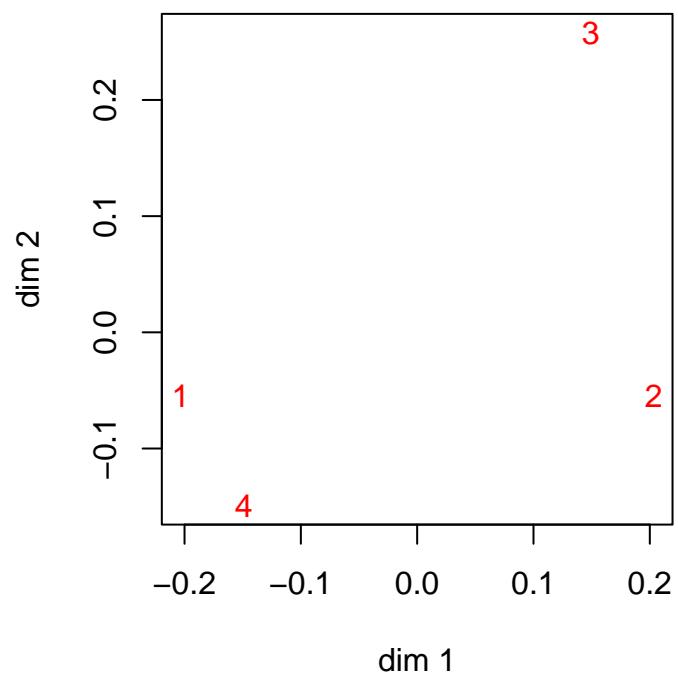


Figure 4.7: Configuration First Saddlepoint

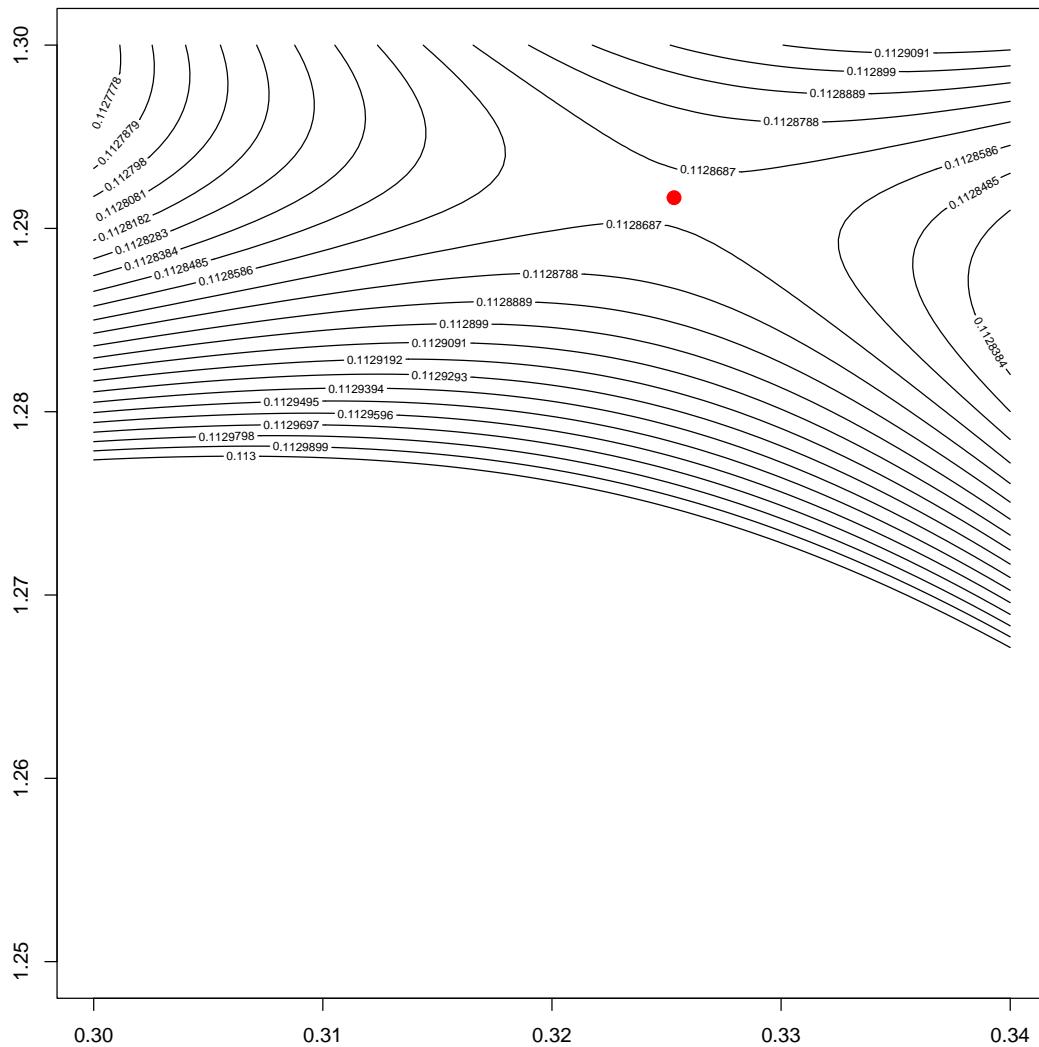


Figure 4.8: Contour First Saddlepoint

4.4.5 Second Saddle Point

The saddle point at $(1.1238371, 0.7762046)$ has stress 0.0672483 and the corresponding configuration is in figure 4.9

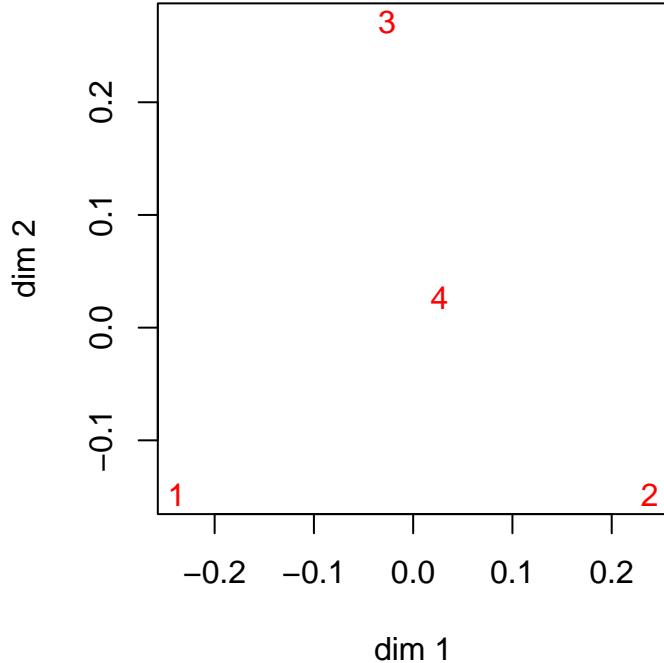


Figure 4.9: Configuration Second Saddlepoint

The eigenvalues of $B(\theta)$ are $(1.4111962, 1)$ and those of the Hessian $I - H(\theta)$ are $(1, -0.0841169)$. The area of the contour plot around the stationary value is in figure 4.10

4.5 Another Look

Remember that $\rho(\theta) = \theta' B(\theta) \theta$. Thus $\sigma(\lambda\theta) = 1 - \lambda\rho(\theta) + \frac{1}{2}\lambda^2\theta'\theta$, and

$$\min_{\lambda} \sigma(\lambda\theta) = 1 - \frac{1}{2} \frac{\rho^2(\theta)}{\theta'\theta}.$$

Thus we can minimize σ over θ by maximizing ρ over the unit circle $\mathcal{S} := \{\theta \mid \theta'\theta = 1\}$. This is a nice formulation, because ρ is norm, i.e. a homoge-

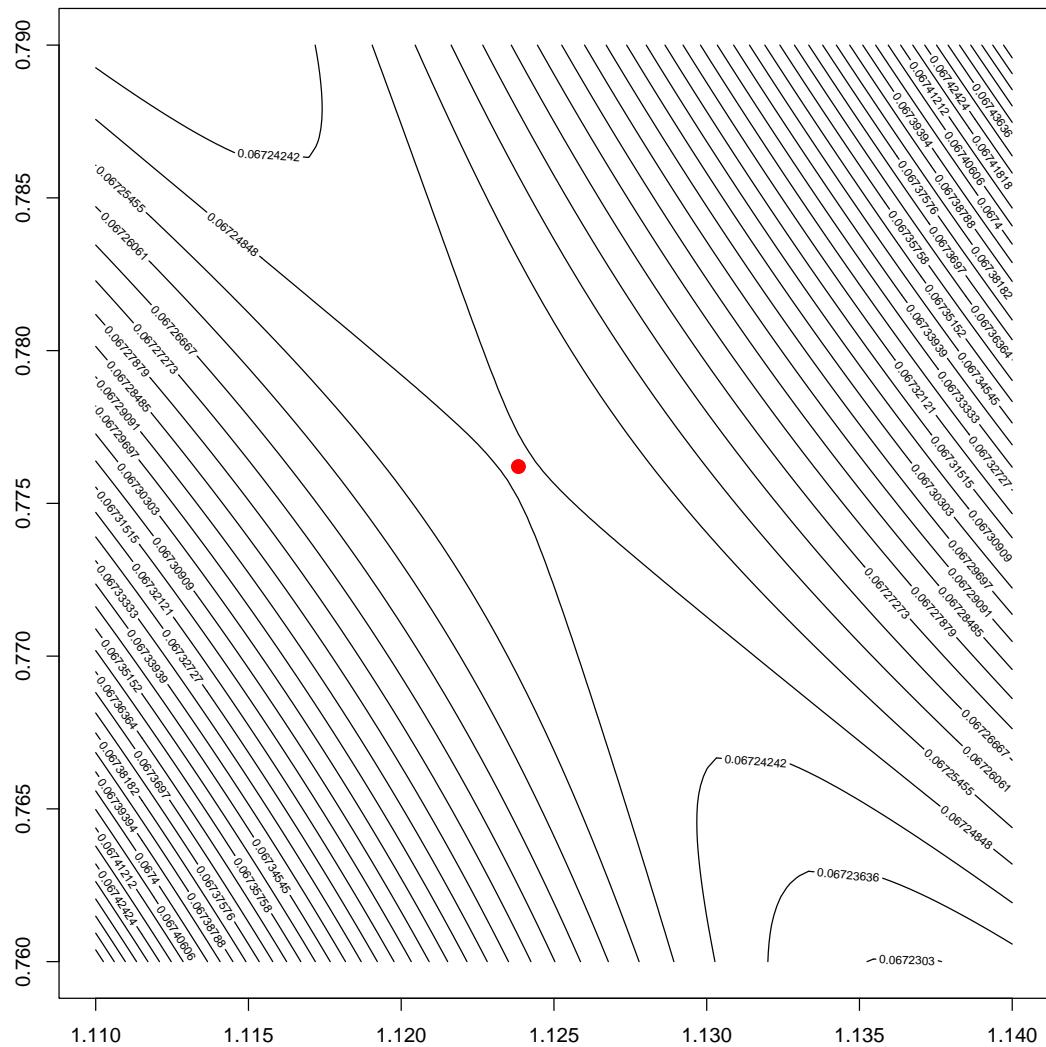


Figure 4.10: Contour Plot Second Saddlepoint

neous convex function of θ . Consequently we have transformed the problem from unconstrained minimization of the DC function (i.e. difference of convex functions) stress to that of maximization of a ratio of norms. In turn this is equivalent to maximization of the convex function ρ over the unit circle, or, again equivalently, over the unit ball, a compact convex set. This transform was first used in MDS by De Leeuw (1977a), partly because it made the theory developed by Robert (1967) available.

The levels sets $\{\theta \mid \rho(\theta) = \kappa\}$ are the ρ -circles defined by the norm ρ . The corresponding ρ -balls $\{\theta \mid \rho(\theta) \leq \kappa\}$ are closed and nested convex sets containing the origin. Thus we want to find the largest ρ -circle that still intersects \mathcal{S} . The similarity with the geometry of eigenvalue problems is obvious.

In our example we know that the global optimum of stress is at $(1.3938469, 0)$, and if we project that point on the circle it becomes $(1, 0)$. The corresponding optimal ρ is 1.3938469. Figure 4.11 gives the contourplot for ρ , with the outer ρ -circle corresponding with the optimal value. The fact that the optimal value contour is disjoint from the interior of \mathcal{S} is necessary and sufficient for global optimality (Dür, Horst, and Locatelli (1998)). Notice the sharp corners in the contour plot, showing the non-differentiability of ρ at the points where $d_{12}(\theta) = 0$. We could also look for the minimum of ρ on the unit circle, which means finding the largest ρ -circle that touches \mathcal{S} on the inside. Inspecting figure 4.11 shows that this will be a point where ρ is not differentiable, i.e. a point with $d_{12}(\theta) = 0$. This minimum ρ problem does not make much sense in the context of multidimensional scaling, however, and it is not related directly to the minimization of stress.

4.6 A Final Look

Now that we know that the MDS problem is equivalent to maximizing ρ on the unit circle, we can use nonlinear coordinates $(\theta_1, \theta_2) = (\sin \xi, \cos \xi)$ to reduce the problem to a one-dimensional unconstrained one in, say, “circle space”. Thus, with the same abuse of notation as for stress, $\rho(\xi) := \rho(\sin \xi, \cos \xi)$, and we have to maximize ρ over $0 \leq \xi \leq \pi$.

In figure 4.12 we have plotted ρ as a function of η . There are blue vertical lines at the three local minima in coefficient space, red vertical lines at the

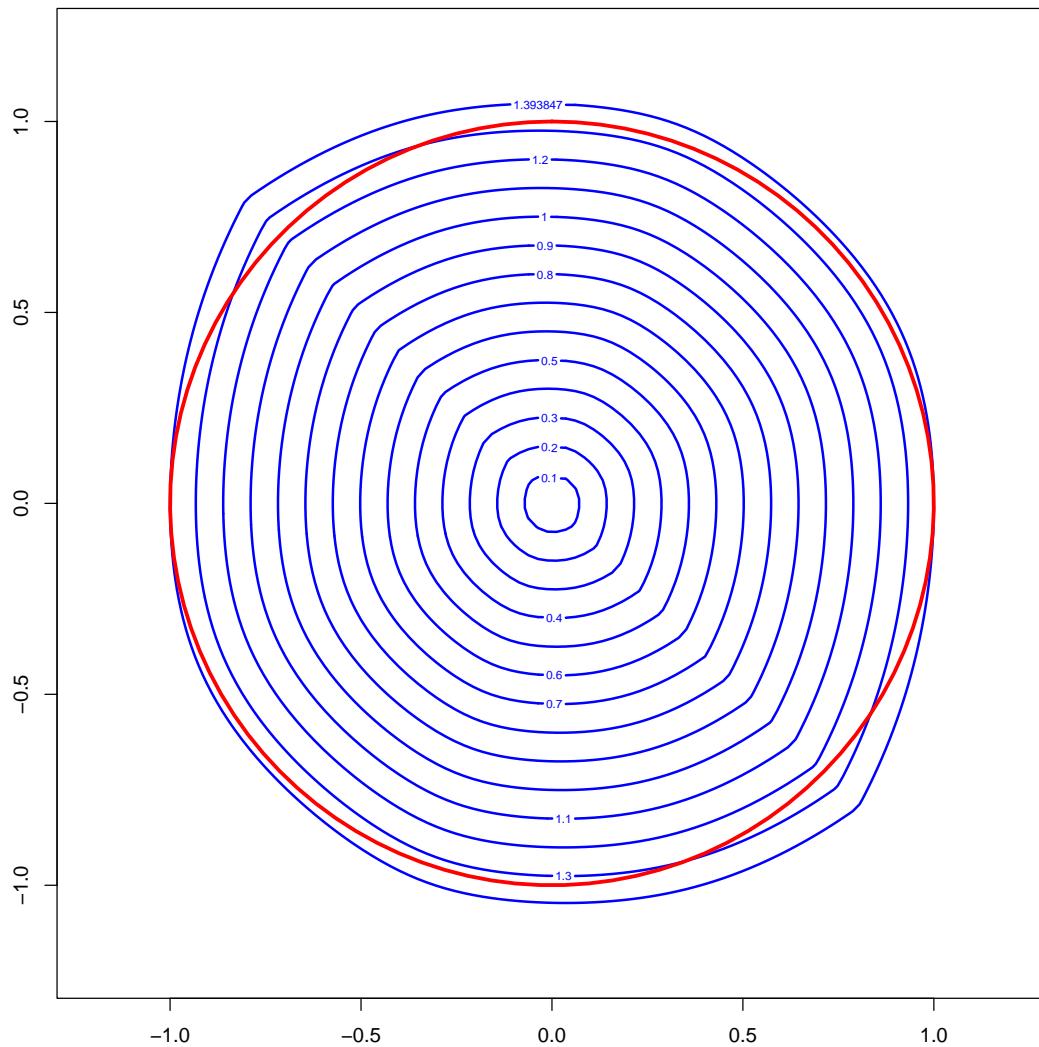


Figure 4.11: Contour Plot for Rho

stationary points, and a green vertical line where $d_{12}(\xi) = 0$. Note that in circle space stress has both multiple local minima and multiple local maxima.

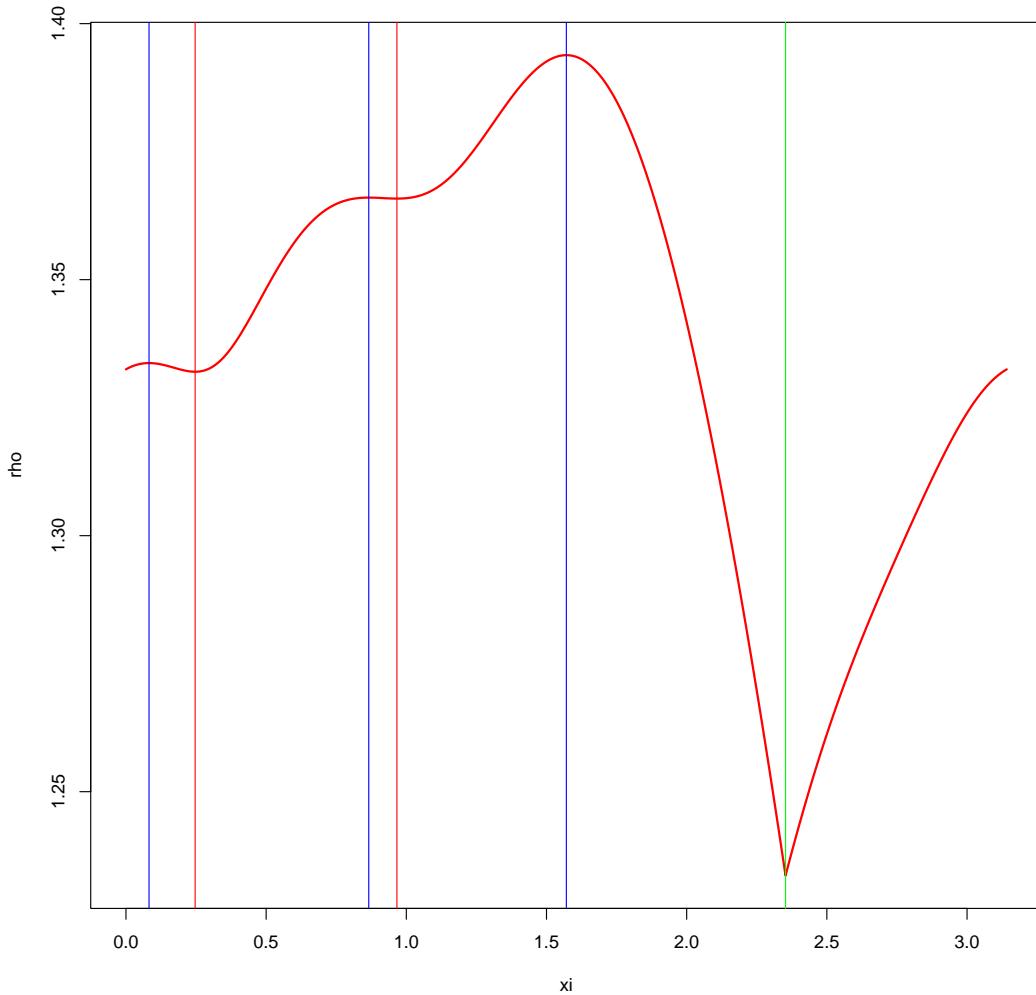


Figure 4.12: One-dimensional Rho

From lemma xxx we see that the second derivative $\mathcal{D}^2\rho(\xi)$ is equal to $\text{tr } H(\xi) - \rho(\xi)$. For the three local minima in coordinate space we find second derivatives 0, 0, 0 in circle space, i.e. they are properly converted to local maxima. The two stationary points in coordinate space have second derivatives 0, 0, and are turned into local minima.

For more general cases, with a basis of n configurations, we know from

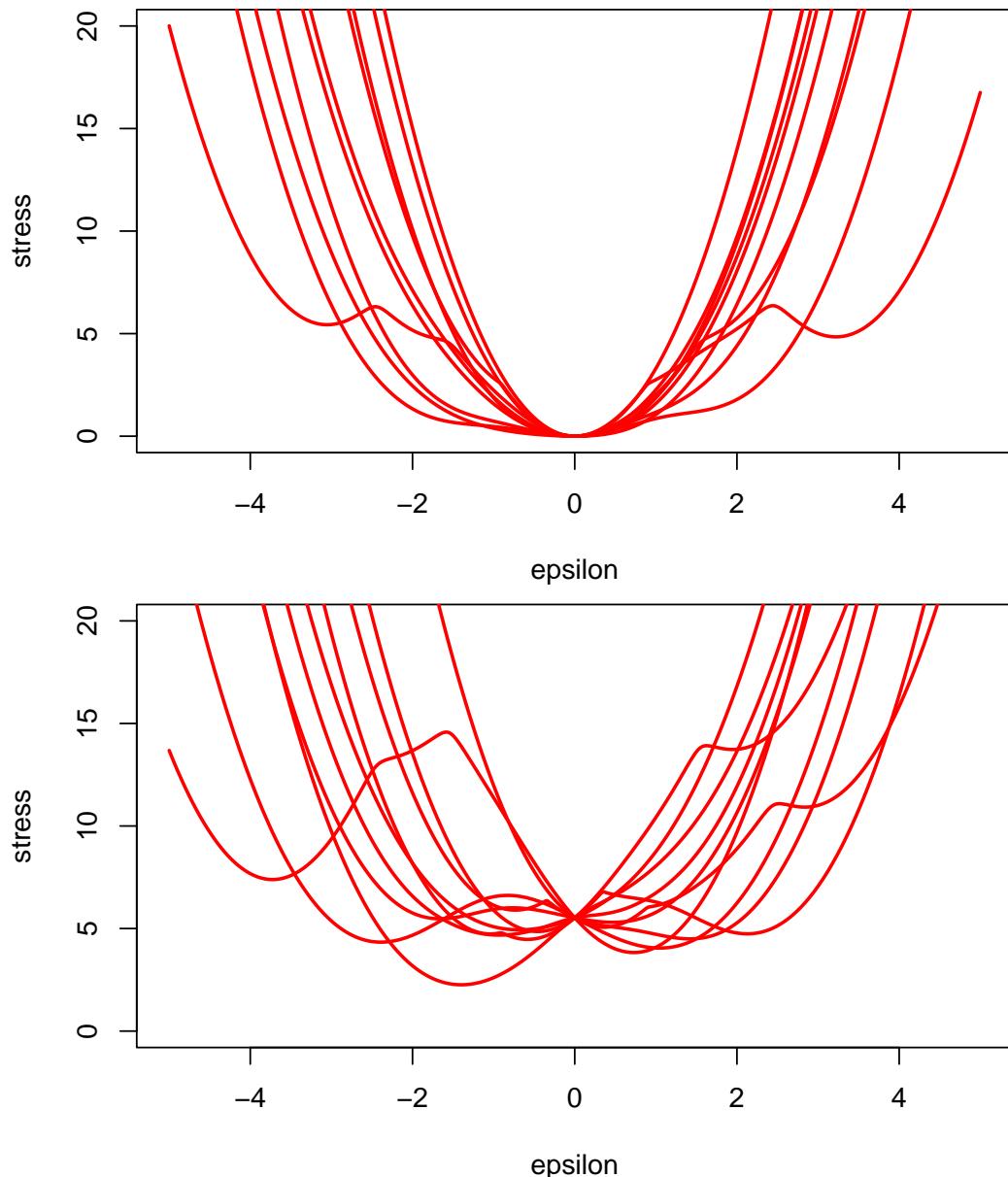
Lyusternik and Schnirelmann (1934) that a continuously differentiable even function on the unit sphere in \mathbb{R}^n has at least n distinct pairs of stationary points.

4.7 Discuss

Note that we have used σ for three different functions. The first one with argument Z is defined on *configuration space*, the second one with argument γ on *coefficient space*, and the third one with argument θ also on *coefficient space*. This is a slight abuse of notation, rather innocuous, but we have to keep it in mind.

From lemma xxx we see that $\mathcal{D}\sigma(X) = \mathcal{D}\sigma(Y) = 0$ then $\mathcal{D}\sigma(1,0) = \mathcal{D}\sigma(0,1) = 0$. Thus stationary points in configuration space are preserved as stationary points in coefficient space, but the reverse implication may not be true. If $\mathcal{D}^2\sigma(X)$ and $\mathcal{D}^2\sigma(Y)$ are positive semi-definite, then so are $\mathcal{D}^2\sigma(1,0)$ and $\mathcal{D}^2\sigma(0,1)$. Thus local minima are preserved. But it is entirely possible that $\mathcal{D}^2\sigma(X)$ and/or $\mathcal{D}^2\sigma(Y)$ are indefinite, and that $\mathcal{D}^2\sigma(1,0)$ and/or $\mathcal{D}^2\sigma(0,1)$ are positive semi-definite. Thus saddle points in configuration space can be mapped into local minima in coefficient space. As we will see this actually happens with Y , the equilateral triangle with center, in our example.

4.8 Coordinates



Let's look at a small example with four points, all dissimilarities equal, and all weights equal to one. There is a local minimum with four points in the corners of a square, with stress equal to 0.0285955. And there is another local minimum with three points forming an equilateral triangle, and the fourth

point in the center. This has stress 0.0669873. We can compute stress for all points of the form $\alpha X + \beta Y$, where X and Y are the two local minima. Figure 4.13 has a contour plot of $\sigma(\alpha, \beta)$, showing the local minima at $(1, 0)$ and $(0, 1)$, and the local maximum at $(0, 0)$.

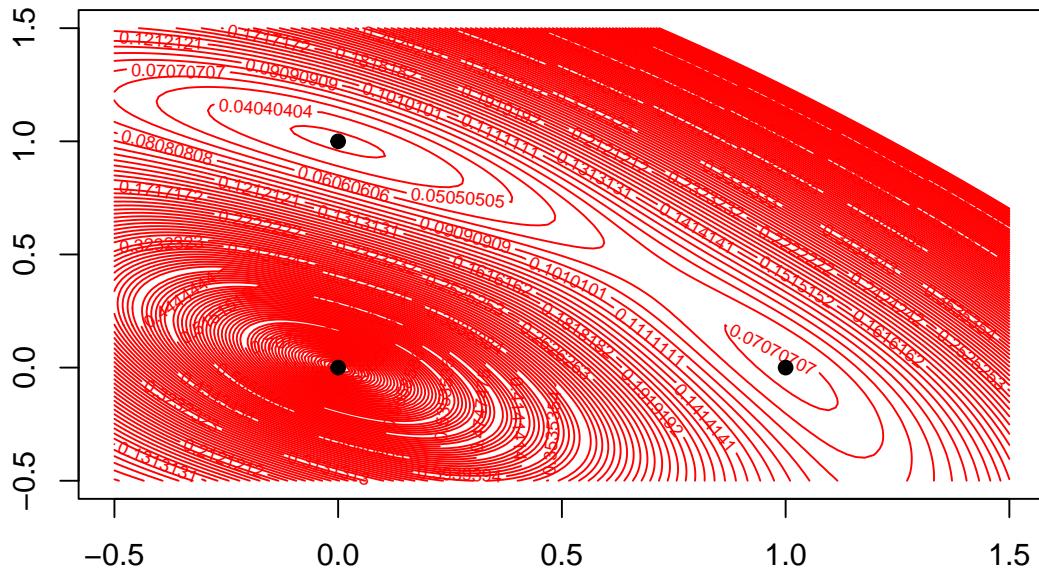


Figure 4.13: Plane spanned by two local minima, equal dissimilarities

Alternatively, we can plot stress on the line connecting X and Y . Note that although stress only has a local maximum at the origin in configuration space, it can very well have local maxima if restricted to lines. In fact on a line connecting two local minima there has to be at least one local maximum.

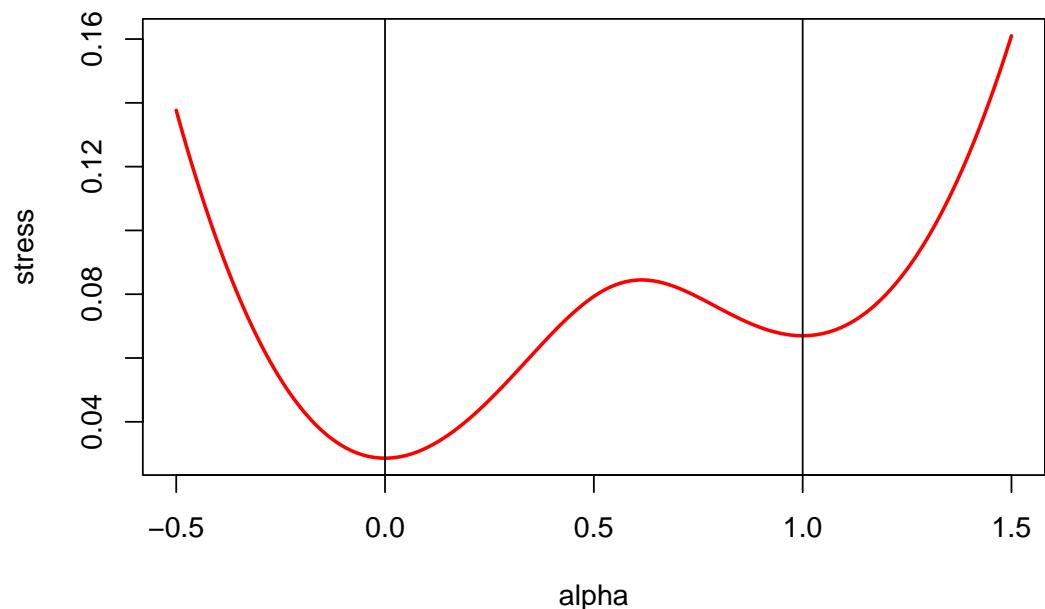


Figure 4.14: Line connecting two local minima, equal dissimilarities

Chapter 5

Classical Multidimensional Scaling

In the early days, exemplified by Messick and Abelson (1956), the key mathematical result used in MDS was Schoenberg's theorem (Schoenberg (1935)), which was made available to psychometricians by G. Young and Householder (1938). Statisticians came to the scene much later, not until Gower (1966).

Schoenberg's theorem gives necessary and sufficient conditions for a symmetric, hollow, and non-negative matrix of dissimilarities to be the distance matrix of n points in \mathbb{R}^p . Several variations of this basic theorem are possible (see Blumenthal (1953), chapter 43). We give the result in the form popularized by Torgerson (1958), with a simplified proof, using notation and terminology due to Critchley (1988).

Classical scaling can be thought of as an independent MDS method, in fact as the only MDS method available until the early 1960's. More commonly these days it is a method to provide a generally excellent starting point for iterative procedures to minimize stress.

5.1 Algebra

5.1.1 Torgerson Transform

The *Torgerson transform* of a matrix is a linear function from the space of real symmetric hollow matrices to the space of doubly-centered real symmetric matrices defined as

$$\tau(S) := -\frac{1}{2}JSJ, \quad (5.1)$$

with J the centering matrix $I - \frac{1}{n}ee'$. For historical reasons @eq:torgerson may be more familiar in elementwise notation. Spelled out it is

$$\tau_{ij}(S) = -\frac{1}{2}\{s_{ij} - s_{i\bullet} - s_{\bullet j} + s_{\bullet\bullet}\}, \quad (5.2)$$

where bullets are indices over which we average.

Some simple calculation with (5.2), using the hollowness of S , gives

$$\tau_{ii}(S) + \tau_{jj}(S) - 2\tau_{ij}(S) = s_{ij}. \quad (5.3)$$

Accordingly, Critchley (1988) defines a linear operator κ on the space of real symmetric matrices by $\kappa(S) := s_{ii} + s_{jj} - 2s_{ij}$. From (5.3) we see that $\kappa(\tau(S)) = S$. Also for all double-centered S we have $\tau(\kappa(S)) = S$. Thus τ and κ are mutually inverse (Critchley (1988), theorem 2.2).

5.1.2 Schoenberg's Theorem

Theorem 5.1. *There is an $X \in \mathbb{R}^{n \times p}$ such that $\Delta = D(X)$ if and only if $\tau(\Delta^{(2)})$ is positive semi-definite of rank $r \leq p$.*

Proof. If Δ are the distances of a column-centered p -dimensional configuration X , then the squared dissimilarities $\Delta^{(2)}$ are of the form $ue' + eu' - 2XX'$, where $u_i = x'_i x_i$ are the squared row lengths. This implies $\tau(\Delta^{(2)}) = XX'$, which is positive semi-definite of rank $r = \text{rank}(X) \leq p$.

Conversely, suppose $\tau(\Delta^{(2)}) = XX'$. Applying κ to both sides and using $\kappa(XX') = D^{(2)}(X)$ gives $\Delta^{(2)} = D^{(2)}(X)$. \square

Accordingly, we call a dissimilarity matrix Δ *Euclidean* if it is symmetric, hollow, and non-negative and has $\tau(\Delta^{(2)}) \gtrsim 0$. The *dimension* of a Euclidean dissimilarity matrix is the rank of $\tau(\Delta^{(2)})$, which is always less than or equal to $n - 1$.

5.2 Approximation

5.2.1 Low Rank Approximation of the Torgersen Transform

In classical MDS we minimize

$$\sigma(X) = \text{tr} \left\{ W(\tau(\Delta^{(2)}) - XX')W \right\}^2 \quad (5.4)$$

over $X \in \mathbb{R}^{n \times p}$, where W is a square non-singular matrix of weights. The minimizer X for the loss function in (5.4) is given by a slight variation of Eckart and Young (1936), discussed first by Keller (1962). See also De Leeuw (1974) and the generalizations to rectangular and singular weight matrices in De Leeuw (1984d). The solution is $X = W^{-1}K\Lambda$, where K are the normalized eigenvectors corresponding with the p largest eigenvalues of the positive part of $W(\tau(\Delta^{(2)})W)$ (i.e. the matrix that results by replacing the negative eigenvalues with zeroes). Thus the rank of the solution X is equal to the minimum of p and the number of positive eigenvalues of $\tau(\Delta^{(2)})$.

In the most classical versions of classical scaling (Torgerson (1952), Torgerson (1958)) there are no weights and the problem that the Torgerson transform may be indefinite is ignored. In fact, going from $\tau(\Delta^{(2)})$ to X is done by “any method of factoring,” including the centroid method, so no specific loss function is minimized.

The loss function (5.4) has been called, somewhat jokingly, *strain* by Takane, Young, and De Leeuw (1977), mainly because *stress* and *sstress* had already been taken. Stress is a weighted least squares loss function defined on the distances, *sstress* on the squared distances, and strain on the inner products.

5.2.2 Minimization of Strain

It may be considered a disadvantage of the classical approach, even if it is described as the minimization of strain, that the MDS equations are $\Delta^{(2)} = D^{(2)(X)}$ while loss is measured on the scalar products and not the distances.

It was first pointed out by De Leeuw and Heiser (1982) that strain can also be written as a loss function defined on the squared distances. In fact strain is equal to

$$\sigma(X) = \frac{1}{4} \text{tr} \left\{ W J(\Delta^{(2)} - D^{(2)}(X)) JW \right\}^2, \quad (5.5)$$

i.e. to an appropriately weighted version of sstress.

An advantage of the classical scaling approach via minimizing strain is that there is no local minimum problem. Finding the optimal configuration is an eigenvalue problem, which allows us to find the global minimum. This has not been emphasized enough, so I'll emphasize it here once again. If iterative basic MDS algorithms use the classical minimum strain solution as their starting point, then they start at the global minimum of a related loss function. Since they are descent algorithms they will improve their own loss functions, but having such an excellent starting point means they avoid many local minima.

Another advantage of the loss function formulation in @ref{eq:strainer} is that it is immediately obvious how to generalize classical scaling when there are missing data or when there is only rank order information. As with stress and sstress we minimize strain over both the configuration X and the missing information.

5.2.3 Approximation from Below

Suppose the Torgerson transform $\tau(\Delta^{(2)})$ is PSD of rank r , with eigen decomposition $K \Lambda^2 K'$, and, using the largest p eigenvalues with corresponding eigenvectors, define $C_p := K_p \Lambda_p^2 K'_p$. Then, in the Loewner order,

$$C_1 \lesssim C_2 \lesssim \cdots \lesssim C_r = \tau(\Delta^{(2)}).$$

Define $X_p = K_p \Lambda_p$. Then applying Critchley's inverse Torgerson transform κ it follows from ... that elementwise

$$D^{(2)}(X_1) \leq D^{(2)}(X_2) \leq \cdots \leq D^{(2)}(X_r) = \Delta^{(2)},$$

and thus also

$$D(X_1) \leq D(X_2) \leq \cdots \leq D(X_r) = \Delta.$$

Thus in the PSD case classical scaling we approximate the dissimilarities *from below*. This result is implicit in Gower (1966) and explicit in De Leeuw and Meulman (1986) and J. J. Meulman (1986).

Benzecri plot

Chapter 6

Minimization of Stress

6.1 Gradient Methods

The initial algorithms for nonmetric MDS Kruskal (1964b) and Guttman (1968) were gradient methods. Thus the gradient, or vector of partial derivatives, was computed in each iteration step, and a step was taken in the direction of the negative gradient (which is also known as the direction of steepest descent).

Informally, if f is differentiable at x then $f(x + h) \approx f(x) + h' \mathcal{D}f(x)$ and the direction h that minimizes the differential (the second term) is $-\mathcal{D}f(x)/\|\mathcal{D}f(x)\|$, the normalized negative gradient. Although psychometricians had been in the business of minimizing least squares loss functions in the linear and bilinear case, this result for general nonlinear functions was new to them. And I, and probably many others, hungrily devoured the main optimization reference in Kruskal (1964b), which was the excellent early review by Spang (1962).

Kruskal's paper also presents an elaborate step-size procedure, to determine how far we go in the negative gradient direction. In the long and convoluted paper Guttman (1968) there is an important contribution to gradient methods in basic MDS. Let's ignore the complications arising from zero distances, which is after all what both Kruskal and Guttman do as well, and assume all distances at configuration X are positive. Then stress is differentiable at X , with gradient

$$\nabla \sigma(X) = - \sum_{i=1}^n \sum_{j=1}^n w_{ij} (\delta_{ij} - d_{ij}(X)) \frac{1}{d_{ij}(X)} (e_i - e_j)(x_i - x_j)'$$

Geometrical interpretation - Gleason

Guttman C-matrix

Ramsay C-matrix

6.2 Initial Configurations

Random

L

Torgerson

Guttman

6.3 On MM Algorithms

The term *majorization* is used in mathematics in many different ways. In this book we use it as a general technique for the construction of *stable* iterative minimization algorithms. An iterative minimization algorithm is stable if it decreases the objective function in each iteration.

Ortega, Rheinboldt Weber Bohning, Lindsay Vosz, Eckart

Special cases of majorization had been around earlier, most notably the smacof algorithm for MDS and the EM algorithm for maximum likelihood with missing data, but in full generality majorization was first discussed in De Leeuw (1994) and Heiser (1995).

Majorization can be used to construct minimization methods, while minorization can construct maximization methods. This cleverly suggests to use the acronym MM algorithms for this class of techniques. An excellent comprehensive account of MM algorithms is Lange (2016). Another such account is slowly growing in one of the companion volumes in this series of personal research histories (De Leeuw (2016a)).

Here we just give a quick introduction to majorization. Suppose f is a real-valued function on $X \subseteq \mathbb{R}^n$. Then a real-valued function g on $X \otimes X$ is said to majorize f on X if

$$g(x, y) \geq f(x) \quad \forall (x, y) \in X \otimes X, \quad (6.1)$$

and

$$g(y, y) = f(y) \quad \forall y \in X. \quad (6.2)$$

Thus for each $y \in X$ the function $g(\bullet, y)$ lies above f , and it touches f from above at $x = y$. Majorization is *strict* if $g(x, y) = f(x)$ if and only if $x = y$, i.e. if y is the only point in X where $g(\bullet, y)$ and f touch.

A *majorization algorithm* to minimize f on X starts with an initial estimate, and then updates the estimate in iteration k by

$$x^{(k+1)} \in \operatorname{argmin}_{x \in X} g(x, x^{(k)}), \quad (6.3)$$

with the understanding that the algorithms stops when $x^{(k)} \in \operatorname{argmin}_{x \in X} g(x, x^{(k)})$.

If we do not stop we have an infinite sequence satisfying the *sandwich inequality*

$$f(x^{(k+1)}) \leq g(x^{(k+1)}, x^{(k)}) \leq g(x^{(k)}, x^{(k)}) = f(x^{(k)}). \quad (6.4)$$

The first inequality in this chain comes from (6.1). It is strict when majorization is strict. The second inequality comes from (6.3), and it is strict if $g(\bullet, y)$ has a unique minimum on X for each y .

6.4 Smacof Algorithm

The basic smacof algorithm, which is the main building block for most of the MDS techniques discussed in this book, updates the configuration $X^{(k)}$ in iteration k by

$$X^{(k+1)} = \mathfrak{G}(X^{(k)}) = V^+ B(X^{(k)}) X^{(k)}. \quad (6.5)$$

so that first $X^{(1)} = \mathfrak{G}(X^{(0)})$, then $X^{(2)} = \mathfrak{G}(X^{(1)}) = \mathfrak{G}(\mathfrak{G}(X^{(0)}))$, and generally $X^{(k)} = \mathfrak{G}^k(X^0)$, where \mathfrak{G}^k is the k-times composition (or iteration) of \mathfrak{G} .

We shall show in this chapter that as $k \rightarrow \infty$ both $\sigma(X^{(k+1)}) - \sigma(X^{(k)}) \rightarrow 0$, and $\eta^2(X^{(k)} - X^{(k+1)}) = (X^{(k+1)} - X^{(k)})' V(X^{(k+1)} - X^{(k)}) \rightarrow 0$. The iterations stop either if $\sigma(X^{(k)}) - \sigma(X^{(k+1)}) < \epsilon$ or if $\eta^2(X^{(k)} - X^{(k+1)}) < \epsilon$, where the ϵ are small cut-off numbers chosen by the user, or if we reach a user-defined maximum number of iterations, and we give up on convergence. The user also chooses if the stop criterion is based on function value changes or configuration changes.

Some quick remarks on implementation. We only have to compute V^+ once, but premultiplying by a full symmetric matrix in each iteration does add quite a few multiplications to the algorithm. If all w_{ij} are one, then $V^+ = \frac{1}{n} J$ and thus $\mathfrak{G}(X^{(k)}) = \frac{1}{n} B(X^{(k)}) X^{(k)}$. In fact we do not even have to carry out this division by n , because the basic algorithm is *self scaling*. which means in this context that $\mathfrak{G}(\alpha X) = \mathfrak{G}(X)$ for all $\alpha \geq 0$.

6.4.1 Global Convergence

The iterations in (6.5) start at some $X^{(0)}$ and then generate five sequences of non-negative numbers. Define $\lambda(X) := \rho(X)/\eta(X)$ and $\gamma(X) := \eta^2(X - \mathfrak{G}(X))$. The five sequences we will look at are

$$\begin{aligned} \sigma_k &:= \sigma(X^{(k)}), \\ \rho_k &:= \rho(X^{(k)}), \\ \eta_k &:= \eta(X^{(k)}), \\ \lambda_k &:= \lambda(X^{(k)}), \\ \gamma_k &:= \gamma(X^{(k)}), \end{aligned} \quad (6.6)$$

Depend on $X^{(0)}$

Zangwill

Argyros

6.4.2 From the CS Inequality

Theorem 6.1.

1. σ_k is a decreasing sequence, bounded below by 0.
2. ρ_k , η_k , and λ_k are increasing sequences, bounded above by 1.
3. γ_k is a null-sequence.

To prove convergence of these sequences we slightly modify and extend the proofs in De Leeuw (1977a) and De Leeuw and Heiser (1977) (while I say to myself: that's 44 years ago).

6.4.3 From Majorization

Smacof is based on the majorization, valid for all configurations X and Y ,

$$\sigma(X) \leq 1 + \eta^2(X - \mathfrak{G}(Y)) - \eta^2(\mathfrak{G}(Y)), \quad (6.7)$$

with equality if and only if $X \propto Y$. If $Y = \alpha X$ for some α then

$$\sigma(X) = 1 + \eta^2(X - \mathfrak{G}(Y)) - \eta^2(\mathfrak{G}(Y)), \quad (6.8)$$

and specifically we have (6.8) if $Y = X$.

Now suppose we have an Y with $Y \neq \mathfrak{G}(Y)$. If $\eta^2(X - \mathfrak{G}(Y)) \leq \eta^2(Y - \mathfrak{G}(Y))$ then

$$\sigma(X) \leq 1 + \eta^2(Y - \mathfrak{G}(Y)) - \eta^2(\mathfrak{G}(Y)) = \sigma(Y) \quad (6.9)$$

The obvious choice for X is $X = \mathfrak{G}(Y)$, which makes $\eta^2(X - \mathfrak{G}(Y)) = 0$, and thus

$$\sigma(X) \leq 1 - \eta^2(\mathfrak{G}(Y)) < 1 + \eta^2(Y - \mathfrak{G}(Y)) - \eta^2(\mathfrak{G}(Y)) = \sigma(Y) \quad (6.10)$$

6.4.4 Component Rotated Smacof

Consider the modified smacof iterations $\tilde{X}^{(k+1)} = X^{(k+1)}L^{(k+1)}$, where $L^{(k+1)}$ are the normalized eigenvectors of $\{X^{(k+1)}\}^T V X^{(k+1)}$. Then

$$\sigma(\tilde{X}^{(k)}) = \sigma(X^{(k)})$$

Thus the modified update produces the same sequence of stress values as the basic update. Also

$$\mathfrak{G}(\tilde{X}^{(k)}) = \mathfrak{G}(X^{(k)})L^{(k)}$$

Thus $\tilde{X}^{(k)}$ and $X^{(k)}$ differ by a rotation for each k . It follows that we can actually compute $\tilde{X}^{(k)}$ from the basic sequence $X^{(k)}$ by rotating the $X^{(k)}$ to principal components. Specifically if X_∞ is a subsequential limit of $X^{(k)}$ then the corresponding limit of $\tilde{X}^{(k)}$ is X_∞ rotated to principal components. Modifying the intermediate updates is just a waste of time, we can simply rotate the final smacof solution. And we should, as we explain in the next section, 6.4.5.

6.4.5 Local Convergence

$$\mathcal{D}\mathfrak{G}(X)(Y) = V^+ \left\{ B(X)Y - \sum_{1 \leq i < j \leq n} w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left(\frac{\text{tr } Y' A_{ij} X}{d_{ij}^2(X)} \right) A_{ij} \right\}$$

For any X : one zero eigenvalue $\mathcal{D}\mathfrak{G}(X)(X) = 0$

If on $\mathbb{R}^{n \times p}$ then an additional p zero eigenvalues $\mathcal{D}\mathfrak{G}(X)(e\mu^T) = 0$

For $X = \mathfrak{G}(X)$ and M anti-symmetric: $\frac{1}{2}p(p-1)E$ unit eigenvalues
 $\mathcal{D}\mathfrak{G}(X)(XM) = \mathfrak{G}(X)M = XM$

6.4.5.1 Rotation to PC

We suppose the configuration X is $n \times p$, with rank p . If the singular value decomposition is $X = K\Lambda L'$ then the rotation to principle components is $\Gamma(X) := K\Lambda = XL$. Thus $\mathcal{D}\Gamma(X)(Y) = YL + X\mathcal{D}L(X)(Y)$. So we need to compute $\mathcal{D}L(X)$, with L the right singular vectors of X , i.e. the eigenvectors of $X^T X$. We use the methods and results from De Leeuw (2007b), which were

applied to similar problems in De Leeuw (2008b), De Leeuw and Sorenson (2012), and De Leeuw (2016b).

Theorem 6.2. *If the $n \times p$ matrix has rank p , singular value decomposition $X = K\Lambda L^T$, with all singular values different, then $\Gamma(X + \Delta) = \Gamma(X) + \Delta L + XLM + o(\|\Delta\|)$, where M is antisymmetric with off-diagonal elements*

$$m_{ij} = \frac{\lambda_i s_{ij} + \lambda_j s_{ji}}{\lambda_i^2 - \lambda_j^2}. \quad (6.11)$$

Proof. The proof involves computing the derivatives of the singular value decomposition of X , which is defined by the equations

$$XL = K\Lambda, \quad (6.12)$$

$$X^T K = L\Lambda, \quad (6.13)$$

$$K^T K = I, \quad (6.14)$$

$$L^T L = LL^T = I. \quad (6.15)$$

We now perturb X to $X + \Delta$, which perturbs L to $L + L_\Delta + o(\|\Delta\|)$, K to $K + K_\Delta + o(\|\Delta\|)$, and Λ to $\Lambda + \Lambda_\Delta + o(\|\Delta\|)$. Substitute this into the four SVD equations for $X + \Delta$ and keep the linear terms.

$$XL_\Delta + \Delta L = K_\Delta \Lambda + K\Lambda_\Delta, \quad (6.16)$$

$$X^T K_\Delta + \Delta^T K = L_\Delta \Lambda + L\Lambda_\Delta, \quad (6.17)$$

$$L_\Delta^T L + L^T L_\Delta = 0, \quad (6.18)$$

$$K_\Delta^T K + K^T K_\Delta = 0. \quad (6.19)$$

Define $M := L^T L_\Delta$ and $N := K^T K_\Delta$. Then equations (6.18) and (6.19) say that M and N are antisymmetric. Also define $S := K^T \Delta L$. Premultiplying equation (6.16) by K^T and (6.17) by L^T gives

$$\Lambda M + S = N\Lambda + \Lambda_\Delta, \quad (6.20)$$

$$\Lambda N + S^T = M\Lambda + \Lambda_\Delta. \quad (6.21)$$

Either of these two equations gives, using the antisymmetry, and thus hollowness, of M and N , that $\Lambda_\Delta = \text{diag}(S)$. Define the hollow matrix $U := S - \text{diag}(S)$. Then

$$\Lambda M - N\Lambda = U, \quad (6.22)$$

$$\Lambda N - M\Lambda = U^T. \quad (6.23)$$

Premultiply (6.22) and postmultiply (6.23) by Λ .

$$\Lambda^2 M - \Lambda N \Lambda = \Lambda U, \quad (6.24)$$

$$\Lambda N \Lambda - M \Lambda^2 = U^T \Lambda. \quad (6.25)$$

If we add these two equations we can solve for the off-diagonal elements of M and find the expression in the theorem. Since $L_\Delta = LM$ this completes the proof. \square

6.4.6 Negative Dissimilarities

$$\sigma(X) = 1 - \sum_{k \in \mathcal{K}_{1+}} w_k \delta_k d_k(X) + \sum_{k \in \mathcal{K}_{1-}} w_k |\delta_k| d_k(X) + \frac{1}{2} \sum_{k \in \mathcal{K}} w_k d_k^2(X). \quad (6.26)$$

Split rho

Heiser (1991)

6.4.7 Unweighting

Consider the problem of minimizing a least squares loss function, defined as $f(x) := (x-y)'W(x-y)$ over x in some set X , where W is a symmetric weight matrix. Sometimes W complicates the problem, maybe because it is too big, too full, singular, or even indefinite. We will use iterative majorization to work around W . See also Kiers (1997) and Groenen, Giaquinto, and Kiers (2003).

Suppose z is another element of X . Think of it as the current best approximation to y that we have, which we want to improve. Then

$$\begin{aligned} f(x) &= (x - y)'W(x - y) \\ &= ((x - z) + (z - y))'W((x - z) + (z - y)) \\ &= f(z) + 2(x - z)'W(z - y) + (x - z)'W(x - z) \end{aligned} \quad (6.27)$$

Now choose a non-singular V such that $W \lesssim V$ and define $u := V^{-1}W(z - y)$. Then we have the majorization

$$f(x) \leq f(z) + 2(x - z)'W(z - y) + (x - z)'V(x - z) = f(z) + 2(x - z)'Vu + (x - z)'V(x - z) = f(z) + (x - (z - u))'V(x - z) \quad (6.28)$$

Here are some ways to choose V . We use $\lambda_{\max}(W)$ and $\lambda_{\min}(W)$ for the largest and smallest eigenvalues of the symmetric matrix W .

For any W we can choose $V = \lambda_{\max}(W)I$. Or, more generally, $V = \lambda_{\max}(D^{-1}W)D$ for any positive definite D . If W is singular we can choose $V = W + \epsilon D$ for any positive definite D . And in the unlikely case that W is indefinite we can choose $V = W + (\epsilon - \lambda_{\min}(W))I$. But if W is indefinite we have more serious problems.

In appendix A.1.18 the R function `lsuw()`, implements the iterative majorization algorithm minimizing $(x - y)'W(x - y)$ over x in some set X . One of the parameters of `lsuw()` is a function `proj()`, which projects a vector on X in the metric define by V . The projection could be on the positive orthant, on a cone with isotone vectors, on a linear subspace, on a sphere, on a set of low-rank matrices, and so on.

As an example choose W as a banded matrix of order 10 with $w_{ij} = 1$ if $|i - j| \leq 3$ and $i \neq j$, $w_{ii} = i$ if $i = j$, and $w_{ij} = 0$ otherwise. We require all 10 elements of x to be the same, and we use $V = \lambda_{\max}(W)I$ (the default).

The iterations are

```
w<-ifelse(outer(1:10,1:10,function(x,y) abs(x-y) <= 3),1,0)
w <- w + diag(0:9)
h1 <- lsuw(1:10, w, projeq)
```

If we use $\lambda_{\max}(D^{-1}W)D$ with $D = \text{diag}(W)$ for V we see the following majorization iterations.

```
d <- diag(w)
v <- max(eigen(1 / d) * w$values) * diag(d)
h2 <- lsuw(1:10, w, v = v, projeq)
```

So the second method of choosing V is a tiny bit less efficient in this case, but it really does not make much of a difference. In both cases x is 6.3009558, 6.3009558, 6.3009558, 6.3009558, 6.3009558, 6.3009558, 6.3009558, 6.3009558, 6.3009558 with function value 595.6699029.

Apply to stress and to

$$\sigma_c(X) := \sum_{1 \leq i < j \leq n} \sum_{1 \leq k < l \leq n} w_{ijkl} (\delta_{ij} - d_{ij}(X)) (\delta_{kl} - d_{kl}(X))$$

If $A \leq B$ (elementwise) then $\sum \sum (b_{ij} - a_{ij})(x_i - x_j)^2 \geq 0$ and thus $V(A) \lesssim V(B)$.

6.4.8 Smacof in Coefficient Space

[## Newton in MDS

6.5 Regions of Attraction

```
delta <- as.matrix (dist (diag (4)))
delta <- delta * sqrt (2 / sum (delta ^ 2))
```

6.5.1 Smacof

We use the smacof() function from the code in the appendix with 100 different starting points of θ , equally spaced on the circle. Figure 6.1 is a histogram of the number of smacof iterations to convergence within 1e-15.

In all cases smacof converges to a local minimum in coefficient space, never to a saddle point. Figure 6.2 shows which local minima are reached from the different starting points. This shows, more or less contrary to what Trosset and Mathar (1997) suggests, that non-global minima can indeed be points of attraction for smacof iterations.

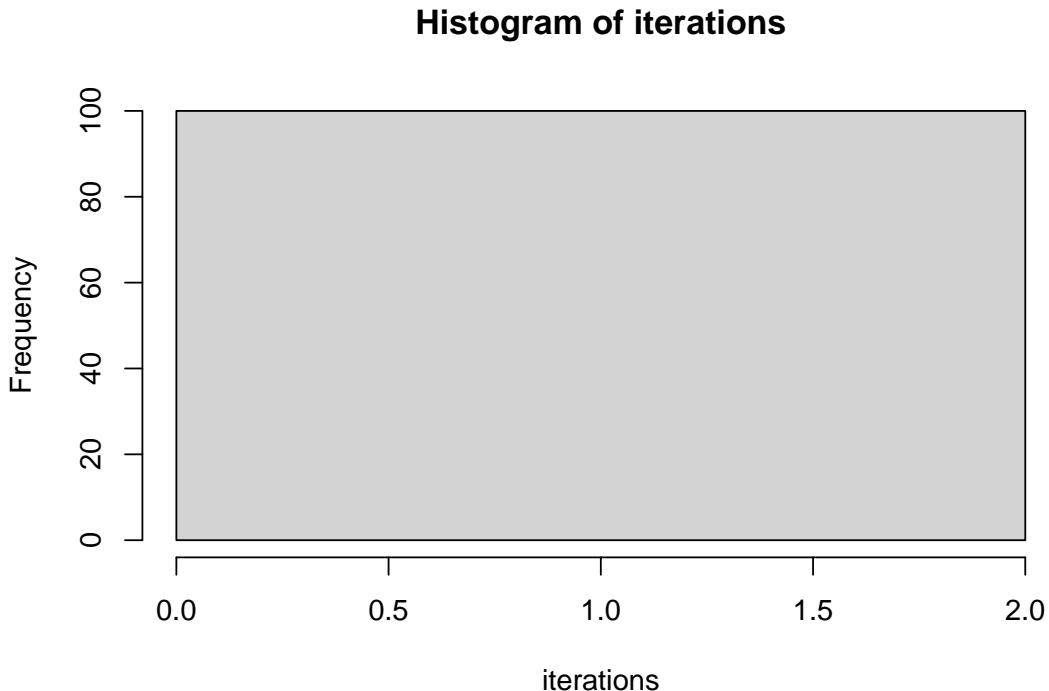


Figure 6.1: Histogram Number of Smacof Iterations

6.5.2 Newton

We repeat the same exercise with Newton's method, which converges from all 100 starting points. In higher dimensions we may not be so lucky. The histogram of iteration counts is in figure 6.3. It shows in this example that `smacof` needs about 10 times the number of iterations that Newton needs. Because `smacof` iterations are much less expensive than Newton ones, this does not really say much about computing times. If we look at figure 6.4 we see the problem with non-safeguarded Newton. Although we have fast

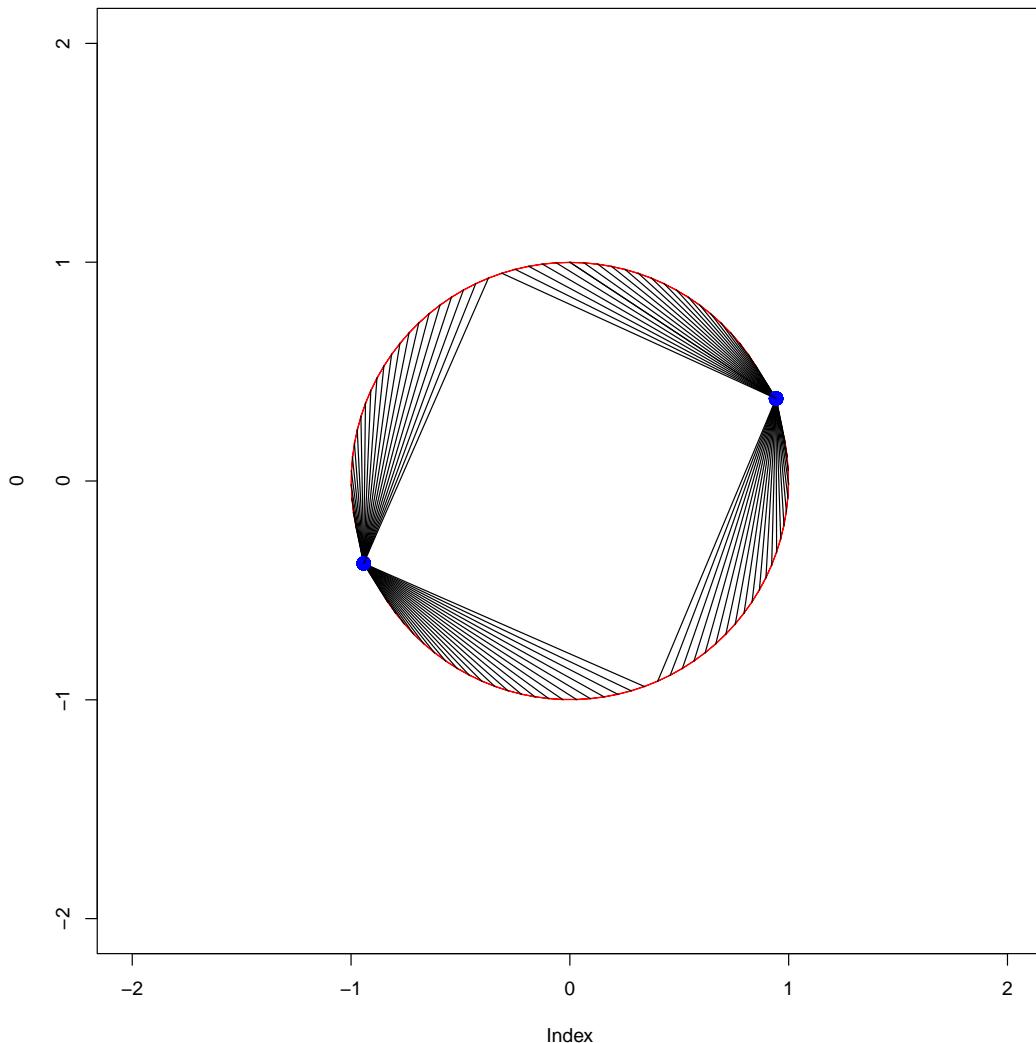


Figure 6.2: Path Endpoints of Smacof Iterations

convergence from all 100 starting points, Newton converges to a saddle point in 0 cases.

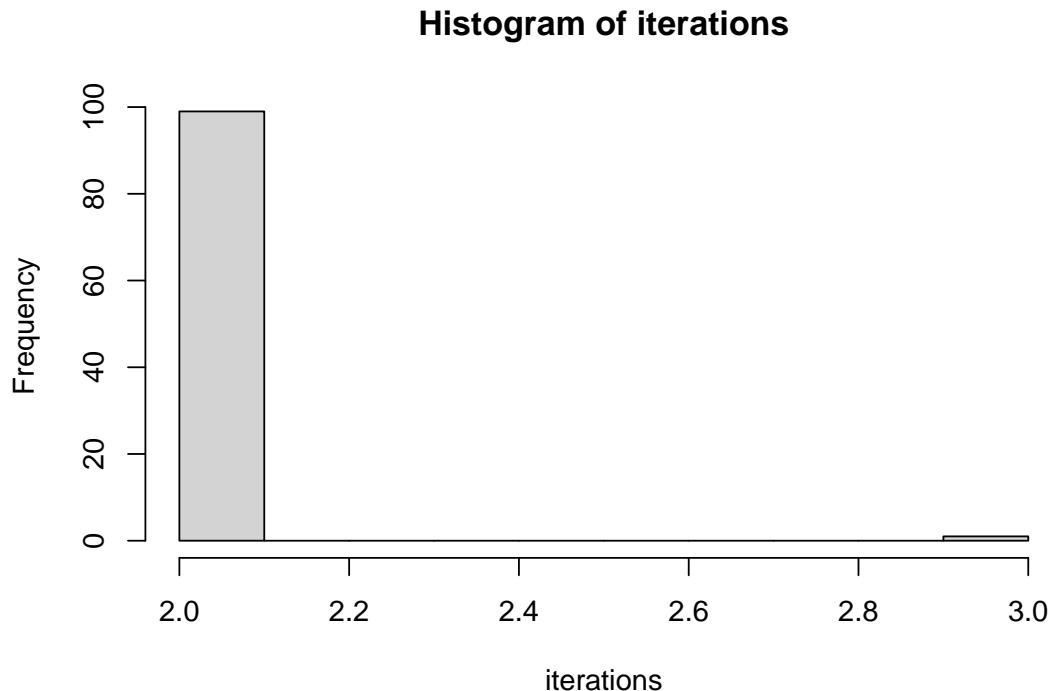


Figure 6.3: Histogram Number of Newton Iterations

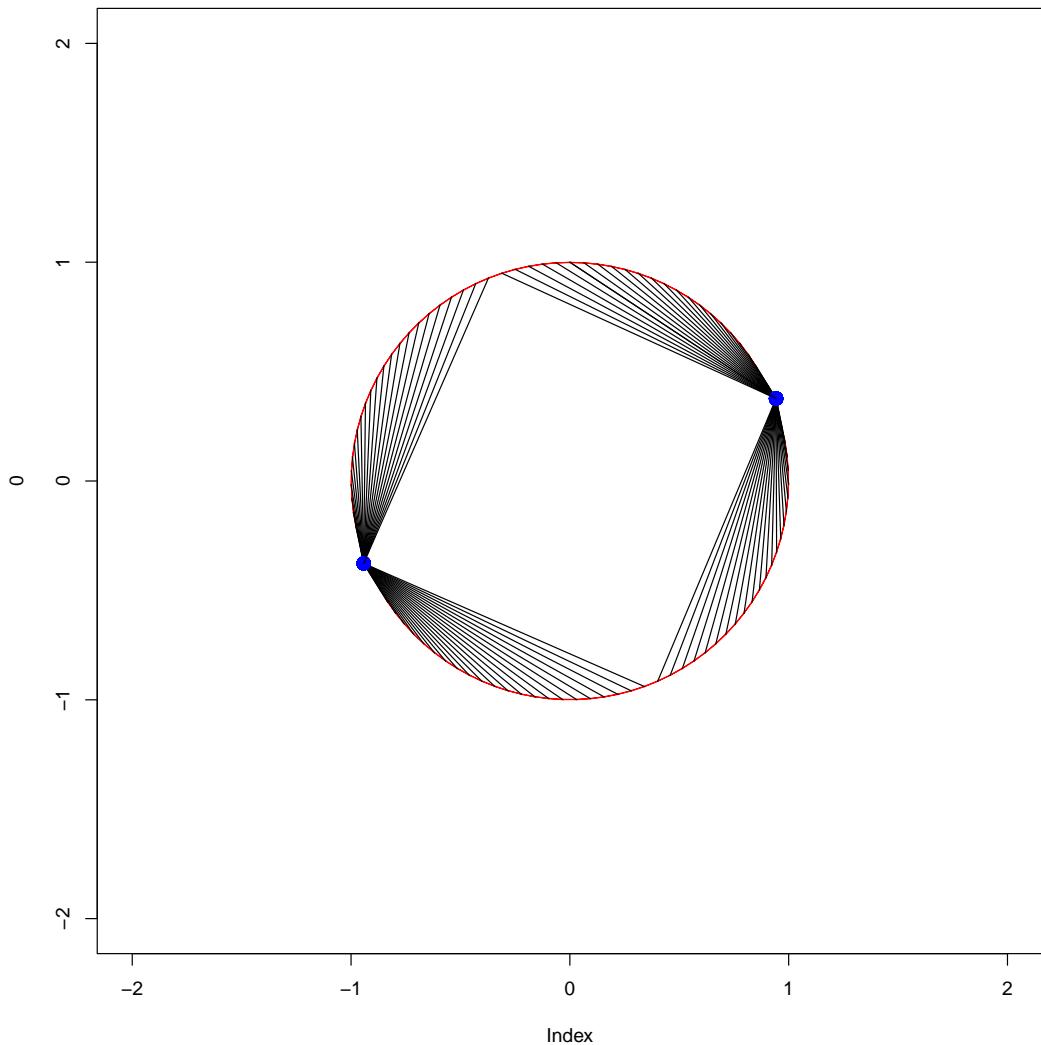


Figure 6.4: Path Endpoints of Newton Iterations



Figure 6.5: Jan de Leeuw, Gilbert Saporta, Yutaka Kanaka in Kolkata, December 1985

Chapter 7

Acceleration of Convergence

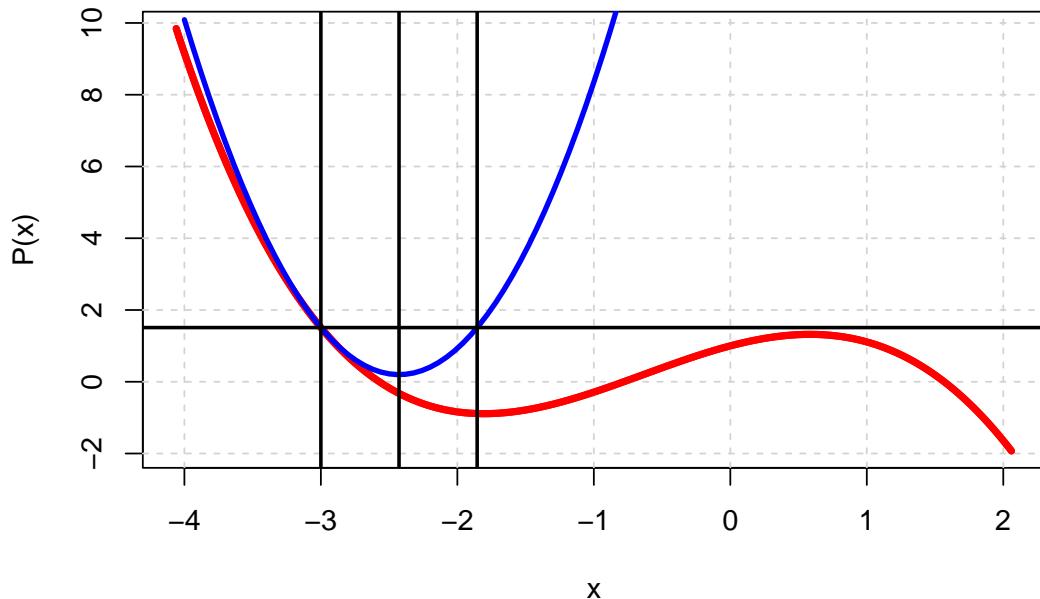
7.1 Simple Acceleration

A simple and inexpensive way to accelerate smacof iterations was proposed by De Leeuw and Heiser (1980).

On the other hand, if we choose $X = 2\mathfrak{G}(Y) - Y$ then again $X \neq Y$, but $\eta^2(X - \mathfrak{G}(Y)) = \eta^2(Y - \mathfrak{G}(Y))$. Thus

$$\sigma(X) \leq 1 + \eta^2(X - \mathfrak{G}(Y)) - \eta^2(\mathfrak{G}(Y)) = 1 + \eta^2(Y - \mathfrak{G}(Y)) - \eta^2(\mathfrak{G}(Y)) = \sigma(Y). \quad (7.1)$$

Let's define the two update rules $\text{up}_A(X) := \mathfrak{G}(X)$ and $\text{up}_B(X) = 2\mathfrak{G}(X) - X$.



This is illustrated in figure We want to locate a local minimum of f , in red, in the interval $(-4, 2)$. In this case we happen to know that f is a quartic polynomial, with minimum -0.8894476 at -1.8044048 . In the interval we are looking at we have $f''(x) \leq 8$. Suppose our initial guess for the location of the minimum is $x = -3$, the first vertical line from the left, with $f(-3)$ equal to 1.51. The upper bound on the second derivative allows us to construct a quadratic majorizer g , in blue, touching f at -3 . Update rule up_A tells us to go to the minimum of g , which is at -2.4275 , the second vertical line. Here g is equal to 0.198975 and f is -0.3245082 .

Rule up_B “overrelaxes” and goes all the way to -1.855 , the third vertical line from the left, where g is equal to both $g(-3)$ and $f(-3)$, and where f is -0.8862781 , indeed much closer to the minimum. Examples such as this make up_B really look good.

De Leeuw and Heiser give a rather informal theoretical justification of up_B as well. Suppose the sequence $X^+ = \mathfrak{G}(X)$ generated by up_A has slow linear convergence with ACR $1 - \epsilon$, where ϵ is positive and small. Then choosing the up_B will change the ACR of $1 - \epsilon$ to $2(1 - \epsilon) - 1 = 1 - 2\epsilon \approx (1 - \epsilon)^2$, and will approximately halve the number of iterations to convergence. This argument is supported by numerical experiments which seem to show that indeed about half the number of iterations are needed. It seems that up_B will get you something for almost nothing, and thus it has been implemented in

many versions of the smacof programs as the default update. Unfortunately this means that many users have obtained, and presumably reported, MDS results that are incorrect.

What is ignored in De Leeuw and Heiser (1980) is that majorization only guarantees that the sequence of loss function values converges for both update methods. The general convergence theory discussed earlier in this chapter shows that for both up_A and up_B the sequence $\{X^{(k)}\}$ has at least one accumulation point, and that the accumulation points of the sequence $\{X^{(k)}\}$ are fixed points of the update rule, which means for both up_A and up_B that at accumulation points X we have $X = \mathfrak{G}(X)$. But it does **not** say that $\{X^{(k)}\}$ converges.

The argument also ignores that at any X the derivative of up_A has a zero eigenvalue, with eigenvector X . For up_B the eigenvector X has eigenvalue equal to -1 , which is the largest one in modulus near any local minimum. And so ...

Suppose we have a configuration of the form αX with $X = \mathfrak{G}(X)$. Then $\text{up}_B(\alpha X) = 2\mathfrak{G}(\alpha X) - \alpha X = (2 - \alpha)X$ and $\text{up}_B((2 - \alpha)X) = \alpha X$. Thus starting with $X^{(1)} = \alpha X$ up_B generates a sequence with even members $(2 - \alpha)X$ and odd members αX . Thus there are two convergent subsequences with accumulation points αX and $(2 - \alpha)X$. And never the twain shall meet.

As far as stress is concerned, note that if $X = \mathfrak{G}(X)$ then $\sigma(\alpha X) = \sigma((2 - \alpha)X)$. Thus the stress values never change, and consequently form a convergent sequence.

We also see that $\text{up}_B^{(2)}(\alpha X) := \text{up}_B(\text{up}_B(\alpha X)) = \alpha X$, which means that αX is a fixed point of $\text{up}_B^{(2)}$ for any fixed point X of up_A and any α .

Another way to express the difference between the two update rule is that up_A is *self-scaling*, i.e. $\mathfrak{G}(\alpha X) = \mathfrak{G}(X)$, while up_B is not. Self-scaling implies $\mathcal{D}\mathfrak{G}(X)(X) = 0$, while for up_B $\mathcal{D}(2\mathfrak{G}(X) - X)(X) = -X$.

Let's now look at a real example. We use the Ekman color similarity data again, this time transformed by $\delta_{ij} = (1 - s_{ij})^3$. The analysis is in two dimensions, with no weights. We run four analyses, by crossing update rules up_A and up_B with stopping criteria $\sigma(X^{(k)}) - \sigma(X^{(k+1)}) < \epsilon$ and $\max_{i,s} |x_{is}^{(k)} - x_{is}^{(k+1)}| < \epsilon$. Let's call these stopping criteria *stop_f* and *stop_x*. In all cases we allow a maximum of 1000 iterations and we set ϵ to 1e-10.

	stop_f	stop_x		stop_f	stop_x
rule A	17	32	rule A	0.4696867	0.4696867
rule B	15	1000	rule B	0.6176456	0.6176456

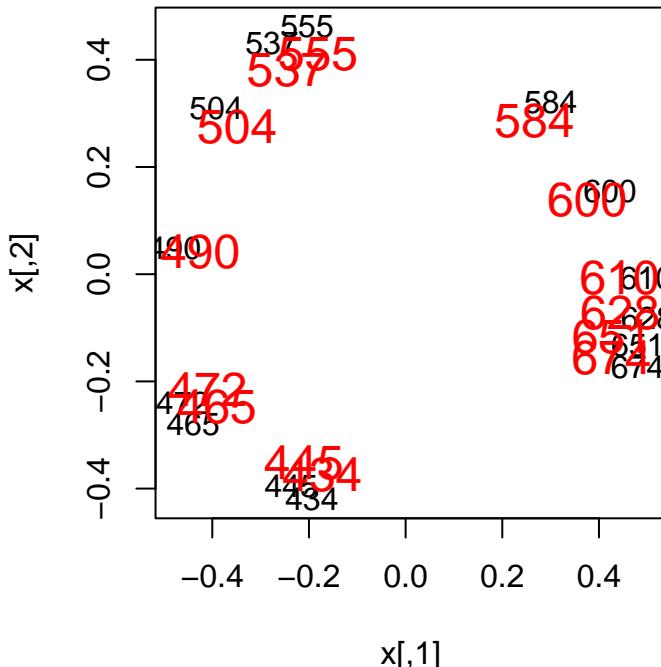
The results are in table ... The first subtable gives the number of iterations, the second the final stress value. We see that generally stop_x requires more iterations than stop_f, because it is a stricter criterion. If we use stop_x then up_B does not converge at all. Both with stop_f and stop_x up_B gives a higher stress value than up_A. And yes, with stop_f (which is the default stop criterion in all smacof programs so far) up_B use fewer iterations than up_A.

To verify that something is seriously wrong with running up_B, we compute the maximum absolute value of the gradient at convergence for both rules and stop_f. For up_A it is 0.0000000010 and for up_B it is 0.7834335001. Once again, with up_B both loss function and configuration converge to an incorrect value.

This can also be illustrated graphically. We see from table ... that up_B with stop_x ends after 1000 iteration. We perform an extra iteration, number 1001, and see how the configuration changes. In figure ... iteration 1000 is in black, iteration 1001 in red with slightly bigger characters. Except for a scaling factor the two configurations are the same. Elementwise dividing the up_B by the up_A final configuration gives a shrink factor α of 1.0595315. This shrink factor can also be computed from the final stress values. Using $\rho(X) = \eta^2(X)$ and $\sigma(X) = 1 - \eta^2(X)$ we find $\sigma(\alpha X) - \sigma(X) = (\alpha - 1)\eta^2(X)$, and thus

$$\alpha = 1 \pm \sqrt{\frac{\sigma(\alpha X) - \sigma(X)}{1 - \sigma(X)}}. \quad (7.2)$$

There are two values α and $2 - \alpha$, equal to 0.9595728 and 1.0404272, because the sequence has two accumulation points.



Things do not look good for up_B but simple remedies are available. The first one is renormalization. After the iterations, with say stop_f , have converged, we scale the configuration such that $\rho(X) = \eta^2(X)$ and recompute stress. This corrects both stress and the configuration to the correct outcome. Another way to normalize is to do another single up_A step after convergence of up_B . This has the same effect. We tried up_B with both renormalization approaches and both stof_f and stop_b . The number of up_B iterations is still the same as in table ... because we just compute something additional at the end. All stress values for the four combinations are now the correct 0.4696867. It seems that using up_B with stop_f and renorm at the end gives us the best of both worlds. It accelerates convergence and it gives the correct loss function values.

Of course up_B with stop_x still does not converge, and probably the best way to deal with that unfortunate fact is to avoid that combination. We can still use stop_x and get acceleration by define a single iteration as $\text{up}_{AB}(x) := \text{up}_A(\text{up}_B(x))$. For comparison purposes we also run $\text{up}_{AA}(x) := \text{up}_A(\text{up}_A(x))$. Both converge to the correct values, up_{AA} in 17 and $\text{up}_{AB}(x)$ in 10 iterations. Again up_{AB} is an attractive strategy. It works with both stop_f and stop_x and it accelerates. Less so than up_B , however. If the

ACR of up_A is $1 - \epsilon$, then, by the same reasoning as before, the ACR of up_{AB} is $(1 - \epsilon)^{\frac{3}{2}}$.

7.2 One-Parameter Methods

Ramsay

De Leeuw (2006)

7.3 SQUAREM

7.4 Vector extrapolation methods

De Leeuw (2008a)

De Leeuw (2008c)

Sidi

Chapter 8

Nonmetric MDS

8.1 Generalities

In non-metric MDS the dissimilarities are not a vector of known non-negative numbers, but they are only known up to a transformation or quantification. Ever since Kruskal (1964a) the approach for dealing with this aspect of the MDS problem is to define stress as a function of both X and Δ , and to minimize

$$\sigma(X, \Delta) := \sum_{1 \leq i < j \leq n} \sum w_{ij} (\delta_{ij} - d_{ij}(X))^2 \quad (8.1)$$

over both configurations X and feasible *disparities* (i.e. transformed dissimilarities). The name *disparities* was coined, as far as I know, by Forrest Young and used in our joint ALS work from the seventies (Takane, Young, and De Leeuw (1977)). Kruskal's name for the transformed or quantified dissimilarities is *pseudo-distances*.

To define a general notion of feasibility of disparities we use the notation $\Delta \in \mathfrak{D}$. Typically, although not necessarily, \mathfrak{D} is a convex set in disparity space. In interval, polynomial, splinical, and ordinal MDS it usually is a convex cone with apex at the origin. This implies that $0 \in \mathfrak{D}$, and consequently that

$$\min_{X \in \mathbb{R}^{n \times p}} \min_{\Delta \in \mathfrak{D}} = 0, \quad (8.2)$$

with the minimum attained at $X = 0$ and $\Delta = 0$. This is a trivial solution, independent of the data. Thus we cannot formulate the NMDS problem as the minimization of stress from equation (8.1) over unconstrained X and over Δ in its cone. We need some way to exclude either $X = 0$ or $\Delta = 0$, or both, from the feasible solutions. This we can do either by normalization of the loss function, or by using constraints that explicitly exclude one or both zero solutions. The commonly used options will be discussed in section 8.4.1 of this chapter.

8.2 Single and Double Phase

The distinction between *single phase* and *double phase* NMDS algorithms, names we borrow from Guttman (1968) (see also Lingoes and Roskam (1973)), has caused a great deal of confusion in the early stages of non-metric MDS (say between 1960 and 1970). Because I have been an active participant (with Carroll, Kruskal, Guttman, Young, and Lingoes) in discussing and perhaps perpetuating this confusion (De Leeuw (1973b)) I will pay some extra attention to it.

By the very definition of the minimum of a function we have the mathematical truism

$$\min_{(X,\Delta) \in \mathfrak{X} \otimes \mathfrak{D}} \sigma(X, \Delta) = \min_{X \in \mathfrak{X}} \min_{\Delta \in \mathfrak{D}} \sigma(X, \Delta) = \min_{X \in \mathfrak{X}} \left\{ \min_{\Delta \in \mathfrak{D}} \sigma(X, \Delta) \right\} = \min_{\Delta \in \mathfrak{D}} \left\{ \min_{X \in \mathfrak{X}} \sigma(X, \Delta) \right\}, \quad (8.3)$$

no matter what the subsets \mathfrak{X} of configuration space and \mathfrak{D} of disparity space are.

8.2.1 Double Phase

In a *double phase algorithm* we alternate the minimization of stress over X and Δ . Thus

$$X^{(k+1)} = \underset{X \in \mathfrak{X}}{\operatorname{argmin}} \sigma(X, \Delta^{(k)}), \quad (8.4)$$

$$\Delta^{(k+1)} = \underset{\Delta \in \mathfrak{D}}{\operatorname{argmin}} \sigma(X^{(k+1)}, \Delta).. \quad (8.5)$$

Thus double phase algorithms are *alternating least squares* or ALS algorithms. The designation “alternating least squares” was first used, AFAIK, by De Leeuw (1968b), and of course it was widely disseminated by the series of ALS algorithms of Young, Takane, and De Leeuw in the seventies (see F. W. Young (1981) for a retrospective summary).

There are some possible variations in the ALS scheme. In equation (8.4) we update X first, and then in equation (8.4) we update Δ . That order can be reversed without any essential changes. More importantly, we have to realize that minimizing over X in equation (8.4) is a basic metric MDS problem, which will generally take an infinite number of iterations for an exact solution. This means we have to truncate the minimization, and stop at some point. And, in addition, equation (8.4) implies we have to find the global minimum over X , which is generally infeasible as well. Thus the ALS scheme as defined cannot really be implemented.

We remedy this situations by switching from minimization in each substep to a decrease, or, notationwise, from argmin to $\operatorname{arglower}$. The resulting update sequence

$$X^{(k+1)} = \underset{X \in \mathfrak{X}}{\operatorname{arglower}} \sigma(X, \Delta^{(k)}), \quad (8.6)$$

$$\Delta^{(k+1)} = \underset{\Delta \in \mathfrak{D}}{\operatorname{arglower}} \sigma(X^{(k+1)}, \Delta).. \quad (8.7)$$

is much more loosely defined as the previous one, because $\operatorname{arglower}$ can be implemented in many different ways. More about that later. But at least the new scheme can actually be implemented.

Algorithm #ref(eq:nmslte1), #ref(eq:nmslte1) is still considered to be ALS, but it is also firmly in the class of *block relaxation* algorithms. General block relaxation, which has alternating least squares, coordinate relaxation, augmentation, EM, and majorization as special cases, was used to describe many data analysis algorithms in De Leeuw (1994). As with ALS, special cases of block relaxation have been around for a long time.

8.2.2 Single Phase

From equation (8.3)

$$\min_{X \in \mathfrak{X}} \min_{\Delta \in \mathfrak{D}} \sigma(X, \Delta) = \min_{X \in \mathfrak{X}} \left\{ \min_{\Delta \in \mathfrak{D}} \sigma(X, \Delta) \right\}. \quad (8.8)$$

So if we define

$$\sigma_*(X) := \min_{\Delta \in \mathfrak{D}} \sigma(X, \Delta), \quad (8.9)$$

the NMDS problem is to minimize σ_* from (8.9) over X . Note there is a σ defined by equation (8.1) on $\mathfrak{X} \otimes \mathfrak{D}$, and a σ_* , defined by equation (8.9), which is a function only of X . It is sometimes said that that Δ is *projected* when going from (8.1) to (8.9), or that σ_* is a *marginal* function.

Once more with feeling. The two-phase σ is a function of two matrix variables X and Δ , the one-phase σ_* is a function of the single matrix variable X . To make this even more clear we can write $\sigma_*(X) = \sigma(X, \Delta(X))$, where

$$\Delta(X) := \operatorname{argmin}_{\Delta \in \mathfrak{D}} \sigma(X, \Delta). \quad (8.10)$$

Of course by projecting out X instead of Δ we could also have defined a loss function which is a function of Δ only, but typically we do not use that alternative projection because it is complicated and heavily nonlinear. Projecting out X is, in fact, solving a standard basic MDS problem. Projecting out Δ is usually much simpler. In most applications \mathfrak{D} is convex, so computing $\Delta(X)$ is computing the projection on a convex set, and projections on convex sets always exist and are unique.

As an aside, projection creates a function of one variable out of a function of two variables. The inverse of projection is called *augmentation*, which starts with a function f of one variable on \mathfrak{X} and tries to find a function of two variables g on $\mathfrak{X} \otimes \mathfrak{Y}$ such that $f(x) = \min_{y \in \mathfrak{Y}} g(x, y)$. If we have found such a g then we can minimize f over \mathfrak{X} by minimizing g over $\mathfrak{X} \otimes \mathfrak{Y}$, for example by block relaxation (De Leeuw (1994)).

One reason there was some confusion, and some disagreement between Kruskal and Guttman, was a result on differentiation of the minimum

function, which was not known in the psychometric community at the time. Guttman thought that σ_* was not differentiable at X , because Δ from (8.10) is a step function. Kruskal proved in Kruskal (1971) that σ_* is differentiable, and saw that the result is basically one in convex analysis, not in classical linear analysis. The result follows easily from directional differentiability in Danskin's theorem (Danskin (1967)) or from the minimax theorems of, for example, Demyanov and Malozemov (1990), using the fact that the projection is unique. More directly, deleeuw_R_73g refers to discussion on page 255 of Rockafellar (1970), following his corollary 26.3.2. We will go into more detail about differentiability, and the differences between Kruskal's and Guttman's loss functions, in the next chapter 11. For now it suffices to note that

$$\mathcal{D}\sigma_*(X) = \mathcal{D}_1\sigma(X, \Delta(X)), \quad (8.11)$$

or, in words, that the derivative of σ_* at X is the partial derivative of σ at $(X, \Delta(X))$.

8.3 Affine NMDS

Basic MDS can now be interpreted as the special case of NMDS in which $\mathfrak{D} = \{\Delta\}$ is a singleton, a set with only one element. Thus $0 \notin \mathfrak{D}$ and we do not have to worry about trivial zero solutions for X .

This extends to missing data basic MDS. We have so far dealt with missing data by setting the corresponding w_{ij} equal to zero. But for the non-missing part we still have fixed numbers in Δ , and thus again $0 \notin \mathfrak{D}$ (unless all dissimilarities are missing). In a sense missing data are our first example of non-metric MDS, because \mathfrak{D} can also be defined as the set

$$\mathfrak{D} = \left\{ \Delta \mid \Delta_0 + \sum_{1 \leq i < j \leq n} \sum \{ \alpha_{ij}(E_{ij} + E_{ji}) \mid \delta_{ij} \text{ is missing} \} \right\}, \quad (8.12)$$

where the E_{ij} are the unit matrices defined in section 25.2.3 and Δ_0 is the non-missing part (which has zeroes for the)

Another example in which $0 \notin \mathfrak{D}$ is the additive constant problem, which we will discuss in detail in section 9.1. Here \mathfrak{D} is the set of all hollow and symmetric matrices of the form $\Delta + \alpha(E - I)$, where the dissimilarities in Δ_0 are known real numbers and where α is the unknown additive constant.

The affine MDS problems also have single phase and double phase algorithms. For missing data single phase stress is

$$\sigma_{\star}(X) = \min_{\Delta \in \mathfrak{D}} \sum_{1 \leq i < j \leq n} w_{ij}(\delta_{ij} - d_{ij}(X))^2 = \sum_{1 \leq i < j \leq n} \tilde{w}_{ij}(\delta_{ij} - d_{ij}(X))^2, \quad (8.13)$$

where $\tilde{w}_{ij} = 0$ if δ_{ij} is missing, and $\tilde{w}_{ij} = w_{ij}$ otherwise. In this case $\sigma_{\star}(X) = \sigma(X)$, the sigma of basic MDS with zero weights for missing data.

For the additive constant problem single phase stress is

$$\sigma_{\star}(X) = \min_{\alpha} \sum_{1 \leq i < j \leq n} w_{ij}(\delta_{ij} + \alpha - d_{ij}(X))^2 = \sum_{1 \leq i < j \leq n} w_{ij}(\delta_{ij} - d_{ij}(X))^2 - (\bar{\delta} - \bar{d}(X))^2 \sum_{1 \leq i < j \leq n} w_{ij}, \quad (8.14)$$

where $\bar{\delta}$ and $\bar{d}(X)$ are the weighted means of the dissimilarities and distances.

8.4 Conic MDS

8.4.1 Normalization

In “wide-sense” non-metric MDS \mathfrak{D} can be any set of hollow, non-negative and symmetric matrices. In “narrow-sense” non-metric MDS \mathfrak{D} is defined by homogeneous linear inequality constraints of the form $\delta_{ij} \leq \delta_{kl}$ (in addition to hollow, non-negative, and symmetric). These constraints, taken together, define a polyhedral convex cone in disparity space. This just means that if Δ_1 and Δ_2 are in \mathfrak{D} then so is $\alpha\Delta_1 + \beta\Delta_2$ for all non-negative α and β .

The disparities are a cone, and thus $0 \in \mathfrak{D}$. This implies that always $\min_X \min_{\Delta \in \mathfrak{D}} \sigma(X, \Delta) = 0$, independently of the data. This is our first example of a **trivial solution**, which have plagued non-metric scaling from

the start. Note that \mathfrak{D} for missing data and for the additive constant are not convex cones, and do not contain the zero matrix.

In our versions of *non-metric* MDS we actually require that the transformed dissimilarities satisfy $\eta_\delta = 1$, so that formula (2.6) is still valid. We call this **explicit normalization of the dissimilarities**.

To explain the different forms of normalization of stress that are needed whenever \mathfrak{D} is a cone we look at some general properties of least squares loss functions. More details are in Kruskal and Carroll (1969) and in De Leeuw (1975a), De Leeuw (2019).

Suppose K and L are cones in \mathbb{R}^n , nor necessarily convex. Our problem is to minimize $\|x - y\|^2$ over both $x \in K$ and $y \in L$. Here $\|x\|^2 = x'Wx$ for some positive definite W . In the MDS context, for x think disparities, for y think distances.

Of course minimizing $\|x - y\|^2$ is too easy, because $x = y = 0$ is the (trivial, and useless) solution. So we need some form of normalization. We distinguish six different ones.

1. implicit x-normalization

$$\min_{x \in K} \min_{y \in L} \frac{\|x - y\|^2}{\|x\|^2}$$

2. implicit y-normalization

$$\min_{x \in K} \min_{y \in L} \frac{\|x - y\|^2}{\|y\|^2}$$

3. implicit xy-normalization

$$\min_{x \in K} \min_{y \in L} \frac{\|x - y\|^2}{\|x\|^2 \|y\|^2}$$

4. explicit x-normalization

$$\min_{x \in K \cap S} \min_{y \in L} \|x - y\|^2$$

5. explicit y-normalization

$$\min_{x \in K} \min_{y \in L \cap S} \|x - y\|^2$$

6. explicit xy-normalization

$$\min_{x \in K \cap S} \min_{y \in L \cap S} \|x - y\|^2$$

If we use a positive definite W to define our inner products and norms, then implicit normalization of x means

$$\min_{x \in X} \min_{y \in Y} \frac{(x - y)' W (x - y)}{x' W x}.$$

Let \mathcal{S}_x and \mathcal{S}_y be the ellipsoids of all x with $x' W x = 1$ and of all y with $y' W y = 1$. Then our implicit normalization problem is equivalent to

$$\min_{\alpha \geq 0} \min_{\beta \geq 0} \min_{x \in X \cap \mathcal{S}_x} \min_{y \in Y \cap \mathcal{S}_y} \frac{(\alpha x - \beta y)' W (\alpha x - \beta y)}{\alpha^2 x' W x} = \min_{x \in X \cap \mathcal{S}_x} \min_{y \in Y \cap \mathcal{S}_y} \min_{\alpha \geq 0} \min_{\beta \geq 0} \frac{\alpha^2 + \beta^2 - 2\alpha\beta x' W y}{\alpha^2} =$$

Thus implicit normalization of x means maximizing $(x' W y)^2$ over $x \in X \cap \mathcal{S}_x$ and $y \in Y \cap \mathcal{S}_y$.

In the same way implicit normalization of y minimizes

$$\min_{x \in X} \min_{y \in Y} \frac{(x - y)' W (x - y)}{y' W y},$$

and in the same way it also leads to maximization of $(x' W y)^2$ over $x \in X \cap \mathcal{S}_x$ and $y \in Y \cap \mathcal{S}_y$. In terms of normalized stress it does not matter if we use the distances or the dissimilarities in the denominator for implicit normalization.

In explicit normalization of x we solve

$$\min_{x \in X \cap \mathcal{S}_x} \min_{y \in Y} \{1 + y' W y - 2y' W x\} = \min_{\beta \geq 0} \min_{x \in X \cap \mathcal{S}_x} \min_{y \in Y \cap \mathcal{S}_y} \{1 + \beta^2 - 2\beta x' W y\} = \min_{x \in X \cap \mathcal{S}_x} \min_{y \in Y \cap \mathcal{S}_y} \{1 - (x' W y)^2\}$$

and the same thing is true for explicit normalization of y , which is

$$\min_{x \in X} \min_{y \in Y \cap S_y} \{1 + x'Wx - 2y'Wx\}$$

So, again, it does not matter which one of the four normalizations we use, explicit/implicit on disparities/distances, the solutions will all be proportional, i.e. the same except for scale factors.

8.4.2 Inner Iterations

8.4.3 Stress-1 and Stress-2

In his original papers Kruskal (1964a) and Kruskal (1964b) defined two versions of normalized stress for nonmetric MDS. The first was

$$\begin{aligned}\sigma_{JBK1}(X) &:= \sqrt{\frac{\sum \sum_{1 \leq i < j \leq n} (\hat{d}_{ij} - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} d_{ij}^2(X)}} \\ \sigma_{JBK2}(X) &:= \sqrt{\frac{\sum \sum_{1 \leq i < j \leq n} (\hat{d}_{ij} - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} (d_{ij}(X) - \bar{d}(X))^2}}\end{aligned}$$

where the \hat{d}_{ij} (the d-hats) are the pseudo-distances obtained by projecting the $d_{ij}(X)$ on the isocone defined by the order of the dissimilarities, i.e. by monotone regression (see section 11.1.1). The $\bar{d}(X)$ in the denominator of σ_{JBK2} is the average of the distances.

There are some differences with the definition of stress in this book.

1. We do not use the square root.
2. We use explicit and not implicit normalization.
3. In NMDS we think of stress as a function of both X and Δ , not of X only (see section 11.3).

Chapter 9

Interval MDS

intro: additive vs interval basic vs ratio

9.1 The Additive Constant

In the early history of MDS dissimilarities were computed from comparative judgments in the Thurstonian tradition.

triads paired comparisons etc positive orthant

These early techniques only gave numbers on an interval scale, i.e. dissimilarities known only up to a linear transformation. In order to get positive dissimilarities a rational origin needed to be found in some way. This is the *additive constant problem*. It can be seen as the first example of nonmetric MDS, in which we have only partially known dissimilarities.

$$\begin{aligned}(\delta_{ij} + \alpha) &\approx d_{ij}(X), \\ \delta_{ij} &\approx d_{ij}(X) + \alpha.\end{aligned}\tag{9.1}$$

The additive constant techniques were more important in the fifties and sixties than they are these days, because they have largely been replaced by iterative nonmetric MDS techniques.

An early algorithm to fit the additive constant based on Schoenberg's theorem was given by Messick and Abelson (1956). Torgerson based, i.e. the

eigenvalues of $\tau(\Delta^{(2)})$. It was a somewhat hopeful iterative technique, without a convergence proof, designed to make the sum of the $n - p$ smallest eigenvalues equal to zero. This is of course only a necessary condition for best approximation, not a sufficient one.

In addition, the Messick-Abelson algorithm sometimes yielded solutions in which the Torgerson transform of the squared dissimilarities had negative eigenvalues, which could even be quite large. That is also somewhat of a problem. Consequently Cooper (1972) proposed an alternative additive constant algorithm, taking his clue from the work of Kruskal.

The solution was to redefine stress as a function of both the configuration and the additive constant. Thus

$$\sigma(X, \alpha) := \sum_{1 \leq j < i \leq n} w_{ij} (\delta_{ij} + \alpha - d_{ij}(X))^2, \quad (9.2)$$

and we minimize this stress over both X and α .

Double phase (ALS)

$$\delta_{ij} + \alpha \geq 0$$

Single Phase

$$\sigma(X) := \min_{\alpha} \sum_{1 \leq j < i \leq n} w_{ij} (\delta_{ij} + \alpha - d_{ij}(X))^2, \quad (9.3)$$

9.1.1 Algebra

The additive constant problem is to find $X \in \mathbb{R}^{n \times p}$ and α such that $\Delta + \alpha(E - I) \approx D(X)$. In this section we look for all α such that $\Delta + \alpha(E - I)$ is Euclidean, i.e. such that there is a configuration X with $\Delta + \alpha(E - I) = D(X)$. This is a one-parameter generalization of Schoenberg's theorem.

It makes sense to require $\alpha \geq 0$, because a negative α would more appropriately be called a subtractive constant. Also, we may want to make sure that the off-diagonal elements of $\Delta + \alpha(E - I)$ are non-negative, i.e. that $\alpha \geq -\delta_{ij}$ for all $i > j$. Note that if we allow a negative α then if all off-diagonal δ_{ij} are equal, say to $\delta > 0$, we have the trivial solution $\alpha = -\delta$ and $X = 0$.

We start with a simple construction.

Theorem 9.1. *For all Δ there is an $\alpha_0 \geq 0$ such that for all $\alpha \geq \alpha_0$ we have $\Delta + \alpha(E - I)$ Euclidean of dimension $r \leq n - 1$.*

Proof. We have, using $\Delta \times (E - I) = \Delta$ and $(E - I) \times (E - I) = E - I$,

$$\tau((\Delta + \alpha(E - I)) \times (\Delta + \alpha(E - I))) = \tau(\Delta \times \Delta) + 2\alpha\tau(\Delta) + \frac{1}{2}\alpha^2 J. \quad (9.4)$$

Thus each off-diagonal element is a concave quadratic in α , which is negative for α big enough. Choose $\alpha_0 \geq 0$ to make all off-diagonal elements negative (and all dissimilarities non-negative). A doubly-centered matrix with all off-diagonal elements negative is positive semi-definite of rank $n - 1$ (Taussky (1949)). \square

Note that by the same argument we can also find a negative α_0 that makes all off-diagonal elements negative and thus $\Delta + \alpha(E - I)$ is again Euclidean of dimension $r \leq n - 1$. But this α_0 will usually result in negative dissimilarities.

Theorem 9.1 can be sharpened for non-Euclidean Δ . Define the following function of α :

$$\lambda_*(\alpha) := \min_{x'x=1, x'e=0} x' \{\tau(\Delta \times \Delta) + 2\alpha\tau(\Delta) + \frac{1}{2}\alpha^2 J\} x. \quad (9.5)$$

This is the smallest non-trivial eigenvalue of the Torgerson transform in (9.4). The matrix $\Delta + \alpha(E - I)$ is Euclidean if and only if $\lambda_*(\alpha) \geq 0$. Note that λ_* is continuous, by a simple special case of the Maximum Theorem (Berge (1963), Chapter VI, section 3), and coercive, i.e. $\lambda_*(\alpha) \rightarrow +\infty$ if $|\alpha| \rightarrow +\infty$.

Theorem 9.2. *For all non-Euclidean Δ there is an $\alpha_1 > 0$ such that for all $\alpha \geq \alpha_1$ we have that $\Delta + \alpha(E - I)$ Euclidean of dimension $r \leq n - 2$.*

Proof. Because Δ is non-Euclidean we have $\lambda_*(0) < 0$. By the construction in theorem 9.1 there is an α_0 such that $\lambda_*(\alpha) > 0$ for all $\alpha > \alpha_0$. By the Maximum Theorem the function λ_* is continuous, and thus, by Bolzano's theorem, there is an α_1 between 0 and α_0 such that $\lambda_*(\alpha_1) = 0$. If there is more than one zero between 0 and α_0 we take the largest one as α_1 . \square

The problem with extending theorem 9.2 to Euclidean Δ is that the equation $\lambda_*(\alpha) = 0$ may have only negative roots, or, even more seriously, no roots at all. This may not be too important from the practical point of view, because observed dissimilarities will usually not be exactly Euclidean. Nevertheless I feel compelled to address it.

Theorem 9.3. *If Δ is Euclidean then $\lambda_*(\alpha)$ is non-negative and non-decreasing on $[0, +\infty)$.*

Proof. If Δ is Euclidean, then $\sqrt{\Delta}$, which is short for the matrix with the square roots of the dissimilarities, is Euclidean as well. This follows because the square root is a Schoenberg transform (Schoenberg (1937), Bavaud (2011)), and it implies that $\tau(\Delta) = \tau(\sqrt{\Delta} \times \sqrt{\Delta})$ is positive semi-definite. Thus the matrix (9.4) is positive semi-definite for all $\alpha \geq 0$. By Danskin's Theorem the one-sided directional derivative of λ_* at α is $2x(\alpha)' \tau(\Delta) x(\alpha) + \alpha$, where $x(\alpha)$ is one of the minimizing eigenvectors. Because the one-sided derivative is non-negative, the function is non-decreasing (in fact increasing if $\alpha > 0$). \square

Of course $\lambda_*(\alpha) = 0$ can still have negative solutions, and in particular it will have at least one negative solution if $\lambda_*(\alpha) \leq 0$ for any α . There can even be negative solutions with $\Delta + \alpha(E - I)$ non-negative.

The solutions of $\lambda_*(\alpha) = 0$ can be computed and studied in more detail, using results first presented in the psychometric literature by Cailliez (1983). We reproduce his analysis here, with a somewhat different discussion that relies more on existing mathematical results.

In order to find the smallest α we solve the quadratic eigenvalue problem (Tisseur and Meerbergen (2001))

$$\{\tau(\Delta \times \Delta) + 2\alpha\tau(\Delta) + \frac{1}{2}\alpha^2 J\}y = 0. \quad (9.6)$$

A solution (y, α) of #ref(eq:qep1) is an eigen pair, in which y is an eigenvector, and α the corresponding eigenvalue. The trivial solution $y = e$ satisfies #ref(eq:qep1) for any α . We are not really interested in the non-trivial eigenvectors here, but we will look at the relationship between the eigenvalues and the solutions of $\lambda_*(\alpha) = 0$.

The eigenvalues can be complex, in which case they do not interest us. If α is a non-trivial real eigenvalue, then the rank of the Torgerson transform of the matrix in #ref(eq:qep1) is $n - 2$, but

To get rid of the annoying trivial solution $y = e$ we use a square orthonormal matrix whose first column is proportional to e . Suppose L contains the remaining $n - 1$ columns. Now solve

$$\{L'\tau(\Delta \times \Delta)L + 2\alpha L'\tau(\Delta)L + \frac{1}{2}\alpha^2 I\}y = 0. \quad (9.7)$$

Note that the determinant of the polynomial matrix in (9.7) is a polynomial of degree $2(n - 1)$ in α , which has $2(n - 1)$ real or complex roots.

The next step is linearization (Gohberg, Lancaster, and Rodman (2009), chapter 1), which means finding a linear or generalized linear eigen problem with the same roots as (9.7). In our case this is the eigenvalue problem for the matrix

$$\begin{bmatrix} 0 & I \\ -2L'\tau(\Delta \times \Delta)L & -4L'\tau(\Delta)L \end{bmatrix} \quad (9.8)$$

9.1.2 Examples

9.1.2.1 Small Example

Here is a small artificial dissimilarity matrix.

```
##   1  2  3  4
## 1 +0 +1 +2 +5
## 2 +1 +0 +4 +2
## 3 +2 +4 +0 +1
## 4 +5 +2 +1 +0
```

It is constructed such that $\delta_{14} > \delta_{12} + \delta_{24}$ and that $\delta_{23} > \delta_{21} + \delta_{13}$. Because the triangle inequality is violated the dissimilarities are not distances in any metric space, and certainly not in a Euclidean one. Because the minimum dissimilarity is $+1$, we require that the additive constant α is at least -1 .

The R function treq() in appendix A.1.9 finds the smallest additive constant such that all triangle inequalities are satisfied. For this example it is $\alpha = 2$.

The Torgerson transform of $\Delta \times \Delta$ is

```
##   1      2      3      4
## 1 +4.312 +2.688 +1.188 -8.188
## 2 +2.688 +2.062 -5.938 +1.188
## 3 +1.188 -5.938 +2.062 +2.688
## 4 -8.188 +1.188 +2.688 +4.312
```

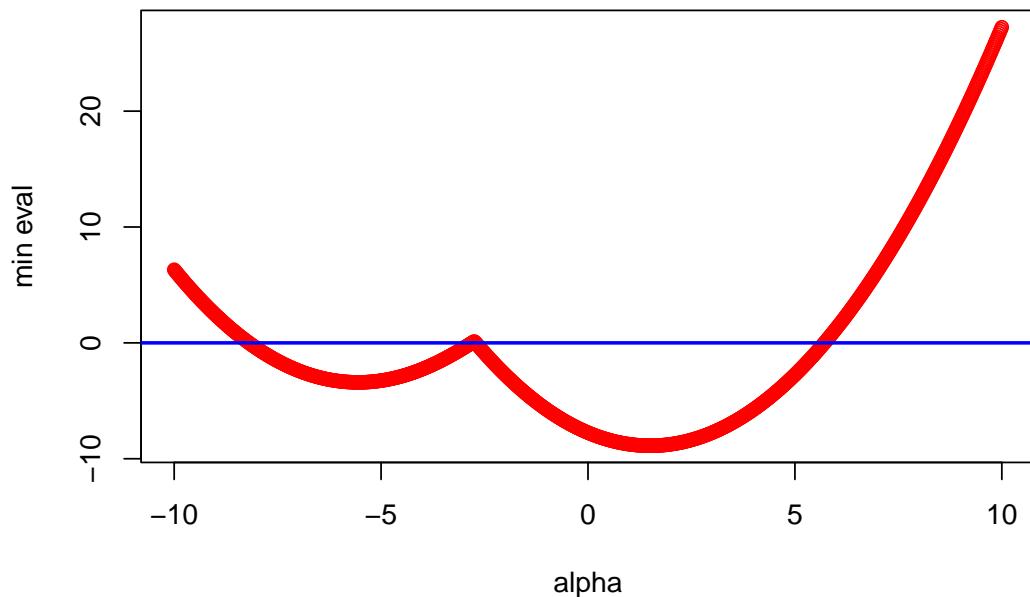
with eigenvalues

```
## [1] +12.954 +7.546 +0.000 -7.750
```

The smallest eigenvalue -7.75 is appropriately negative, and theorem 9.2 shows that $\Delta \times \Delta + 7.75(E - I)$ are squared distances between four points in the plane.

The upper bound for the smallest α from theorem 9.1, computed by the R function acbound(), is 9.309475.

It is useful to look at a graphical representation of the minimum non-trivial eigenvalue of $\tau((\Delta + \alpha(E - I)) \times (\Delta + \alpha(E - I)))$ as a function of α . The R function aceval() generates the data for the plot.



We see that the minimum non-trivial eigenvalue is a continuous function of α , but one which certainly is not convex or concave or differentiable. The graph crosses the horizontal axes near -8, -3, and +6.

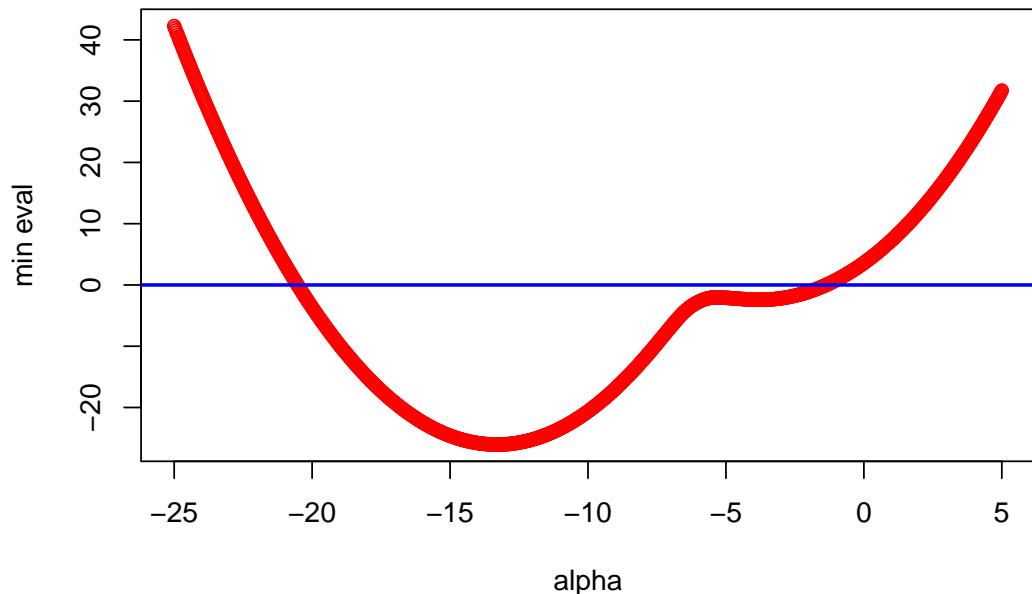
To make this precise we apply the theory of section xxx. The R function acqep() finds the six non-trivial eigenvalues

```
## [1] -8.192582+0.000000i  5.713075+0.000000i -3.500000+2.179449i
## [4] -3.500000-2.179449i -2.807418+0.000000i -2.713075+0.000000i
```

Two of the eigenvalues are complex conjugates, four are real. Of the real eigenvalues three are negative, and only one is positive, equal to +5.713. The table above gives the eigenvalues of the Torgerson transform, using all four real eigenvalues for α . The three negative ones do result in a positive semi-definite matrix with rank equal to $n - 2$, but they also create negative dissimilarities.

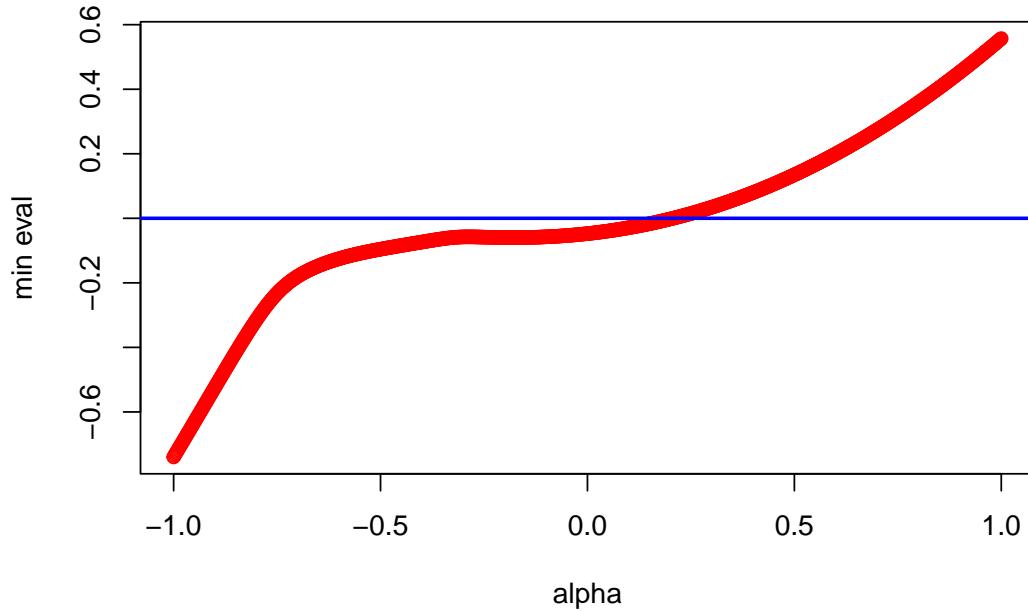
```
## -8.193 ***** +38.098 +13.885 +0.000 -0.000
## +5.713 ***** +61.116 +43.441 +0.000 -0.000
## -2.807 ***** +3.115 +0.402 -0.000 -0.000
## -2.713 ***** +3.228 +0.215 +0.000 +0.000
```

9.1.2.2 De Gruijter Example

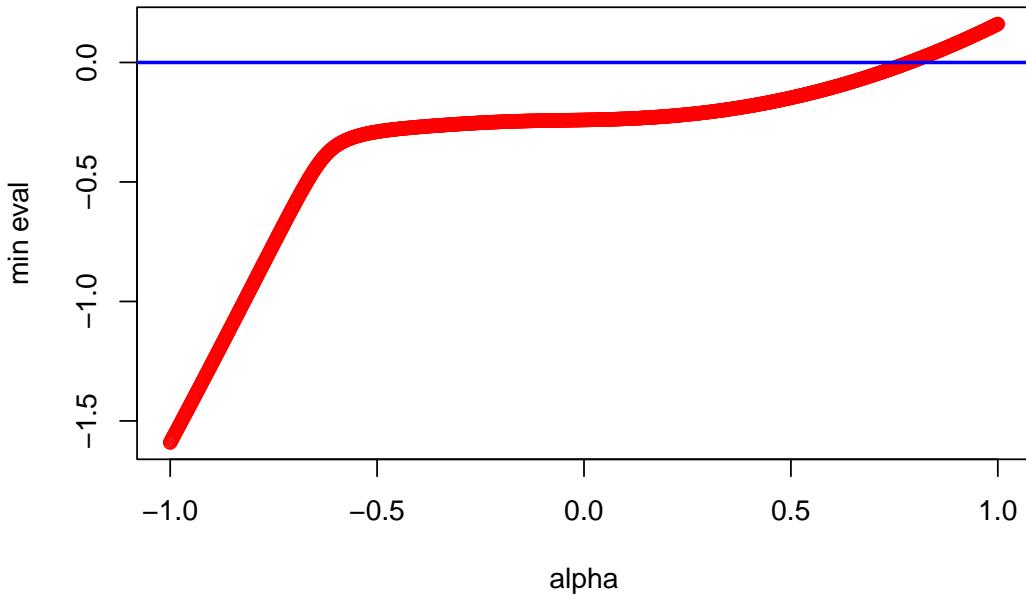


```
## [1] -20.527411+0.000000i -10.174103+0.000000i -9.472504+0.000000i
## [4] -6.622263+0.352619i -6.622263-0.352619i -5.885691+0.287544i
## [7] -5.885691-0.287544i -5.640580+0.366889i -5.640580-0.366889i
## [10] -4.391289+0.253248i -4.391289-0.253248i -3.708911+0.386844i
## [13] -3.708911-0.386844i -3.238930+0.000000i -2.311379+0.000000i
## [16] -1.369315+0.000000i
```

9.1.2.3 Ekman Example



```
## [1] -5.713009655+0.0000000i -3.782729083+0.0000000i -1.791313475+0.0000000i
## [4] -1.628964140+0.0000000i -0.976213035+0.0000000i -0.744289350+0.0495939i
## [7] -0.744289350-0.0495939i -0.682321433+0.0000000i -0.534849034+0.0000000i
## [10] -0.513033529+0.0000000i -0.497908376+0.0248145i -0.497908376-0.0248145i
## [13] -0.372321687+0.1313892i -0.372321687-0.1313892i -0.388308013+0.0000000i
## [16] -0.229813135+0.1825985i -0.229813135-0.1825985i -0.286712033+0.0000000i
## [19] -0.212601059+0.1185199i -0.212601059-0.1185199i 0.206312577+0.0000000i
## [22] -0.194299448+0.0000000i 0.132767430+0.0000000i -0.079646956+0.0000000i
## [25] -0.024193535+0.0000000i -0.006762279+0.0000000i
```



```

## [1] -7.9740652+0.0000000i -4.8679294+0.0000000i -1.2242342+0.0000000i
## [4] -0.9822376+0.0000000i  0.7856448+0.0000000i  0.6486775+0.0000000i
## [7] -0.5540332+0.0000000i -0.5426006+0.0129703i -0.5426006-0.0129703i
## [10] 0.4864182+0.0000000i -0.1118921+0.3923586i -0.1118921-0.3923586i
## [13] 0.3829746+0.0000000i  0.3530897+0.0000000i -0.3516103+0.0000000i
## [16] -0.3073607+0.0000000i -0.1265941+0.2630789i -0.1265941-0.2630789i
## [19] -0.0735448+0.2698631i -0.0735448-0.2698631i -0.2333027+0.0000000i
## [22] -0.0081376+0.1948006i -0.0081376-0.1948006i -0.1380254+0.1175071i
## [25] -0.1380254-0.1175071i  0.1205026+0.0000000i

```

9.1.3 A Variation

Alternatively, we could define our approximation problem as finding $X \in \mathbb{R}^{n \times p}$ and α such that $\sqrt{\delta_{ij}^2 + \alpha} \approx d_{ij}(X)$, or, equivalently, $\Delta \times \Delta + \alpha(E - I) \approx D(X) \times D(X)$.

Theorem 9.4. *For any $X \in \mathbb{R}^{n \times p}$ with $p = n - 2$ there is an α such that $\sqrt{\delta_{ij}^2 + \alpha} = d_{ij}(X)$.*

Proof. Now we have

$$\tau(\Delta \times \Delta + \alpha(E - I)) = \tau(\Delta \times \Delta) + \frac{1}{2}\alpha J. \quad (9.9)$$

The eigenvalues of $\tau(\Delta \times \Delta) + \frac{1}{2}\alpha J$ are zero and $\lambda_s + \frac{1}{2}\alpha$, where the λ_s are the $n - 1$ non-trivial eigenvalues of $\tau(\Delta \times \Delta)$. If $\underline{\lambda}$ is smallest eigenvalue we choose $\alpha = -2\underline{\lambda}$, and $\tau(\Delta \times \Delta) + \frac{1}{2}\alpha J$ is positive semi-definite of rank $r \leq n - 2$. \square

Note that theorem 9.2 implies that for any Δ there is a strictly increasing differentiable transformation to the space of Euclidean distance matrices in $n - 2$ dimensions. This is a version of what is sometimes described as *Guttman's n-2 theorem* (Lingoes (1971)). The proof we have given is that from De Leeuw (1970), Appendix B.

ALS Negative delta

Euclidean

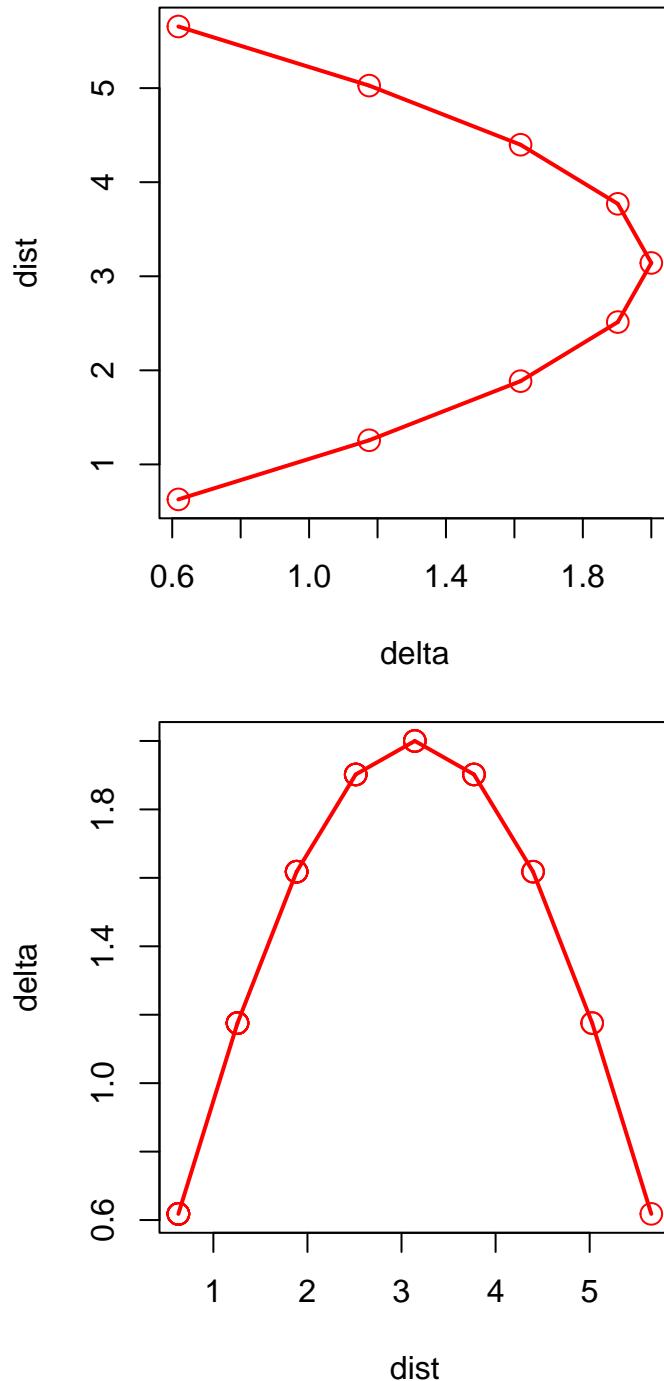
$$\sqrt{2 - 2 \cos |i - j| \theta}$$

Circular

$$|i - j| \frac{2\pi}{n}$$

Linear

$$|i - j|$$



Chapter 10

Polynomial and Splinical MDS

10.1 Polynomial MDS

$$\sigma(X) = \sum_{1 \leq i < j \leq n} w_{ij} (P_r(\delta_{ij}) - d_{ij}(X))^2$$

$$P_r(\delta_{ij}) := \sum_{s=0}^r \alpha_s \delta_{ij}^s.$$

The polynomial P_r is *tied down* if $\alpha_0 = 0$, and thus $P_r(0) = 0$.

Vandermonde matrix

10.1.1 Monotone Polynomials

10.2 Splinical MDS

10.3 Splines

In this section we give a short introduction, with examples, to (univariate) splines, B-splines, and I-splines. It is taken from De Leeuw (2017a), with some edits to make it fit into the book. The report it was taken from has more detail and more examples.

To define *spline functions* we first define a finite sequence of *knots* $T = \{t_j\}$ on the real line, with $t_1 \leq \dots \leq t_p$, and an *order* m . In addition each knot t_j has a *multiplicity* m_j , the number of knots equal to t_j . We suppose throughout that $m_j \leq m$ for all j .

A function f is a *spline function of order m* for a knot sequence $\{t_j\}$ if

1. f is a polynomial π_j of degree at most $m - 1$ on each half-open interval $I_j = [t_j, t_{j+1})$ for $j = 1, \dots, p$,
2. the polynomial pieces are joined in such a way that $\mathcal{D}_-^{(s)} f(t_j) = \mathcal{D}_+^{(s)} f(t_j)$ for $s = 0, 1, \dots, m - m_j - 1$ and $j = 1, 2, \dots, p$.

Here we use $\mathcal{D}_-^{(s)}$ and $\mathcal{D}_+^{(s)}$ for the left and right s^{th} -derivative operator. If $m_j = m$ for some j , then the second requirement is empty, if $m_j = m - 1$ then the second requirement means $\pi_j(t_j) = \pi_{j+1}(t_j)$, i.e. we require continuity of f at t_j . If $1 \leq m_j < m - 1$ then f must be $m - m_j - 1$ times differentiable, and thus continuously differentiable, at t_j .

In the case of simple knots (with multiplicity one) a spline function of order one is a *step function* which steps from one level to the next at each knot. A spline of order two is piecewise linear, with the pieces joined at the knots so that the spline function is continuous. Order three means a piecewise quadratic function which is continuously differentiable at the knots. And so on.

10.3.1 B-splines

Alternatively, a spline function of order m can be defined as a linear combination of *B-splines* (or *basic splines*) of order m on the same knot sequence. A B-spline of order m is a spline function consisting of at most m non-zero polynomial pieces. A B-spline $\mathcal{B}_{j,m}$ is determined by the $m + 1$ knots $t_j \leq \dots \leq t_{j+m}$, is zero outside the interval $[t_j, t_{j+m})$, and positive in the interior of that interval. Thus if $t_j = t_{j+m}$ then $\mathcal{B}_{j,m}$ is identically zero.

For an arbitrary finite knot sequence t_1, \dots, t_p , there are $p - m$ B-splines to of order m to be considered, although some may be identically zero. Each of the splines covers at most m consecutive intervals, and at most $m - 1$ different B-splines are non-zero at each point.

10.3.1.1 Boundaries

B-splines are most naturally and simply defined for doubly infinite sequences of knots, that go to $\pm\infty$ in both directions. In that case we do not have to worry about boundary effects, and each subsequence of $m + 1$ knots defines a B-spline. For splines on finite sequences of p knots we have to decide what happens at the boundary points.

There are B-splines for t_j, \dots, t_{j+m} for all $j = 1, \dots, p - m$. This means that the first $m - 1$ and the last $m - 1$ intervals have fewer than m splines defined on them. They are not part of what De Boor (2001), page 94, calls the *basic interval*. For doubly infinite sequences of knots there is not need to consider such a basic interval.

If we had m additional knots on both sides of our knot sequence we would also have m additional B-splines for $j = 1 - m, \dots, 0$ and m additional B-splines for $j = p - m + 1, \dots, p$. By adding these additional knots we make sure each interval $[t_j, t_{j+1})$ for $j = 1, \dots, p - 1$ has m B-splines associated with it. There is still some ambiguity on what to do at t_p , but we can decide to set the value of the spline there equal to the limit from the left, thus making the B-spline left-continuous there.

In our software we will use the convention to define our splines on a closed interval $[a, b]$ with r *interior knots* $a < t_1 < \dots < t_r < b$, where interior knot t_j has multiplicity m_j . We extend this to a series of $p = M + 2m$ knots, with $M = \sum_{j=1}^r m_j$, by starting with m copies of a , appending m_j copies of t_j for each $j = 1, \dots, r$, and finishing with m copies of b . Thus a and b are both knots with multiplicity m . This defines the *extended partition* (Schumaker (2007), p 116), which is just handled as any knot sequence would normally be.

10.3.1.2 Normalization

The conditions we have mentioned only determine the B-spline up to a normalization. There are two popular ways of normalizing B-splines. The N -splines $N_{j,m}$, a.k.a. the *normalized B-splines* j or order m , satisfies

$$\sum_j N_{j,m}(t) = 1. \tag{10.1}$$

Note that in general this is not true for all t , but only for all t in the *basic interval*.

Alternatively we can normalize to M -splines, for which

$$\int_{-\infty}^{+\infty} M_{j,m}(t) dt = \int_{t_j}^{t_{j+k}} M_{j,m}(t) dt = 1. \quad (10.2)$$

There is the simple relationship

$$N_{j,m}(t) = \frac{t_{j+m} - t_j}{m} M_{j,m}(t). \quad (10.3)$$

10.3.1.3 Recursion

B-splines can be defined in various ways, using piecewise polynomials, divided differences, or recursion. The recursive definition, first used as actually defining B-splines by De Boor and Höllig (1985), is the most convenient one for computational purposes, and that is the one we use.

The recursion is due independently to M. G. Cox (1972) for simple knots and to De Boor (1972) in the general case, is

$$M_{j,m}(t) = \frac{t - t_j}{t_{j+m} - t_j} M_{j,m-1}(t) + \frac{t_{j+m} - t}{t_{j+m} - t_j} M_{j+1,m-1}(t), \quad (10.4)$$

or

$$N_{j,m}(t) = \frac{t - t_j}{t_{m+j-1} - t_j} N_{j,m-1}(t) + \frac{t_{j+m} - t}{t_{j+m} - t_{j+1}} N_{j+1,m-1}(t). \quad (10.5)$$

A basic result in the theory of B-splines is that the different B-splines are linearly independent and form a basis for the linear space of spline functions (of a given order and knot sequence).

In section A.1.18 the basic BSPLVB algorithm from De Boor (2001), page 111, for normalized B-splines is translated to R and C. There are two auxiliary routines, one to create the extended partition, and one that uses bisection to locate the knot interval in which a particular value is located (Schumaker (2007), p 191). The R function `bsplineBasis()` takes an arbitrary knot sequence. It can be combined with `extendPartition()`, which uses inner knots and boundary points to create the extended partition.

10.3.1.4 Illustrations

For our example, which is the same as the one from figure 1 in Ramsay (1988), we choose $a = 0$, $b = 1$, with simple interior knots 0.3, 0.5, 0.6. First the *step functions*, which have order 1.

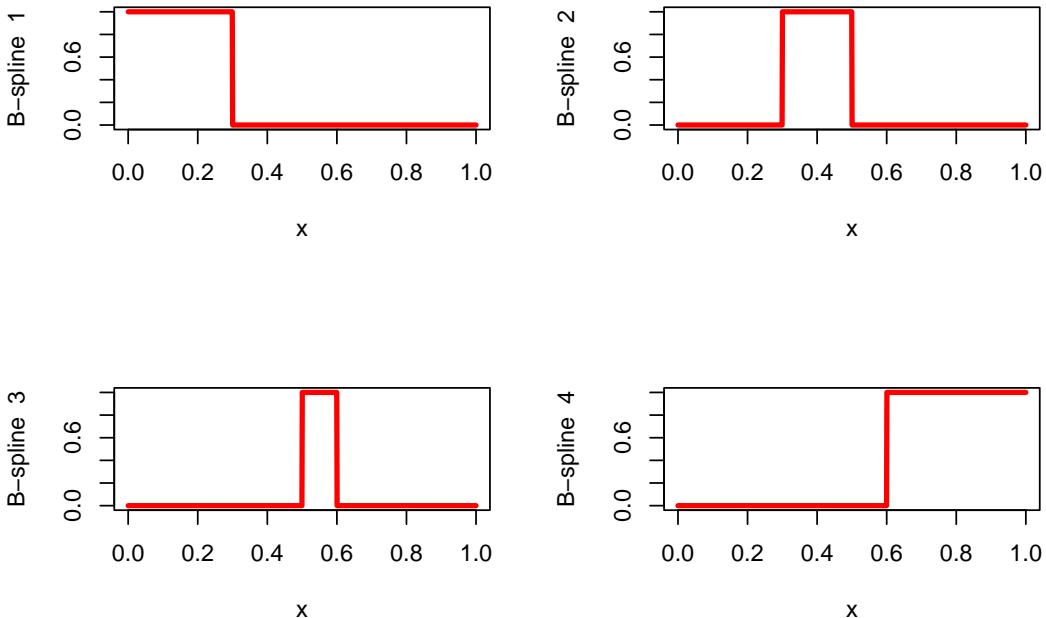


Figure 10.1: Zero Degree Splines with Simple Knots

Now the *hat functions*, which have order 2, again with simple knots.

Next piecewise quadratics, with simple knots, which implies continuous differentiability at the knots. This are the N-splines corresponding with the M-splines in figure 1 of Ramsay (1988).

If we change the multiplicities to 1, 2, 3, then we lose some of the smoothness.

10.3.2 I-splines

There are several ways to require splines to be monotone increasing. Since B-splines are non-negative, the definite integral of a B-spline of order m from the beginning of the interval to a value x in the interval is an increasing spline of order $m+1$. Integrated B-splines are known as *I-splines* (Ramsay (1988)).

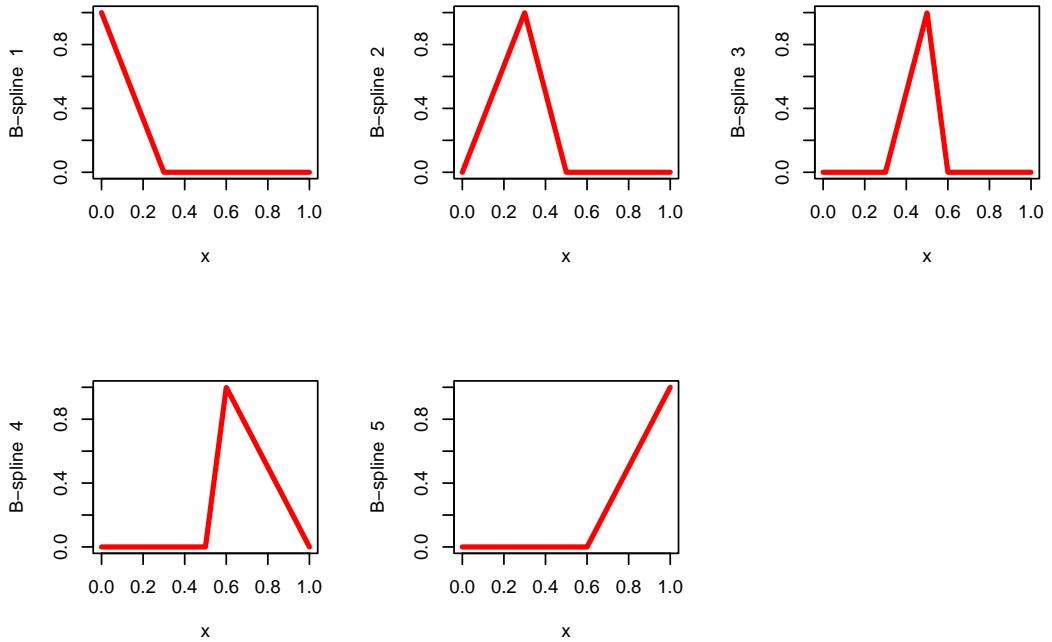


Figure 10.2: Piecewise Linear Splines with Simple Knots

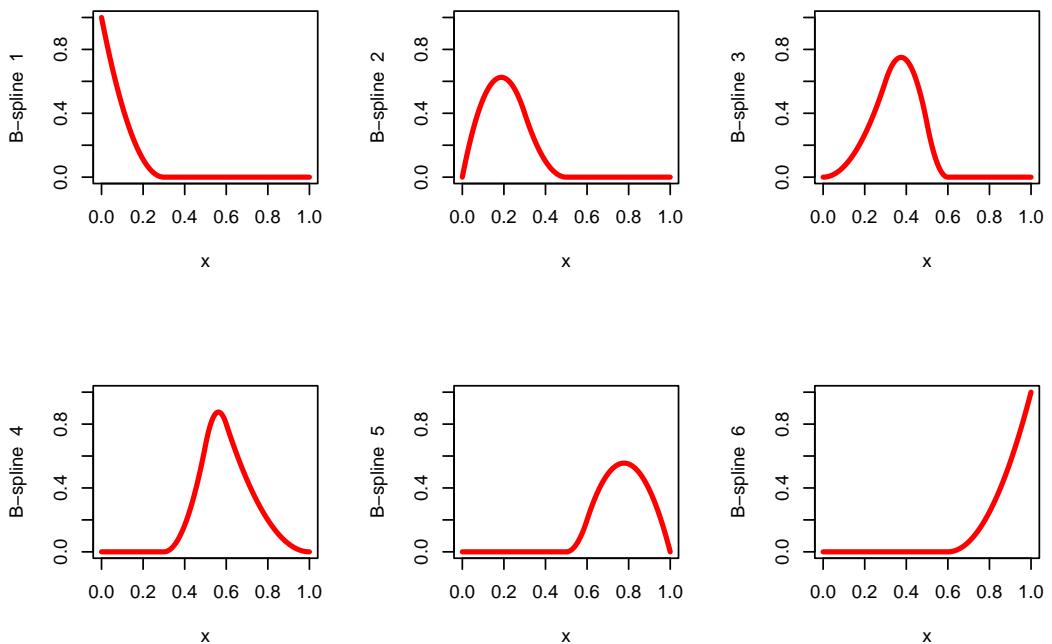


Figure 10.3: Piecewise Quadratic Splines with Simple Knots

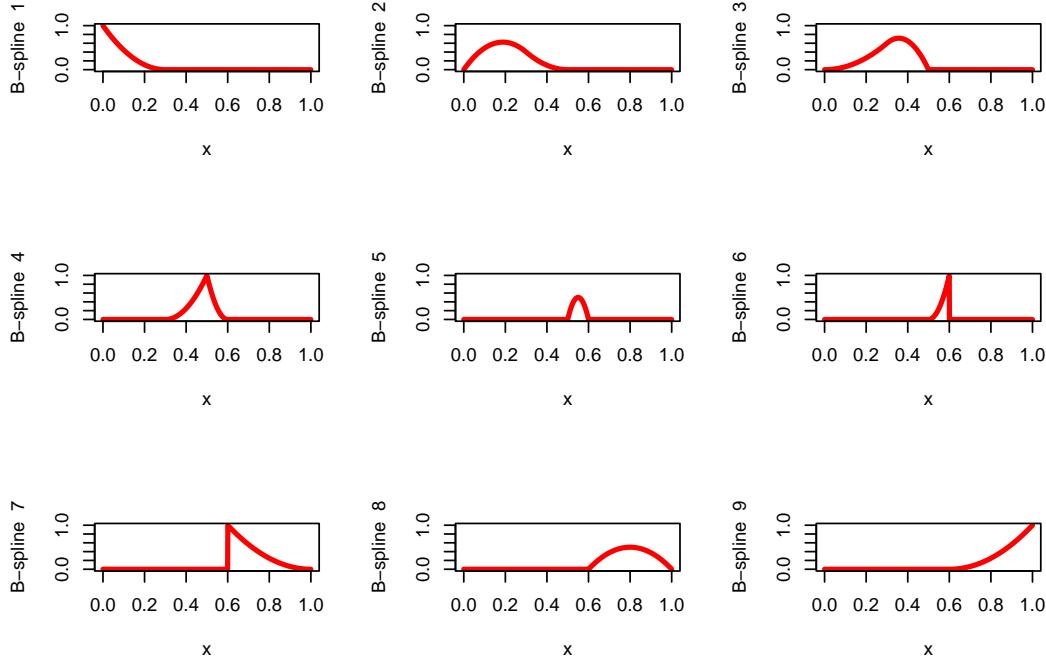


Figure 10.4: Piecewise Quadratic Splines with Multiple Knots

Non-negative linear combinations I-splines can be used as a basis for the convex cone of increasing splines. Note, however, that if we use an extended partition, then all I-splines start at value zero and end at value one, which means their convex combinations are those splines that are also probability distributions on the interval. To get a basis for the increasing splines we need to add the constant function to the I-splines and allow it to enter the linear combination with either sign.

I-splines are most economically computed by using the formula first given by Gaffney (1976). If ℓ is defined by $t_{j+\ell-1} \leq x < t_{j+\ell}$ then

$$\int_{x_j}^x M_{j,m}(t)dt = \frac{1}{m} \sum_{r=0}^{\ell-1} (x - x_{j+r}) M_{j+r, m-r}(x)$$

It is somewhat simpler, however, to use lemma 2.1 of De Boor, Lyche, and Schumaker (1976). This says

$$\int_a^x M_{j,m}(t)dt = \sum_{\ell \geq j} N_{\ell, m+1}(x) - \sum_{\ell \geq j} N_{\ell, m+1}(a),$$

If we specialize this to I-splines, we find , as in De Boor (1976), formula 4.11,

$$\int_{-\infty}^x M_{j,m}(t)dt = \sum_{\ell=j}^{j+r} N_{\ell,m+1}(x)$$

for $x \leq t_{j+r+1}$. This shows that I-splines can be computed by using cumulative sums of B-spline values.

Note that using the definition using integration does not give a natural way to define increasing splines of degree one, i.e. increasing step functions. There is no such problem with the cumulative sum approach.

10.3.2.1 Increasing Coefficients

As we know, a spline is a linear combination of B-splines. The formula for the derivative of a spline, for example in De Boor (2001), p 116, shows that a spline is increasing if the coefficients of the linear combination of B-splines are increasing. Thus we can fit an increasing spline by restricting the coefficients of the linear combination to be increasing, again using the B-spline basis.

It turns out this is in fact identical to using I-splines. If the B-spline values at n points are in an $n \times r$ matrix H , then non-decreasing coefficients β are of the form $\beta = S\alpha + \gamma e_r$, where S is lower-diagonal with all elements on and below the diagonal equal to one, where $\alpha \geq 0$, where e_r has all elements equal to one, and where γ can be of any sign. So $H\beta = (HS)\alpha + \gamma e_n$. Thus non-decreasing coefficients is the same thing as using cumnulative sums of the B-spline basis.

10.3.2.2 Increasing Values

Finally, we can simply require that the n elements of $H\beta$ are increasing. This is a less restrictive requirement, because it allows for the possibility that the spline is decreasing between data values. It has the rather serious disadvantage, however, that it does its computations in n -dimensional space, and not in r -dimensional space, where $r = M + m$, which is usually much smaller than n . Software for the increasing-value restrictions has been written by De Leeuw (2015). In our software, however, we prefer the `cumsum()` approach. It is less general, but considerably more efficient.

We use the same Ramsay example as before, but now cumulatively. First we integrate step functions with simple knots, which have order one, using `isplineBasis()`. The corresponding I-splines are piecewise linear with order two.

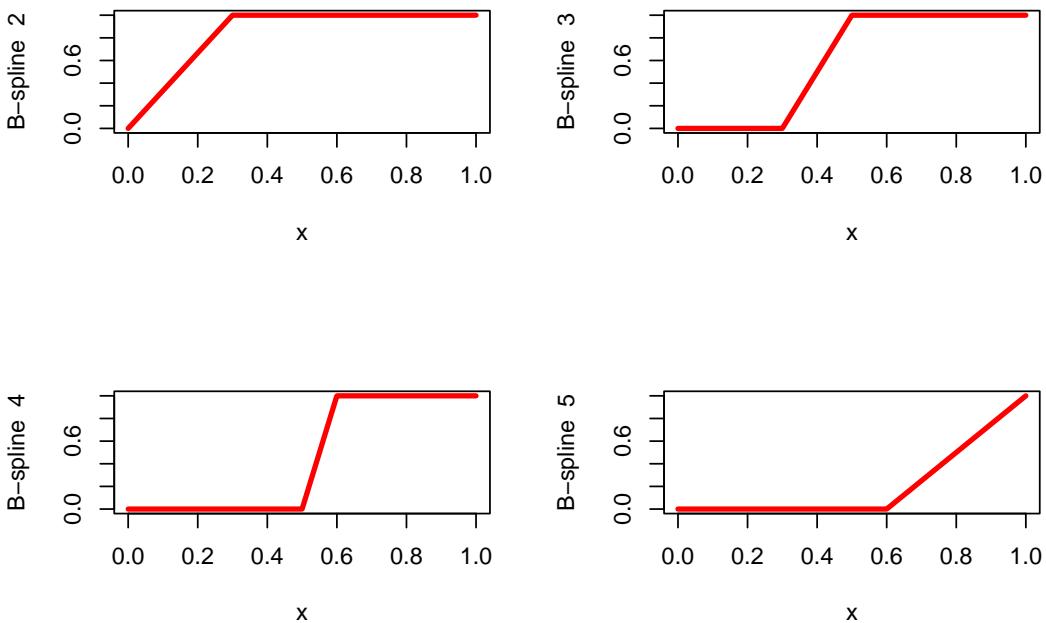


Figure 10.5: Not Sure

Now we integrate the hat functions, which have order 2, again with simple knots, to find piecewise quadratic I-splines of order 3. These are the functions in the example of Ramsay (1988).

Finally, we change the multiplicities to 1, 2, 3, and compute the corresponding piecewise quadratic I-splines.

```
r Iorder3mult2, echo = FALSE, fig.align = "center", fig.cap
= "Monotone Piecewise Quadratic Splines with Multiple Knots"
multiplicities <- c(1, 2, 3) order <- 3 knots <- extendPartition
(innerknots, multiplicities, order, lowend, highend)$knots h <-
isplineBasis(x, knots, order) par(mfrow = c(3, 3)) for (j in
2:ncol(h)) { ylab <- paste("B-spline", formatC(j,
digits = 1, width = 2, format = "d")) plot
(x, h[, j], type = "l", col = "RED", lwd =
3, ylab = ylab ) }
```

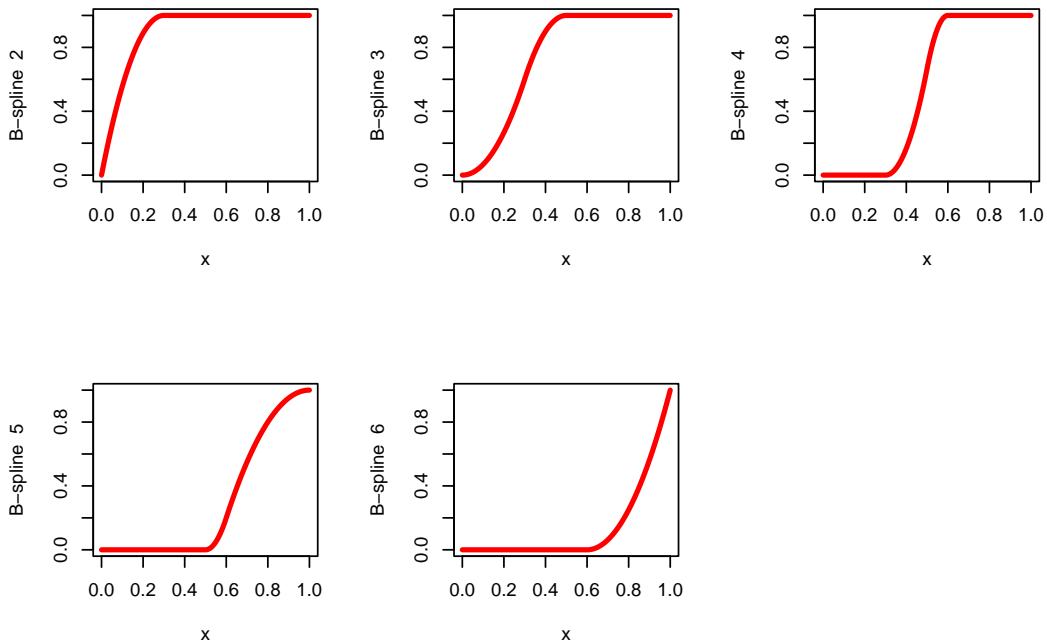


Figure 10.6: Monotone Piecewise Linear Splines with Simple Knots

10.3.3 Time Series Example

Our first example smoothes a time series by fitting a spline. We use the number of births in New York from 1946 to 1959 (on an unknown scale), from Rob Hyndman's time series archive.

```
births <- scan ("http://robjhyndman.com/tsdldata/data/nybirths.dat")
```

10.3.3.1 B-splines

First we fit B-splines of order three. The basis matrix uses x equal to $1 : 168$, with inner knots 12, 24, 36, 48, 60, 72, 84, 96, 108, 120, 132, 144, 156, and interval $[1, 168]$.

```
innerknots <- 12 * 1:13
multiplicities <- rep(1, 13)
lowend <- 1
```

```

highend <- 168
order <- 3
x <- 1:168
knots <-
  extendPartition (innerknots, multiplicities, order, lowend, highend)$knots
h <- bsplineBasis (x, knots, order)
u <- lm.fit(h, births)
res <- sum ((births - h %*% u$coefficients) ^ 2) / 2

```

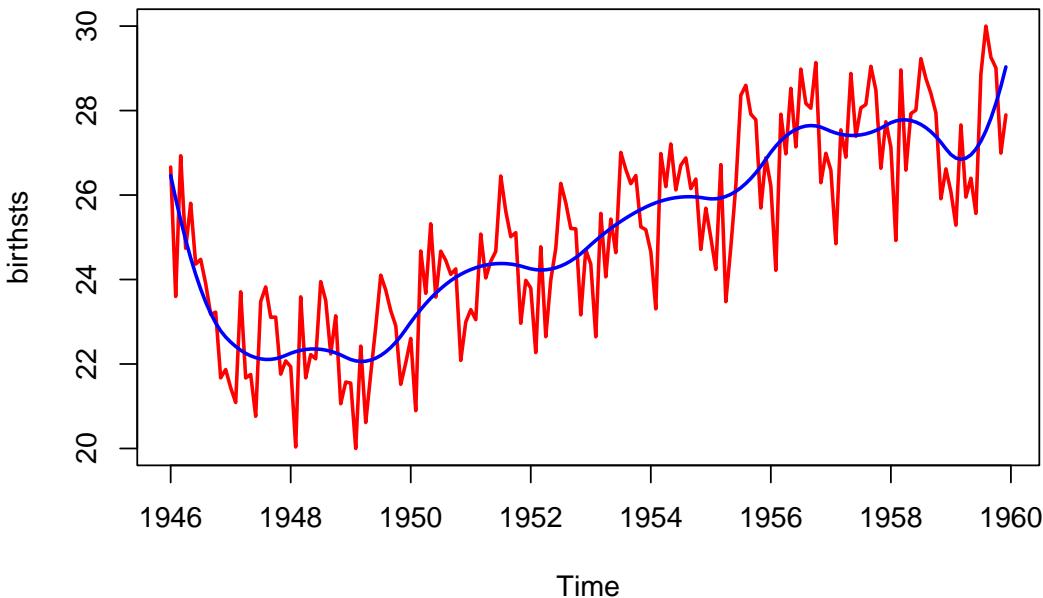


Figure 10.7: Monotone Piecewise Quadratic Splines with Simple Knots

The residual sum of squares is 114.6917709.

10.3.3.2 I-splines

We now fit the I-spline using the B-spline basis. Compute $Z = HS$ using `cumsum()`, and then \bar{y} and \bar{Z} by centering (subtracting the column means). The formula is

$$\min_{\alpha \geq 0, \gamma} \text{SSQ } (y - Z\alpha - \gamma e_n) = \min_{\alpha \geq 0} \text{SSQ } (\bar{y} - \bar{Z}\alpha).$$

We use `pnnls()` from Wang, Lawson, and Hanson (2015).

```

knots <- extendPartition (innerknots, multiplicities, order, lowend, highend)$knots
h <- isplineBasis (x, knots, order)
g <- cbind (1, h[,-1])
u <- pnnls (g, births, 1)$x
v <- g%*%u

```

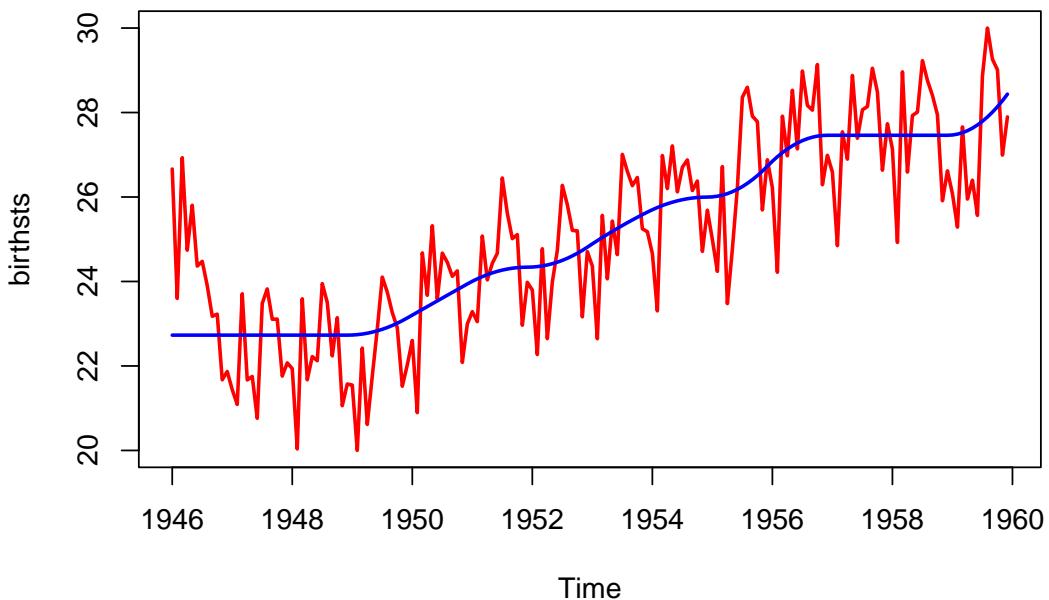


Figure 10.8: Monotone Piecewise Linear Splines with Simple Knots

The residual sum of squares is 144.2027491.

10.3.3.3 B-Splines with monotone weights

Just to make sure, we also solve the problem

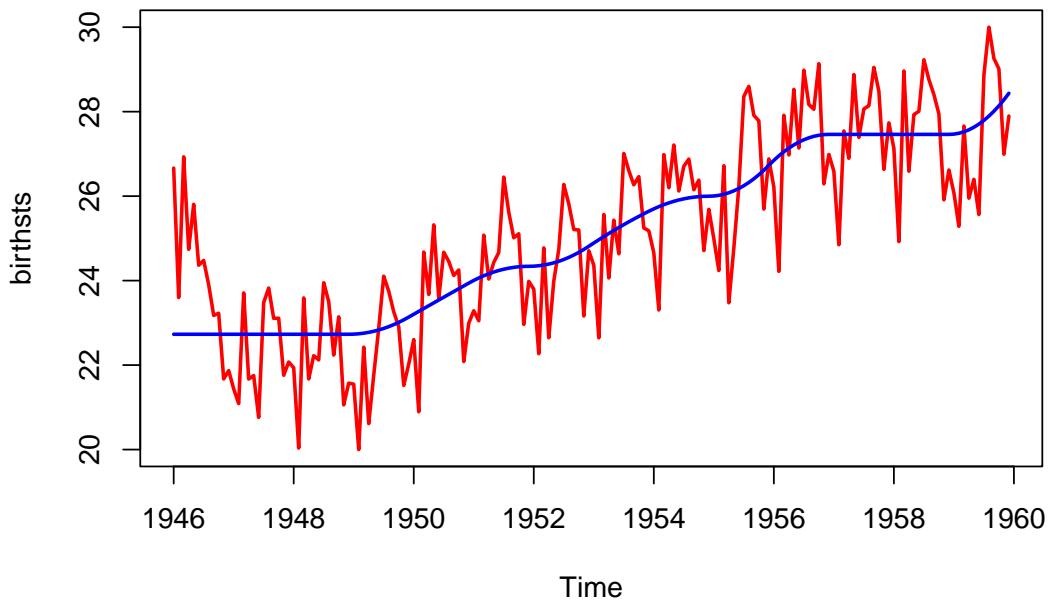
$$\min_{\beta_1 \leq \beta_2 \leq \dots \leq \beta_p} \text{SSQ}(y - X\beta),$$

which should give the same solution, and the same loss function value, because it is just another way to fit I-splines. We use the `lsi()` function from Wang, Lawson, and Hanson (2015).

```

knots <-
  extendPartition (innerknots, multiplicities, order, lowend, highend)$knots
h <- bsplineBasis (x, knots, order)
nb <- ncol (h)
d <- matrix(0, nb - 1, nb)
diag(d) = -1
d[outer(1:(nb - 1), 1:nb, function(i, j)
  (j - i) == 1)] <- 1
u <- lsi(h, births, e = d, f = rep(0, nb - 1))
v <- h %*% u

```



The residual sum of squares is 144.2027491, indeed the same as before.

10.3.3.4 B-Splines with monotone values

Finally we solve

$$\min_{x'_1 \beta \leq \dots \leq x'_n \beta} \text{SSQ} (y - X\beta)$$

using qpmaj() from section 25.7.1.

```

knots <-
  extendPartition (innerknots, multiplicities, order, lowend, highend)$knots
h <- bsplineBasis (x, knots, order)
a <- diff(diag(nrow(h))) %*% h
u <- qpmaj(births, h = h, a = a)

```

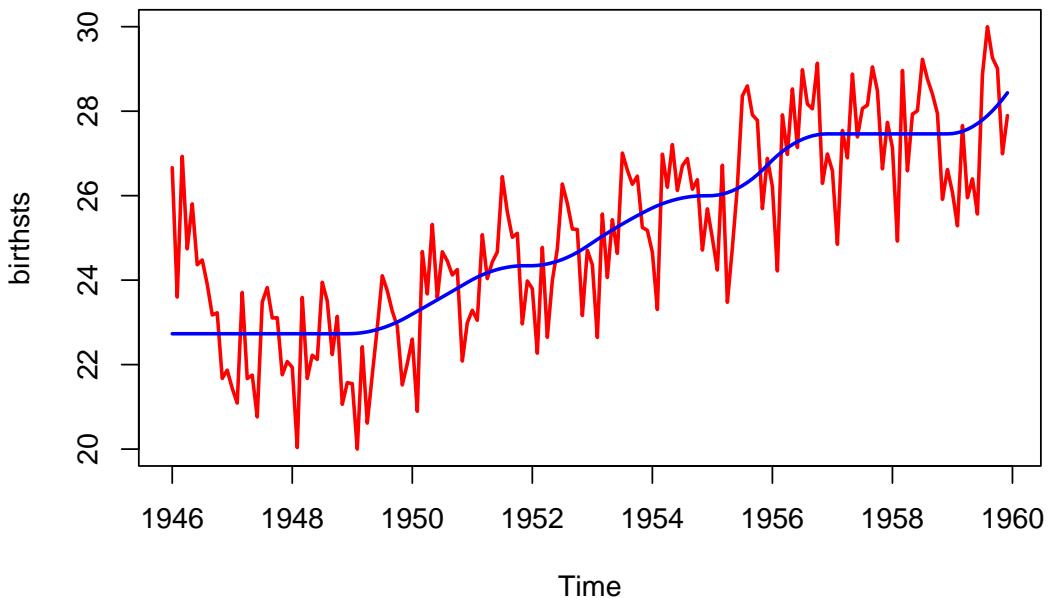


Figure 10.9: Monotone Piecewise Quadratic Splines with Multiple Knots

The residual sum of squares is 144.1574541 , which is indeed smaller than the I-splines value, although only very slightly so.

10.3.4 Monotone Splines

Chapter 11

Ordinal MDS

11.1 Monotone Regression

Is it really what we want

11.1.1 Simple Monotone Regression

Ever since Kruskal (1964a) and Kruskal (1964b) monotone regression has played an important part in non-metric MDS. Too important, perhaps. Initially there was some competition from the rank images of Guttman (1968), but that competition has largely faded over time.

We only give the barest outline in this section. More details are in De Leeuw, Hornik, and Mair (2009). In (simple least squares) monotone (or isotone) regression we minimize $(x - y)'W(x - y)$, where $W \gtrsim 0$ is diagonal, over x satisfying $x_1 \leq \dots \leq x_n$. The vector y is the *target* or the *data*.

The algorithm, which is extremely fast and of order n , is based on the simple rule that if elements are out of order, then you compute their weighted average, forming blocks, keeping track of the block sizes and block weights. A very efficient implementation is in Busing (2021).

A simple illustration. The first column are the value that we compute the best monotone fit for, the second columns are the size of the blocks after

merging. In this case there are no weights, in fact the block sizes serve as weights.

$$(1, 2, 1, 3, 2, -1, 3) \quad (1, 1, 1, 1, 1, 1, 1) \quad (11.1)$$

$$\left(1, \frac{3}{2}, 3, 2, -1, 3\right) \quad (1, 2, 1, 1, 1, 1) \quad (11.2)$$

$$\left(1, \frac{3}{2}, \frac{5}{2}, -1, 3\right) \quad (1, 2, 2, 1, 1) \quad (11.3)$$

$$\left(1, \frac{3}{2}, \frac{4}{3}, 3\right) \quad (1, 2, 3, 1) \quad (11.4)$$

$$\left(1, \frac{7}{5}, 3\right) \quad (1, 5, 1) \quad (11.5)$$

Expanding using the block size gives the solution $(1, \frac{7}{5}, \frac{7}{5}, \frac{7}{5}, \frac{7}{5}, 3)$.

In the second example we do have weights, in the second column, and we use a third column for blocks size.

$$(1, 2, 1, 3, 2, -1, 3) \quad (1, 2, 3, 4, 3, 2, 1) \quad (1, 1, 1, 1, 1, 1, 1) \quad (11.6)$$

$$\left(1, \frac{7}{5}, 3, 2, -1, 3\right) \quad (1, 5, 4, 3, 2, 1) \quad (1, 2, 1, 1, 1, 1) \quad (11.7)$$

$$\left(1, \frac{7}{5}, \frac{18}{7}, -1, 3\right) \quad (1, 5, 7, 2, 1) \quad (1, 2, 2, 1, 1) \quad (11.8)$$

$$\left(1, \frac{7}{5}, \frac{16}{9}, 3\right) \quad (1, 5, 9, 1) \quad (1, 2, 3, 1) \quad (11.9)$$

Expansion gives the solution $(1, \frac{7}{5}, \frac{7}{5}, \frac{16}{9}, \frac{16}{9}, \frac{16}{9}, 3)$.

The usual monotone regression algorithms used in MDS allow for slightly more complicated orders to handle ties in the data. There are basically three approaches to ties implemented. In what Kruskal calls the *primary approach*, only order relations between tie blocks are maintained. Within blocks no order is imposed. In the *secondary approach* we require equality in tie blocks. Ties in the data means we impose ties in the isotone regression. Both approaches can be incorporated in simple monotone regression by pre-processing. The secondary approach starts with the weighted averages of the tie blocks, the primary approach orders the data within tie blocks so they

are non-decreasing. De Leeuw (1977b) showed that this preprocessing does indeed give the least squares solution for both approaches. In the same paper he also introduces a less restrictive *tertiary approach*, which merely requires that the averages of the tie blocks are in the required order.

11.1.2 Weighted Monotone Regression

$$(x - y)'V(x - y)$$

$$V(x - y) = A'\lambda$$

$$Ax \geq 0$$

$$\lambda \geq 0$$

$$\lambda'Ax = 0$$

$$x = y + V^{-1}A'\lambda$$

go to the dual if $\lambda_i > 0$ then $a'_i x = 0$

unweighting actually proves weighted is unweighted for something else

$$MR(x + \epsilon y) = MR(x) + \epsilon B(y) \text{ if } MR(x) = Bx$$

11.1.3 Normalized Cone Regression

De Leeuw (1975a)

Bauschke, Bui, and Wang (2018)

11.2 Alternating Least Squares

smacof: hard squeeze double phase

11.3 Kruskal's Approach

In Guttman's terminology Kruskal's approach is hard squeeze single phase. Thus what is minimized is

$$\sigma_{JBK}(X) := \min_{\Delta \in \mathfrak{D}} \sqrt{\frac{\sum \sum_{1 \leq i < j \leq n} (\delta_{ij} - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} d_{ij}^2(X)}}$$

11.4 Guttman's Approach

The main alternative to the Kruskal approach to MDS, besides smacof, is the Smallest Space Analysis (SSA) of Guttman and Lingoes. I have ambiguous feelings about the fundamental SSA paper of Guttman (1968). It is, no doubt, a milestone MDS paper, and some of the distinctions it makes (which we will discuss later in this section) are clearly important. Its use of matrix algebra, wherever possible, is an improvement over Kruskal (1964a), and the correction matrix algorithm for SSA is an immediate predecessor of smacof. But it seems to me the derivation of the correction matrix algorithm is incomplete and could even be called incorrect. The rank images used by Guttman and Lingoes in SSA seem an ad-hoc solution invented by someone who did not yet know about monotone regression. And, above all, the paper exudes a personality cult-like atmosphere that is somewhat repellent to me. There are no gurus in science. Or at least there should not be. It is true that between 1930 and 1960 Guttman invented and elucidated about 75% of the psychometrics of his time, but 75% is still less than 100%. This book you are reading now may set a record in self-citation, but that makes sense because it is supposed to document my work in MDS and to give access to the pdf's of my unpublished work. I try to be careful not to take credit for results that did not originate with me, and to give appropriate attributions in all cases.

11.4.0.1 Rank Images

The rank image transformation, which replaces the monotone regression in Kruskal's approach, has a rather complicated definition. It is simple enough

Table 11.1: Semi-strong Rank Images

	1	2	3	4	5	6	7
$\$\\Delta\$$	1	2	2	3	3	4	5
$\$D(X)\$$	1	3	1	3	4	3	4
$\$D(X) \\ \\text{ordered}\$$	1	1	3	3	3	4	4
$\$D^*\\star\$$	1	1	3	3	3	4	4

when both Δ and $D(X)$ have no ties. In that case the rank image D^* is just the unique permutation of $D(X)$ that is monotone with Δ . Thus

$$\delta_{ij} < \delta_{kl} \Leftrightarrow d_{ij}^* < d_{kl}^*. \quad (11.10)$$

If there are ties in Δ and/or $D(X)$ then some of the uniqueness and simplicity will get lost. Guttman (1968) introduces an elaborate notation for rank images with ties, but that notation does neither him nor the reader any favors.

If there are ties in $D(X)$ you use the rank order of the corresponding elements of Δ to order $D(X)$ within tie blocks. If two elements are tied both in $D(X)$ and Δ , then their order in the tie block is arbitrary.

Suppose the Δ have R tie-blocks, in increasing order, with m_1, \dots, m_R elements. The smallest m_1 elements of the vector of distances become the first m_1 elements of D^* , the next m_2 elements of D^* are the next smallest m^2 elements of distance vector, and so on for all tie blocks. Thus tied elements in Δ can become untied in D^* and untied elements in Δ can becomes tied in D^* . We require

$$\delta_{ij} < \delta_{kl} \Rightarrow d_{ij}^* \leq d_{kl}^* \quad (11.11)$$

This corresponds with Kruskal's primary approach to ties. Guttman calls it *semi-strong monotonicity*. There is a small numerical example in table 11.1.

The sum of the squared differences between $D(X)$ and D^* is 6.

Alternatively, we can require that tied elements in Δ correspond with tied elements in D^* . Guttman calls this *strong monotonicity*, and requires in addition that D^* has the same number of blocks, with the same block sizes,

Table 11.2: Strong Rank Images

	1	2	3	4	5	6	7
\$\Delta\$	1	2	2	3	3	4	5
\$D(X)\$	1	3	1	3	4	3	4
\$D(X) \setminus \text{ordered}\$	1	1	3	3	3	4	4
\$D^*_{\text{star}}\$	1	2	2	3	3	4	4

as Δ . Instead of copying ordered blocks from the sorted distances, we compute averages of blocks, and copy those into D^*_{star} . This corresponds with Kruskal's secondary approach to ties. Thus the elements of D^* are no longer a permutation of those in $D(X)$. We have @eq:nmrankimage1, and also

$$\delta_{ij} = \delta_{kl} \Rightarrow d_{ij}^* = d_{kl}^* \quad (11.12)$$

Our numerical example is now in table 11.2.

Now the sum of squared differences between $D(X)$ and D^* is 4, which means, surprisingly, that strong monotonicity gives a better fit than semi-strong monotonicity. This cannot happen with monotone regression, where the primary approach to ties always has a better fit than the secondary approach.

11.4.0.2 Single and Double Phase

Note: suppose $\|D_1^* - D(X_2)\|^2 < \|D_1^* - D(X_1)\|^2$ but $\|D_2^* - D(X_2)\|^2 > \|D_1^* - D(X_2)\|^2$

$$\sigma_G(X) = \frac{\sum \sum_{1 \leq i < j \leq n} w_{ij} (d_{ij}^*(X) - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} w_{ij} d_{ij}^2(X)}.$$

$$\sigma_G(X, D^*) = \frac{\sum \sum_{1 \leq i < j \leq n} (d_{ij}^* - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} d_{ij}^2(X)}.$$

11.4.0.3 Hard and Soft Squeeze

$$\sigma(X) := \min_{\delta \in \mathbb{K} \cap \mathbb{S}} \sum_{k \in \mathcal{K}} w_k (\delta_k - d_k(X))^2$$

question: in double phase do rank images decrease stress ? My guess is yes.
Are they continuous ?

$$\rho_G(X) = \max_{P \in \Pi} \delta' P d(X)$$

is a continuous function of X . Also (Shepard)

$$D_+ \rho_G(X) = \max_{P \in \Pi(X)} \delta' P \mathcal{D}d(X)$$

rank-images Pd are not continuous

11.4.1 Smoothness of Ordinal Loss Functions

Kruskal

$$\min_{\hat{D} \in \mathfrak{D}} \sum \sum w_{ij} (\hat{d}_{ij} - d_{ij}(X))^2$$

is a differentiable function of X

Double phase rank image

$$\sigma_{LG}(X) = \min_P \|Pd(X) - d(X)\|^2$$

P in the Birkhoff polytope and satisfying inequalities, equalities.

P given by $\max_P d(X)' P d(X)$ De Leeuw (1973b)

11.5 Scaling with Distance Bounds

11.6 Bounds on Stress

Chapter 12

Unidimensional Scaling

For Unidimensional Scaling (UDS or 1MDS) the configuration is a matrix $X \in \mathbb{R}^{n \times 1}$. We can trivially identify the single-column matrix X with the vector x of coordinates of n points on the real line. All our previous general MDS results remain valid for UDS, but we will use the additional structure that comes with $p = 1$ to discuss a number of special results.

Unidimensional scaling under different names in the literature such as seriation in archeology and sequencing in genetics. Often the algorithms permute the rows and columns of a matrix of dissimilarities to approximate some special structure. In this book seriation and sequencing are always understood to be the minimization of stress over x , i.e. minimization of

$$\sigma(x) = \sum_{1 \leq i < j \leq n} \sum w_{ij} (\delta_{ij} - |x_i - x_j|)^2 \quad (12.1)$$

12.1 An example

We start the chapter with some pictures, similar to the ones in chapter 4. There are four objects. Dissimilarities are again chosen to be all equal, in this case to $\frac{1}{6}\sqrt{6}$. Weights are all equal to one.

We look at stress on the two-dimensional subspace spanned by the two vectors $y = (0, -1, +1, 0)$ and $z = (-1.5, -.5, +.5, +1.5)$. First we nor-

malize both y and z by $\rho = \eta^2$. This gives $y = (0, -\frac{1}{8}\sqrt{6}, +\frac{1}{8}\sqrt{6}, 0)$ and $z = (-\frac{1}{8}\sqrt{6}, -\frac{1}{24}\sqrt{6}, +\frac{1}{24}\sqrt{6}, +\frac{1}{8}\sqrt{6})$.

We know from previous results (for example, De Leeuw and Stoop (1984)) that the equally spaced configuration z is the global minimizer of stress over \mathbb{R}^4 . Of course it is far from unique, because all 24 permutations of z have the same function value, and are consequently also global minima. In fact, there are 24 local minima, which are all global minima as well.

In the example, we do not minimize over all of \mathbb{R}^4 , but only over the subspace of linear combinations of y and z . These linear combinations, with coefficients α and β , are given by

$$x = \alpha y + \beta z = \frac{1}{24}\sqrt{6} \begin{bmatrix} -3\beta \\ -3\alpha - \beta \\ 3\alpha + \beta \\ 3\beta \end{bmatrix}, \quad (12.2)$$

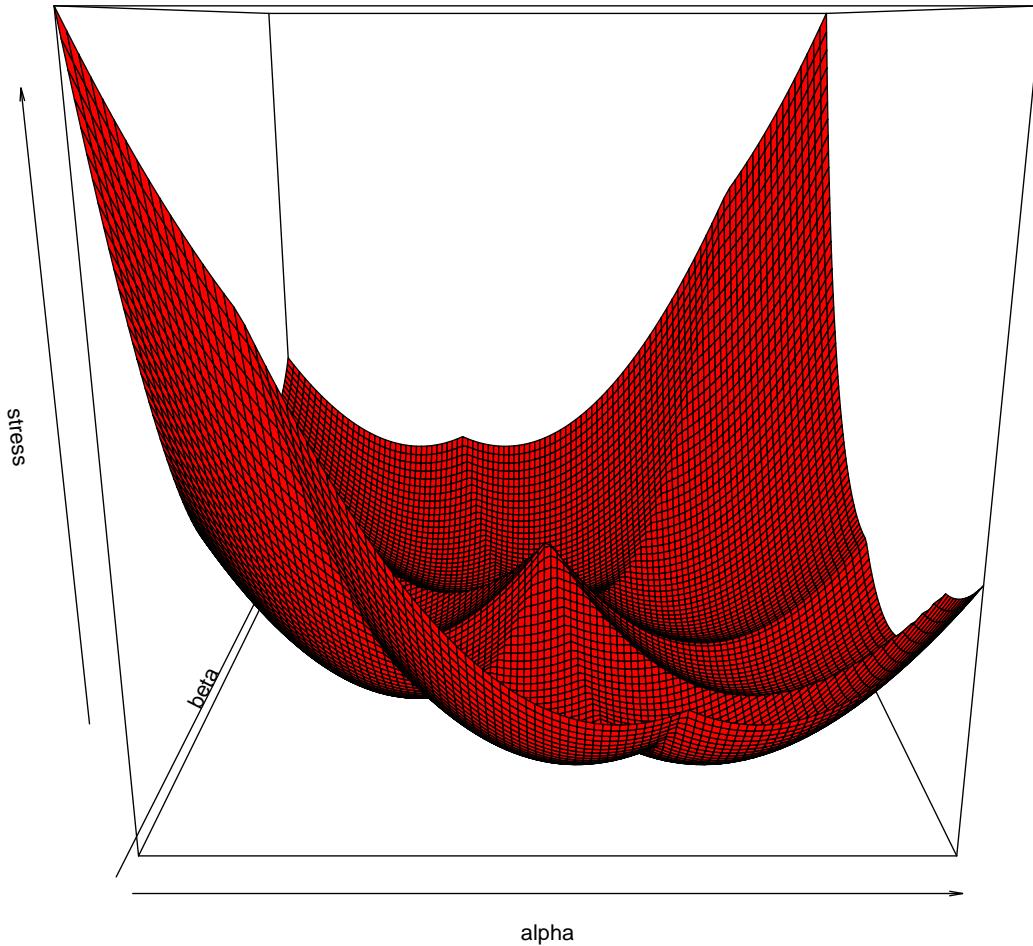
with distances

$$D(x) = \frac{1}{24}\sqrt{6} \begin{bmatrix} 0 & & & \\ |3\alpha - 2\beta| & 0 & & \\ |3\alpha + 4\beta| & |6\alpha + 2\beta| & 0 & \\ |6\beta| & |3\alpha + 4\beta| & |3\alpha - 2\beta| & 0 \end{bmatrix}. \quad (12.3)$$

We see that on the line $\beta = \frac{3}{2}\alpha$ both $d_{12}(x)$ and $d_{34}(x)$ are zero, on $\beta = -3\alpha$ we have $d_{23}(x) = 0$, on $\beta = 0$ we have $d_{14}(x) = 0$, and finally $d_{13}(x) = d_{24}(x) = 0$ on $\beta = -\frac{3}{4}\alpha$. On those lines, through the origin, stress is not differentiable.

12.1.1 Perspective

We first make a global perspective plot, with both α and β in the range $(-2.0, +2.0)$.

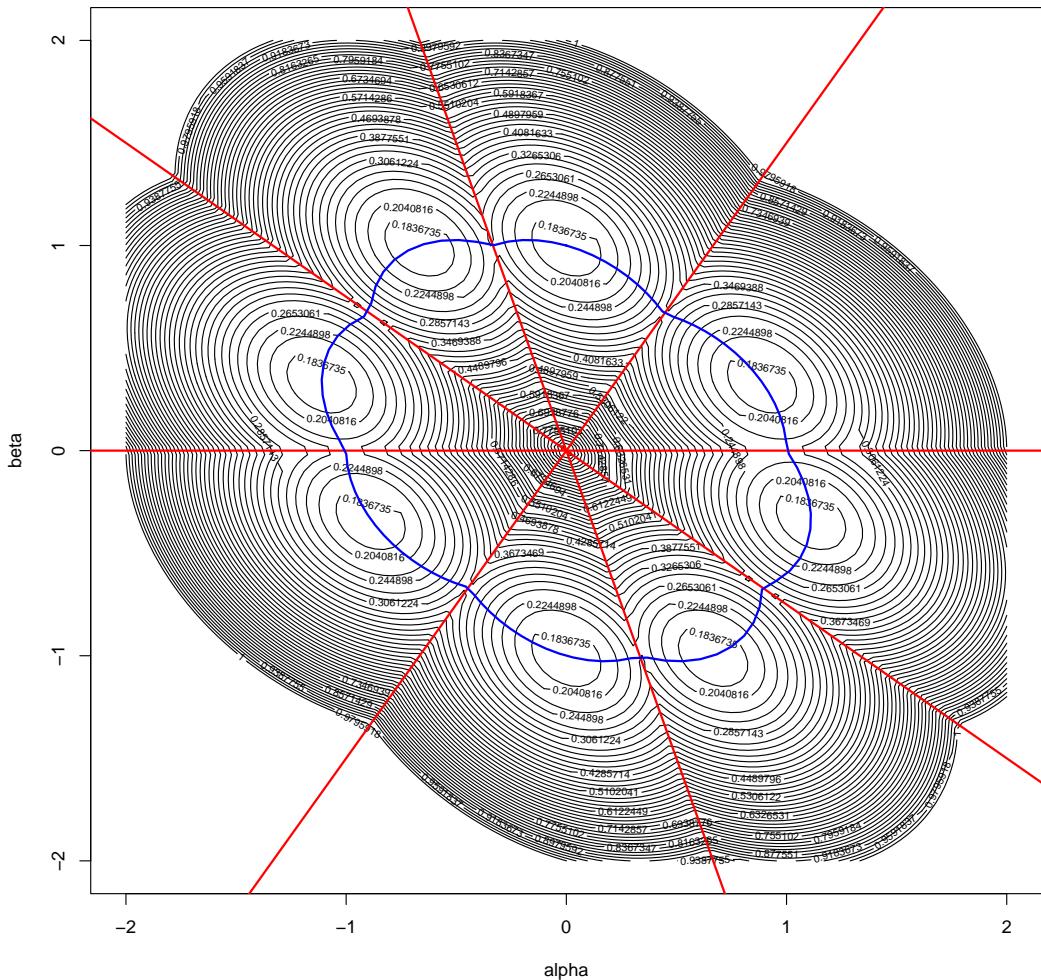


What do we see ? Definitely more ridges and valleys than in the two-dimensional example of chapter 4. In the one-dimensional case there is a ridge wherever two coordinates are equal, and thus one or more distances are zero. It is clear that at the bottom of each of the valleys there sits a local minimum.

12.1.2 Contour

A contour plot gives some additional detail. In the plot we have drawn the four lines through the origin where one or more distances are zero (in red), and we have drawn the curve where $\eta^2(x) = \rho(x)$ (in blue). Thus all local

minima are on the blue line. The intersections of the red and the blue lines are the local minima of stress restricted to the red line. In those points there are both directions of ascent (along the red lines, in both directions) and of descent (into the adjoining valleys, in all directions).



We see once more the importance of the local minimum result from De Leeuw (1984c) that we discussed in section 2.6.2. The special relevance of this result for UMDS was already pointed out by Pliner (1996). At a local minimum all distances are positive, and thus local minima must be in the interior of the eight cones defined by the four zero-distance lines. There are no saddle points, and only the single local maximum at the origin.

12.2 Order Formulation

Define an *isocone* as a closed convex cone of isotone vectors, and $\text{int}(K)$ as its interior. Thus

$$K := \{x \in \mathbb{R}^n \mid x_{i_1} \leq \cdots \leq x_{i_n}\}, \quad (12.4)$$

and

$$\text{int}(K) = \{x \in \mathbb{R}^n \mid x_{i_1} < \cdots < x_{i_n}\}. \quad (12.5)$$

where (i_1, \dots, i_n) is a permutation of $(1, \dots, n)$. There are $n!$ such closed isocones, and their union is all of \mathbb{R}^n . Thus $\min_x \sigma(x) = \min_{K \in \mathcal{K}} \min_{x \in K} \sigma(x) =$ where \mathcal{K} are the $n!$ isocones.

For UMDS purposes the isocones are paired, because the negative of each configuration has the same distances between the n points, and thus the same stress. Thus each isocone and its negative cone are equivalent for UMDS, and we only have to consider $(n!)/2$ distinct orders.

Let us consider the problem of minimizing σ over a fixed $K \in \mathcal{K}$. Now

$$\rho(x) = \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} s_{ij} (x_i - x_j),$$

and $s_{ij} = \text{sign}(x_i - x_j)$ is the sign matrix of x . Sign matrices, strict sign matrices, the set \mathcal{S} of sign matrices, and the sign matrix function S are defined in 25.2.4.

For all $x \in \text{int}(K)$ the matrix S is the same strict sign matrix. Now

$$\rho(x) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij} (x_i - x_j) = x' t_K,$$

where t_K is the vector of row sums of the Hadamard product $W \times \Delta \times S$, or

$$\{t_K\}_i := \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}.$$

Again t_K only depends of K , not on x as long as $x \in \text{int}(K)$.

Thus on K

$$\sigma(x) = 1 - 2x't_K + x'Vx = 1 + (x - V^{-1}t_K)'V(x - V^{-1}t_K) - t'_K V^{-1} t_K.$$

If there are no weights the t_K were first defined using isocones in De Leeuw and Heiser (1977). They point out that minimizing $(x - V^{-1}t)'V(x - V^{-1}t)$ over $x \in K$ is a monotone regression problem (see 11.1).

A crucial next step is in De Leeuw (2005c), using the basic result in De Leeuw (1984c). De Leeuw (2005c) does use weights. We know if x is a local minimum then it must be in the interior of the isocone. If $V^{-1}t_K$ is not in interior, then monotone regression will creates ties, and thus x will not be in the interior either. In fact for local minima of UMDS it is necessary and sufficient that $V^{-1}t_K$ is in the interior of K . This result, without weights and in somewhat different language, is also in Pliner (1984). Thus we can limit our search to those isocones for which $V^{-1}t_K \in \text{int}(K)$. For those isocones, say the set \mathcal{K}° , the local minimum is at $x = V^{-1}t_K$.

Thus

$$\min_{K \in \mathcal{K}} \min_{x \in K} \sigma(x) = 1 - \max_{K \in \mathcal{K}^\circ} t'_K V^{-1} t_K.$$

There is also an early short but excellent paper by Defays (1978), which derives basically the same result in a non-geometrical way. Defays does not use weights, so in his paper V^{-1} is $n^{-1}I$.

In the two-dimensional subspace of the example some of the $n!$ cones are empty.

12.3 Permutation Formulation

12.4 Sign Matrix Formulation

Sign matrices, strict sign matrices, the set \mathcal{S} of sign matrices, and the sign matrix function S are defined in 25.2.4. Using sign matrices we can write

$$\rho(x) = \max_{S \in \mathcal{S}} \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} s_{ij} (x_i - x_j), \quad (12.6)$$

with the maximum attained for $S = S(x)$. If we define

$$t_i(y) := \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}(y), \quad (12.7)$$

then

$$\sigma(x) = \min_y \{1 + (x - V^{-1}t(y))'V(x - V^{-1}t(y)) - t(y)'V^{-1}t(y)\}. \quad (12.8)$$

This implies

$$\min_x \sigma(x) = 1 - \max_y t(y)'V^{-1}t(y) \quad (12.9)$$

12.5 Algorithms for UMDS

12.5.1 SMACOF

12.5.2 SMACOF (smoothed)

Now local minimum $x_i \neq x_j$

$$\min_{x \in K} \sigma(x) = \sum_{1 \leq i < j \leq n} \sum w_{ij} (\delta_{ij} - s_{ij}(x))(x_i - x_j)^2$$

Each isocone has a sign matrix (hollow, antisymmetric)

$$s_{ij}(x) = \begin{cases} +1 & \text{if } x_i > x_j, \\ -1 & \text{if } x_i < x_j, \\ 0 & \text{if } x_i = x_j. \end{cases}$$

$$\rho(x) = \sum_{i=1}^n \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}(x)(x_i - x_j) \geq \sum_{i=1}^n \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}(y)(x_i - x_j) = 2 \sum_{i=1}^n x_i \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}(y)$$

Now

$$\rho(x) = \sum_{i=1}^n \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}(x)(x_i - x_j), \quad (12.10)$$

and for all $y \in \mathbb{R}^n$

$$\rho(x) \geq \sum_{i=1}^n \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}(y)(x_i - x_j) = 2 \sum_{i=1}^n x_i \sum_{j=1}^n w_{ij} \delta_{ij} s_{ij}(y). \quad (12.11)$$

Stress is the maximum of a finite number of quadratics.

12.5.3 Branch-and-Bound

12.5.4 Dynamic Programming

12.5.5 Simulated Annealing

12.5.6 Penalty Methods

Chapter 13

Full-dimensional Scaling

13.1 Convexity

13.2 Optimality

13.3 Iteration

13.4 Cross Product Space

So far we have formulated the MDS problem in *configuration space*. Stress is a function of X , the $n \times p$ configuration matrix. We now consider an alternative formulation, where stress is a function of a positive semi-definite C of order n . The relevant definitions are

$$\sigma(C) := 1 - 2\rho(C) + \eta(C), \quad (13.1)$$

where

$$\begin{aligned} \rho(C) &:= \mathbf{tr} B(C)C, \\ \eta(C) &:= \mathbf{tr} VC, \end{aligned}$$

with

$$B(C) := \sum_{1 \leq i < j \leq n} \sum \begin{cases} w_{ij} \frac{\delta_{ij}}{d_{ij}(C)} A_{ij} & \text{if } d_{ij}(C) > 0, \\ 0 & \text{if } d_{ij}(C) = 0. \end{cases}$$

and $d_{ij}^2(C) := \mathbf{tr} A_{ij} C$.

We call the space of all positive semi-definite $n \times n$ matrices *cross product space*. The problem of minimizing σ over $n \times p$ -dimensional configuration space is equivalent to the problem of minimizing σ over the set of matrices C in $n \times n$ -dimensional cross product space that have rank less than or equal to p . The corresponding solutions are related by the simple relationship $C = XX'$.

Theorem 13.1. *Stress is convex on cross product space.*

Proof. First, η is linear in C . Second,

$$\rho(C) = \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij} \sqrt{\mathbf{tr} A_{ij} C}.$$

This is the weighted sum of square roots of non-negative functions that are linear in C , and it is consequently concave. Thus σ is convex. \square

Unfortunately the subset of cross product space of all matrices with rank less than or equal to p is far from simple (see Datorro (2015)), so computational approaches to MDS prefer to work in configuration space.

13.5 Full-dimensional Scaling

Cross product space, the set of all positive semi-definite matrices, is a closed convex cone \mathcal{K} in the linear space of all $n \times n$ symmetric matrices. This has an interesting consequence.

Theorem 13.2. *Full-dimensional scaling, i.e. minimizing σ over \mathcal{K} , is a convex programming problem. Thus in FDS all local minima are global. If $w_{ij} \delta_{ij} > 0$ for all i, j then the minimum is unique.*

This result has been around since about 1985. De Leeuw (1993) gives a proof, but the report it appeared in remained unpublished. A published proof is in De Leeuw and Groenen (1997). Another treatment of FDS, with a somewhat different emphasis, is in De Leeuw (2014a).

Now, by a familiar theorem (Theorem 31.4 in Rockafellar (1970)), a matrix C minimizes σ over \mathcal{K} if and only if

$$C \in \mathcal{K}, \quad (13.2)$$

$$V - B(C) \in \mathcal{K}, \quad (13.3)$$

$$\mathbf{tr} C(V - B(C)) = 0. \quad (13.4)$$

We give a computational proof of this result for FDS that actually yields a bit more.

Theorem 13.3. *For $\Delta \in \mathcal{K}$ we have*

$$\sigma(C + \epsilon\Delta) = \sigma(C) - 2\epsilon^{\frac{1}{2}} \sum_{\mathbf{tr} A_i C = 0} w_i \delta_i \sqrt{\mathbf{tr} A_i \Delta} + \epsilon \mathbf{tr} (V - B(C))\Delta + o(\epsilon). \quad (13.5)$$

::: {.proof} Simple expansion. :::

Theorem 13.4. *Suppose C is a solution to the problem of minimizing σ over \mathcal{K} . Then*

- $\mathbf{tr} A_{ij}C > 0$ for all i, j for which $w_{ij}\delta_{ij} > 0$.
- $V - B(C)$ is positive semi-definite.
- $\mathbf{tr} C(V - B(C)) = 0$.
- If C is positive definite then $V = B(C)$ and $\sigma(C) = 0$.

Proof. The $\epsilon^{\frac{1}{2}}$ term in (13.5) needs to vanish at a local minimum. This proves the first part. It follows that at a local minimum

$$\sigma(C + \epsilon\Delta) = \sigma(C) + \epsilon \mathbf{tr} (V - B(C))\Delta + o(\epsilon).$$

If $V - B(C)$ is not positive semi-definite, then there is a $\Delta \in \mathcal{K}$ such that $\mathbf{tr} (V - B(C))\Delta < 0$. Thus C cannot be the minimum, which proves the second part. If we choose $\Delta = C$ we find

$$\sigma((1 + \epsilon)C) = \sigma(C) + \epsilon \operatorname{tr} (V - B(C))C + o(\epsilon).$$

and choosing ϵ small and negative shows we must have $\operatorname{tr} (V - B(C))C = 0$ for C to be a minimum. This proves the third part. Finally, if σ has a minimum at C , and C is positive definite, then from parts 2 and 3 we have $V = B(C)$. Comparing off-diagonal elements shows $\Delta = D(C)$, and thus $\sigma(C) = 0$. \square

If C is the solution of the FDS problem, then $\operatorname{rank}(C)$ defines the *Gower rank* of the dissimilarities. The number of positive eigenvalues of the negative of the doubly-centered matrix of squared dissimilarities, the matrix factored in classical MDS, defines the *Torgerson rank* of the dissimilarities. The *Gower conjecture* is that the Gower rank is less than or equal to the Torgerson rank. No proof and no counter examples have been found.

We compute the FDS solution using the smacof algorithm

$$X^{(k+1)} = V^+ B(X^{(k)}) \quad (13.6)$$

in the space of all $n \times n$ configurations, using the identity matrix as a default starting point. Since we work in configuration space, not in crossproduct space, this does not guarantee convergence to the unique FDS solution, but after convergence we can easily check the necessary and sufficient conditions of theorem 13.4.

As a small example, consider four points with all dissimilarities equal to one, except δ_{14} which is equal to three. Clearly the triangle inequality is violated, and thus there certainly is no perfect fit mapping into Euclidean space.

The FDS solution turns out to have rank two, thus the Gower rank is two. The singular values of the FDS solution are

```
## [1] 0.4508464709 0.2125310645 0.0000001303
```

Gower rank two also follows from the eigenvalues of the matrix $B(C)$, which are

```
## [1] 1.0000000000 1.0000000000 0.9205543464
```

13.6 Ekman example

The Ekman (1954) color data give similarities between 14 colors.

```
##      434  445  465  472  490  504  537  555  584  600  610  628  651
## 445 0.86
## 465 0.42 0.50
## 472 0.42 0.44 0.81
## 490 0.18 0.22 0.47 0.54
## 504 0.06 0.09 0.17 0.25 0.61
## 537 0.07 0.07 0.10 0.10 0.31 0.62
## 555 0.04 0.07 0.08 0.09 0.26 0.45 0.73
## 584 0.02 0.02 0.02 0.02 0.07 0.14 0.22 0.33
## 600 0.07 0.04 0.01 0.01 0.02 0.08 0.14 0.19 0.58
## 610 0.09 0.07 0.02 0.00 0.02 0.02 0.05 0.04 0.37 0.74
## 628 0.12 0.11 0.01 0.01 0.01 0.02 0.02 0.03 0.27 0.50 0.76
## 651 0.13 0.13 0.05 0.02 0.02 0.02 0.02 0.02 0.20 0.41 0.62 0.85
## 674 0.16 0.14 0.03 0.04 0.00 0.01 0.00 0.02 0.23 0.28 0.55 0.68 0.76
```

We use three different transformations of the similarities to dissimilarities. The first is $1 - x$, the second $(1 - x)^3$ and the third $\sqrt[3]{1 - x}$. We need the following iterations to find the FDS solution (up to a change in loss of 1e-15).

```
## power = 1.00  itel =     6936  stress =  0.0000875293
## power = 3.00  itel =      171   stress =  0.0110248119
## power = 0.33  itel =     423    stress =  0.0000000000
```

For the same three solutions we compute singular values of the thirteen-dimensional FDS solution.

```
## [1] 0.1797609824 0.1454675297 0.0843865491 0.0777136109 0.0486123551
## [6] 0.0393576522 0.0236290817 0.0162344515 0.0072756171 0.0000031164
## [11] 0.0000000009 0.0000000000 0.0000000000
##
## [1] 0.2159661347 0.1549184093 0.0000000727 0.0000000041 0.0000000000
## [6] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

```

## [11] 0.0000000000 0.0000000000 0.0000000000
##
## [1] 0.1336126813 0.1139019875 0.0880453752 0.0851609618 0.0710424935
## [6] 0.0664988952 0.0561005006 0.0535112029 0.0492295395 0.0479964575
## [11] 0.0468628701 0.0410193579 0.0388896490

```

Thus the Gower ranks of the transformed dissimilarities are, respectively, nine (or ten), two, and thirteen. Note that for the second set of dissimilarities, with Gower rank two, the first two principal components of the thirteen-dimensional solution are the global minimizer in two dimensions. To illustrate the Gower rank in yet another way we give the thirteen non-zero eigenvalues of $V^+B(X)$, so that the Gower rank is the number of eigenvalues equal to one. All three solutions satisfy the necessary and sufficient conditions for a global FDS solution.

```

## [1] 1.0000000432 1.0000000222 1.0000000012 1.0000000005 1.0000000002
## [6] 1.0000000001 1.0000000000 1.0000000000 0.9999993553 0.9989115116
## [11] 0.9976821885 0.9942484083 0.9825147154
##
## [1] 1.0000000000 1.0000000000 0.9234970864 0.9079012130 0.8629365849
## [6] 0.8526920031 0.8298036209 0.8145561677 0.7932385763 0.7916517225
## [11] 0.7864426781 0.7476794757 0.7282682474
##
## [1] 1.0000000820 1.0000000241 1.0000000047 1.0000000009 1.0000000004
## [6] 1.0000000003 1.0000000001 1.0000000001 1.0000000001 1.0000000000
## [11] 0.9999999999 0.9999999689 0.9999999005

```

We also plot the first two principal components of the thirteen-dimensional FDS solution. Not surprisingly, they look most circular and regular for the solution with power three, because this actually is the global minimum over two-dimensional solutions. The other configurations still have quite a lot of variation in the remaining dimensions.

Figure @ref{fig:ekmantrans} illustrates that the FDS solution with power 3 is quite different from power 1 and power one 1/3. Basically the transformations with lower powers result in dissimilarity measures that are very similar to Euclidean distances in a high-dimensional configuration, while power equal to 3 makes the dissimilarities less Euclidean. This follows from metric transform

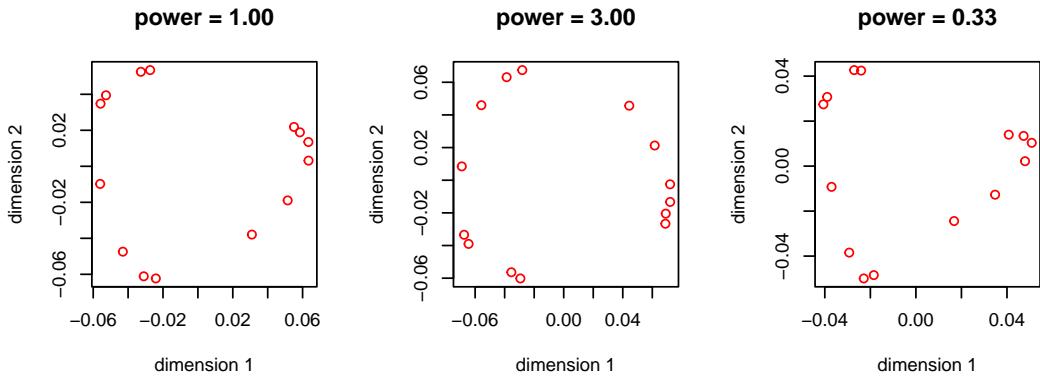


Figure 13.1: Ekman data, configurations for three powers

theory, where concave increasing transforms of finite metric spaces tend to be Euclidean. In particular the square root transformation of a finite metric space has the Euclidean four-point property, and there is a $c > 0$ such that the metric transform $f(t) = ct/(1+ct)$ makes a finite metric space Euclidean (Maehara (1986)).

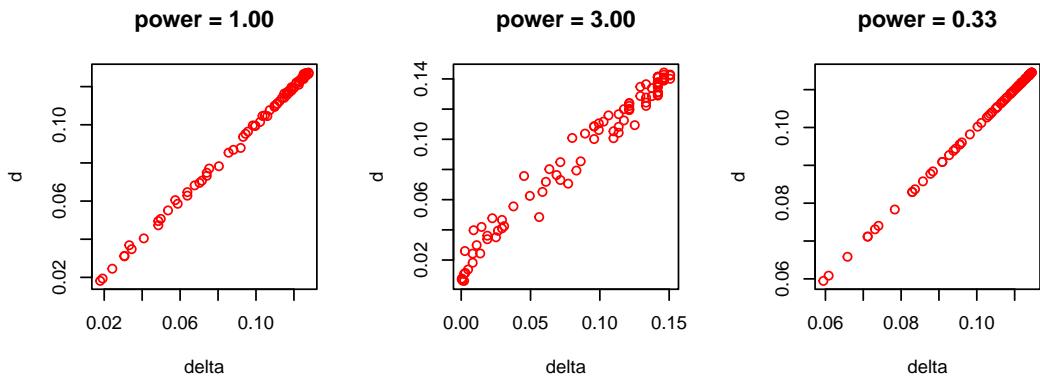


Figure 13.2: Ekman data, fit plots for three powers

Chapter 14

Unfolding

In unfolding the objects are partitioned into two sets of, say, n and m objects and only the nm between-set dissimilarities are observed. The within-set weights are zero. Thus we minimize

$$\sigma(X, Y) = \sum_{i=1}^n \sum_{j=1}^m w_{ij} (\delta_{ij} - d(x_i, y_j))^2 \quad (14.1)$$

over $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^{m \times p}$.

$\frac{1}{2}\{(n+m)(n+m-1) - 2nm\}$ are missing

note that we can have $d_{ij}(X) = 0$ at a local minimum.

14.1 Algebra

The missing data in unfolding complicate the MDS problem, in the same way as the singular value decomposition of a rectangular matrix is more complicated than the eigen decomposition of a symmetric matrix.

The problem we want to solve in this section is recovering X and Y (up to a translation and rotation) from $D(X, Y)$.

The first matrix algebra results in metric unfolding were due to Ross and Cliff (1964). An actual algorithm for the “ignore-errors” case was proposed by Schönemann (1970). Schönemann’s technique was studied in more detail by Gold (1973) and Heiser and De Leeuw (1979).

This section discusses a slightly modified version of Schönemann (1970).

First, we compute the Torgerson transform $E(X, Y) = -\frac{1}{2}JD^{(2)}(X, Y)J$. It was observed for the first time by Ross and Cliff (1964) that $E(X, Y) = JXY'J$.

Assume that $E(X, Y) = JXY'J$ is a full-rank decomposition, and that the rank of $E(X, Y)$ is r . Note that there are cases in which the rank of JX or JY is strictly smaller than the rank of X or Y . If X , for example, has columns x and $e - x$, with x and e linearly independent, then its rank is two, while JX with columns Jx and $-Jx$ has rank one.

Suppose $E(X, Y) = GH'$ is another full-rank decomposition. Then there exist vectors u and v with r elements and a non-singular T of order r such that

$$\begin{aligned} X &= GT + eu', \\ Y &= HT^{-t} + ev'. \end{aligned} \tag{14.2}$$

We can assume without loss of generality that the centroid of the X configuration is in the origin, so that $JX = X$, and $u = 0$ in the first equation of (14.2).

We use the QR decomposition to compute the rank r of $E(X, Y)$, and the factors G and H .

We now use (14.2) to show that $F = D^{(2)}(X, Y) + 2GH'$ is of the form $F = \gamma + \alpha e' + e\beta'$, with $\gamma = v'v$ and $M = TT'$.

$$\begin{aligned} \alpha_i &= g'_i Mg_i - 2g'_iT v, \\ \beta_j &= h'_j M^{-1} h_j + 2h'_j T^{-t} v. \end{aligned} \tag{14.3}$$

It follows that $JF = J\alpha e'$ and $FJ = e\beta'J$. Thus $J\alpha$ is any column of JF and $J\beta$ is any row of FJ .

Consider the first equation of (14.3). For the time being, we ignore the second one. Suppose M_k is a basis for the space of real symmetric matrices of order p with the $\frac{1}{2}p(p+1)$ elements $e_s e'_t + e_t e'_s$ for $s \neq t$ and $e_s e'_s$ for the diagonal. Define $q_{ik} := g'_i M_k g_i$. Then

$$J\alpha = J \begin{bmatrix} Q & -2G \end{bmatrix} \begin{bmatrix} \mu \\ Tv \end{bmatrix}, \tag{14.4}$$

with μ the coordinates of M for the basis M_k .

Equations (14.4) are n linear equations in the $\frac{1}{2}p(p+1) + p = \frac{1}{2}p(p+3)$ unknowns μ and Tv . Assume they have a unique solution. Then $M = \sum \mu_k M_k$ is PSD, and can be eigen-decomposed as $M = K\Lambda^2K'$. Set $T = K\Lambda$

```
set.seed(12345)
x <- matrix(rnorm(16), 8, 2)
x <- apply(x, 2, function(x) x - mean(x))
y <- matrix(rnorm(10), 5, 2)
a <- rowSums(x ^ 2)
b <- rowSums(y ^ 2)
d <- sqrt(outer(a, b, "+") - 2 * tcrossprod(x, y))
```

14.1.1 One-dimensional

The one-dimensional case is of special interest, because it allows us to construct a single joint metric scale for row objects and column objects from metric dissimilarities. We have to find a solution to $\delta_{ij} = |x_i - y_j|$, without making assumptions about the order of the projections on the dimension. Compute any solution for Jg and Jh from $\tau(\Delta^{(2)}) = Jgh'J$. For data with errors we would probably use the SVD. Assume without loss of generality that $Jg = g$. Then the general solution is $x = \tau g$ and $y = \tau^{-1}h + \nu e$ for some real τ and ν .

Now

$$\Delta^2 = \tau^2 g^{(2)}e' + \tau^{-2}e(h'_j)^{(2)} + \nu^2 E - 2gh' - 2\tau\nu g_ie' \quad (14.5)$$

are nm equations in the two unknowns (τ, ν) . They can be solved by many methods, but we go the Schönemann way. Column-centering gives

$$J(\Delta^{(2)} + 2g_jh_j) = \tau^2 Jg^{(2)} - 2\tau\nu g_i \quad (14.6)$$

while row-centering gives

$$(\Delta^{(2)} + 2g_jh_j)J = \tau^{-2}e(h^{(2)})'J. \quad (14.7)$$

14.2 Classical Unfolding

Multidimensional unfolding as a data analysis technique was introduced by Coombs (1964).

bennett-hays hays-bennett bennett

SMACOF - Heiser and De Leeuw (1979)

Form of V

What happens to nonzero theorem ? within-set distances can be zero

$$\Delta = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

$x_1 = x_2 = 0$ $y_1 = 1, y_2 = 2, y_3 = 3$

14.3 Nonmetric Unfolding

row-conditional busing van deun deleeuw_R_06a

Stress3 – Roskam

14.3.1 Degenerate Solutions

What are they

14.3.1.1 Which Stress

$$\sigma(X) = \sum_{i=1}^n \sum_{j=1}^n w_{ij} (\delta_{ij} - d_{ij}(X))^2$$

Weak order, plus normalization. Two-point solution.

14.3.1.2 l'Hopital

Illustration.

$\delta_{12} > \delta_{13} = \delta_{23}$.

$$d_{12}(X_\epsilon) = 1$$

$$d_{13}(X_\epsilon) = d_{23}(X_\epsilon) = 1 + \frac{1}{2}\epsilon.$$

Then

$$\lim_{\epsilon \rightarrow 0} D(X_\epsilon) = \Delta.$$

euclidean for $\epsilon \geq -1$

14.3.1.3 Penalizing

14.3.1.4 Restricting Regression

Busing

Van Deun

Chapter 15

Constrained Multidimensional Scaling

As we have seen in section 1.3 CMDS is defined as the generalization of basic MDS in which we want to solve

$$\min_{X \in \Omega} \sum_{1 \leq i < j \leq n} w_{ij}(\delta_{ij} - d_{ij}(X))^2,$$

where Ω is a subset of $\mathbb{R}^{n \times p}$.

$X = F(\theta)$ vs $F(X) \geq 0$ $G(X) = 0$

Primal-Dual

Configuration-Distances $F(D(X)) \geq 0$ and $G(D(X)) = 0$

15.1 Basic Partitioning

In the smacof approach a comprehensive approach to constrained MDS was developed in De Leeuw and Heiser (1980). It is a primal method that does not involve penalty parameters, and it defines the constraints directly on the configuration.

The starting point is the *majorization partitioning*.

$$\sigma(X) \leq 1 + \eta^2(X - \mathfrak{G}(Y)) - \eta^2(\mathfrak{G}(Y)), \quad (15.1)$$

with equality, of course, if $X = Y$.

The smacof algorithm for constrained MDS has consequently two steps. In the first we compute the Guttman transform of the current configuration, and in the second we find the metric projection of this Guttman transform on the constraint set (in the metric defined by V). Thus, in shorthand,

$$X^{(k+1)} \in \operatorname{Argmin}_{Y \in \Omega} \eta^2(Y - \Gamma(X^{(k)})). \quad (15.2)$$

To emphasize we look for a fixed point of the composition of two maps, the Guttman transform and the projection operator Π_Ω , we can write in even shorter hand

$$X^{(k+1)} \in \Pi_\Omega(\Gamma(X^{(k)}))$$

The smacof formulation of the CMDS problem is elegant, if I say so myself, but it is not always simple from the computational point of view. The Guttman transform is easy enough to compute, but projecting on Ω in the V metric may be complicated, depending on how Ω is defined. In this chapter we will discuss a number of examples with varying degrees of difficulty in computing the *smacof projection*.

15.2 Unweigthing

For some types of constraints, for example the circular and elliptical MDS discussed in section 15.5, unweighted least squares is computationally simpler than weighted least squares. In those cases it generally pays to use majorization to go from a weighted to an unweighted problem (see also Groenen, Giaquinto, and Kiers (2003)). This will tend to increase the number of iterations of smacof, but the computation within each iteration will be considerably faster.

From equation (15.2), the projection problem in constrained MDS is to minimize the weighted least squares loss function $\phi(X) := \operatorname{tr} (Z - X)'V(Z - X)$ over $X \in \Omega$. Now suppose θ is the largest eigenvalue of V , so that $V \lesssim \theta I$, and suppose $Y \in \Omega$. Then

$$\begin{aligned}\phi(X) = \text{tr} ((Z - Y) - (X - Y))'V((Z - Y) - (X - Y)) \leq \\ \phi(Y) - 2 \text{tr} (Z - Y)'V(X - Y) + \theta \text{tr} (X - Y)'(X - Y).\end{aligned}\quad (15.3)$$

Completing the square gives the majorization

$$\phi(X) \leq \phi(Y) + \theta \text{tr} (X - \bar{Y})'(X - \bar{Y}) - \theta \text{tr} \bar{Y}'\bar{Y}, \quad (15.4)$$

with \bar{Y} the matrix-convex combination

$$\bar{Y} := (I - \frac{1}{\theta}V)Y + \frac{1}{\theta}VZ. \quad (15.5)$$

The weighted projection problem from equation (15.2) is replaced by one or more inner iterations of an unweighted projection problem. Set $X^{(k,1)} = X^{(k)}$ and

$$X^{(k,l+1)} \in \underset{Y \in \Omega}{\text{Argmin}} \text{tr} (Y - \bar{X}^{k,l})'(Y - \bar{X}^{k,l}). \quad (15.6)$$

After stopping the inner iterations at $X^{(k,l+s)}$ we set $X^{(k+1)} = X^{(k,l+s)}$. All $X^{(k,l)}$ remain feasible, loss decreases in each inner iteration, and as long as the metric projections are continuous the map from $X^{(k)}$ to $X^{(k+1)}$ is continuous as well.

15.3 Constraints on the Distances

15.3.1 Rectangles

15.4 Linear Constraints

15.4.1 Uniqueness

$$X = (Z \mid D)$$

$$X = (Z \mid \alpha I)$$

Distance smoothing

15.4.2 Combinations

$$X = \sum \alpha_r Z_r$$

15.4.3 Step Size

$$X = Z + \alpha G$$

15.4.4 Single Design Matrix

$$X = ZU$$

15.4.5 Multiple Design Matrices

$$x_s = G_s u_s$$

$$d_{ij}^2(X) = \sum_{s=1}^p x_s' A_{ij} x_s = \sum_{s=1}^p u_s' G_s' A_{ij} G_s u_s$$

15.5 Circular MDS

De Leeuw (2007c), De Leeuw (2007a), De Leeuw (2005b)

There are situations in which it is desirable to have a configuration with points that are restricted to lie on some surface or manifold in \mathbb{R}^p . Simple examples are the circle in \mathbb{R}^2 or the sphere in \mathbb{R}^3 . Some applications are discussed in T. F. Cox and Cox (1991) (also see T. F. Cox and Cox (2001), section 4.6), in Borg and Lingoes (1980), in Papazoglou and Mylonas (2017), and in De Leeuw and Mair (2009), section 5. The most prominent examples are probably the color circle and the spherical surface of the earth, but there

are many other cases in which MDS solutions show some sort of “horseshoe” (De Leeuw (2007a)).

15.5.1 Some History



Figure 15.1: John van de Geer

Permit me to insert some personal history here. Around 1965 I got to work at the Psychological Institute. At the time Experimental Psychology and Methodology were in the same department, with John van de Geer as its chair. John had a long-running project with Pim Levelt and Reinier Plomp at the Institute for Perception RVO/TNO on perceptual and cognitive aspects of musical intervals. In Van de Geer, Levelt, and Plomp (1962), for example, they used various cutting-edge techniques at the time, the semantic differential for data collection, the centroid method for factor analysis, and oblique simple structure rotation. A couple of years later the cutting edge had moved to triadic comparisons for data collection and nonmetric multidimensional scaling (Levelt, Van De Geer, and Plomp (1966)). The analysis

in Levelt, Van De Geer, and Plomp (1966) revealed a parabolic horseshoe structure of the musical intervals.

This inspired John to find a technique to fit quadratic (and higher order, if necessary) structures to scatterplots. If X is a two-dimensional configuration of n points, then form the $n \times 6$ matrix Z with columns $1, x_1, x_2, x_i^2, x_2^2, x_1 x_2$. Now find α with $\alpha' \alpha = 1$ such that $\alpha' Z' Z \alpha$ is as small as possible. This gives the normalized eigenvector corresponding with the smallest eigenvalue of $Z' Z$, or, equivalently, the right singular vector corresponding with the smallest singular value of Z . It is easy to see how this approach generalizes to more dimensions and higher order algebraic surfaces. I remember with how much awe this technique was received by the staff of the Psychological Institute. It probably motivated me in 1966 to develop similar techniques and get my portion of awe.

Levelt, Van De Geer, and Plomp (1966) used the curve fitting technique to draw the best fitting parabola in the two-dimensional scatterplot of musical intervals. In their discussion they suggested that a similar quadratic structure could be found if similarities between political parties were analyzed, because for people in the middle of the left-right scale extreme-left and extreme-right parties would tend to be similar. If the effect of extremity was strong enough, the two extreme might even bend towards each other, leading to an ellipse rather than a parabola. In 1966 John asked student-researcher Dato de Gruijter to figure out if this curving back actually happened, which lead to De Gruijter (1967). Dato collected triadic comparisons between nine Dutch political parties, cumulated over 100 psychology students. The curve fitting technique indeed found best fitting ellipses.

15.5.2 Primal Methods

We follow De Leeuw and Mair (2009) in distinguishing *primal* and *dual* methods. In a primal method the surface we fit is specified in parametric form. The points on the circle, for example, have $(x_{i1}, x_{i2}) = (\sin(\xi_i), \cos(\xi_i))$. Rotational invariance of MDS means we can assume the center of the circle is in the origin. This is the approach of T. F. Cox and Cox (1991). They substitute the parametrix expression for the circle in the formula for stress and minimize over the spherical coordinates ξ_i using gradient methods. They develop a similar method for the sphere in \mathbb{R}^3 . For those who want to go to

higher dimensions we illustrate a parametric representation for \mathbb{R}^4 .

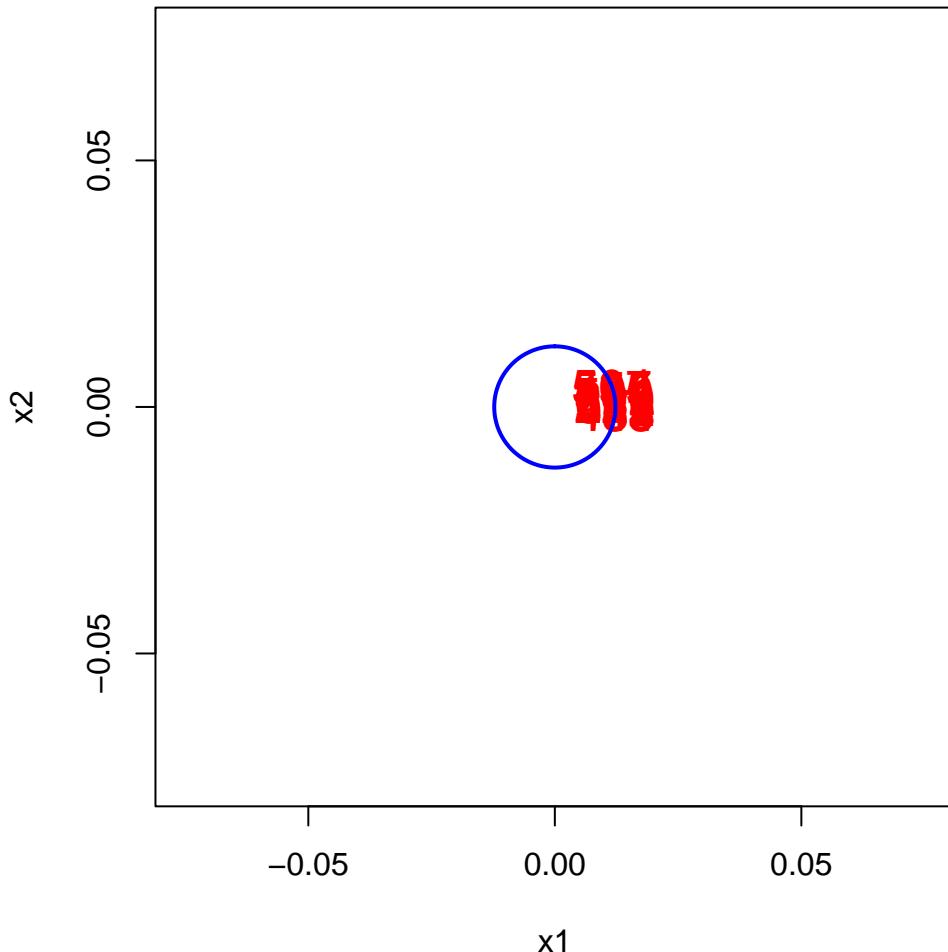
$$(x_{i1}, x_{i2}, x_{i3}, x_{i4}) = (\sin(\xi_i) \cos(\theta_i) \sin(\mu_i), \sin(\xi_i) \cos(\theta_i) \cos(\mu_i), \sin(\xi_i) \sin(\theta_i), \cos(\xi_i)).$$

Spherical coordinates soon get tedious, and De Leeuw and Mair (2009) simply require the distances of all points to the origin to be the same constant. Note that this puts the center of the fitted sphere in the origin, which means that in general the center of the point cloud cannot be taken to be in the origin as well. In smacof we use the

$$\phi(X, \lambda) := \text{tr } (Z - \lambda X)'V(Z - \lambda X)$$

over the radius λ and the configuration X , which is constrained to have $\text{diag } XX' = I$. De Leeuw and Mair (2009) project out λ and minimize $\phi(X, \star) := \min_{\lambda} \phi(X, \lambda)$ over X , using Dinkelbach majorization (Dinkelbach (1967)), a block relaxation that cycles over rows of X . Solving for each p vector of coordinates requires solving a secular equation.

Here we proceed slightly differently. Our first step is to get rid of V using the formulas in 15.2.



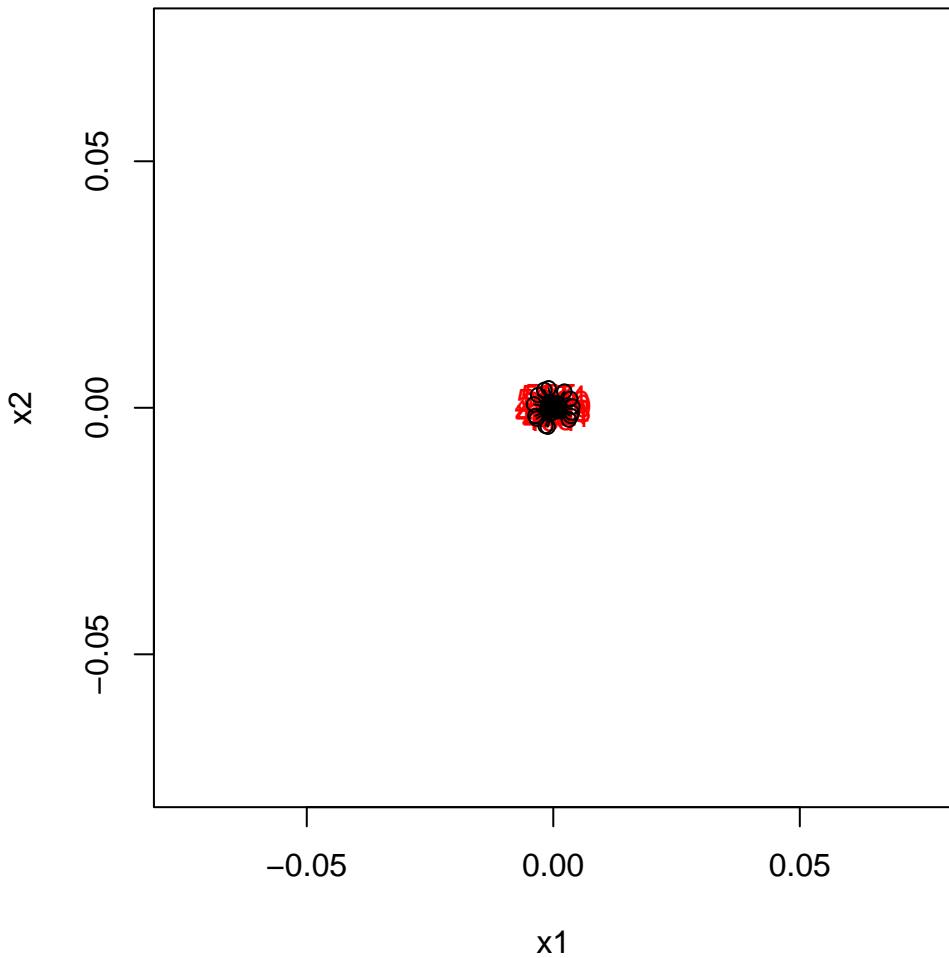
After 2 iterations the primal method converges to a stress value of 0.4654367.
The circle has radius 0.0122974.

15.5.3 Dual Methods

In a dual method we use unrestricted smacof, but we add a penalty to the loss if the configurations do not satisfy the constraints. We use a quadratic penalty, mainly because that fits seamlessly into the smacof approach.

We add one point, the center of the circle, with coordinates x_0 to the configuration, and we require that all n other points have an equal distance from the center. The n dissimilarities $\delta_{0,i}$ are unknown, so we use alternating

least squares and estimate the missing dissimilarities by minimizing stress over them, requiring them to be all equal. All weights $w_{0,i}$ are chosen equal to the penalty parameter ω . The solution for the common $\delta_{0,i}$ is obviously the average of the n distances $d_{0,i}$. In this case it is not necessary to use majorization to transform to unweighted least squares.



15.6 Elliptical MDS

15.6.1 Primal

The smacof projection problem for a p -axial ellipsoid minimizes

$$\phi(Y, \Lambda) := \text{tr} (Z - Y\Lambda)'V(Z - Y\Lambda)$$

with $\text{diag } YY' = I$ and with Λ diagonal and PSD.

ALS

Minimizing ϕ over Λ for fixed Y is easy. For dimension s we have

$$\lambda_s = \frac{y'_s V z_s}{y'_s V y_s}.$$

To minimize ϕ over Y for fixed Λ we use $Z - Y\Lambda = (Z\Lambda^{-1} - Y)\Lambda$ so that

$$\phi(Y, \Lambda) = \text{tr} \Lambda^2 (\tilde{Z} - Y)'V(\tilde{Z} - Y)$$

with $\tilde{Z} = Z\Lambda^{-1}$. We now use a slight modification of the majorization technique in section 15.2. Set $Y = Y_{\text{old}} + (Y - Y_{\text{old}})$. Then

$$\phi(Y, \Lambda) = \text{tr} \Lambda^2 ((\tilde{Z} - Y_{\text{old}}) - (Y - Y_{\text{old}}))'V((\tilde{Z} - Y_{\text{old}}) - (Y - Y_{\text{old}})) = \phi(Y_{\text{old}}, \Lambda) - 2 \text{tr} \Lambda^2 (\tilde{Z} - Y_{\text{old}})'V(Y - Y_{\text{old}})$$

$$\text{tr} \Lambda^2 (Y - Y_{\text{old}})'V(Y - Y_{\text{old}}) \leq \theta \lambda_{\max}^2 \text{tr} (Y - Y_{\text{old}})'(Y - Y_{\text{old}})$$

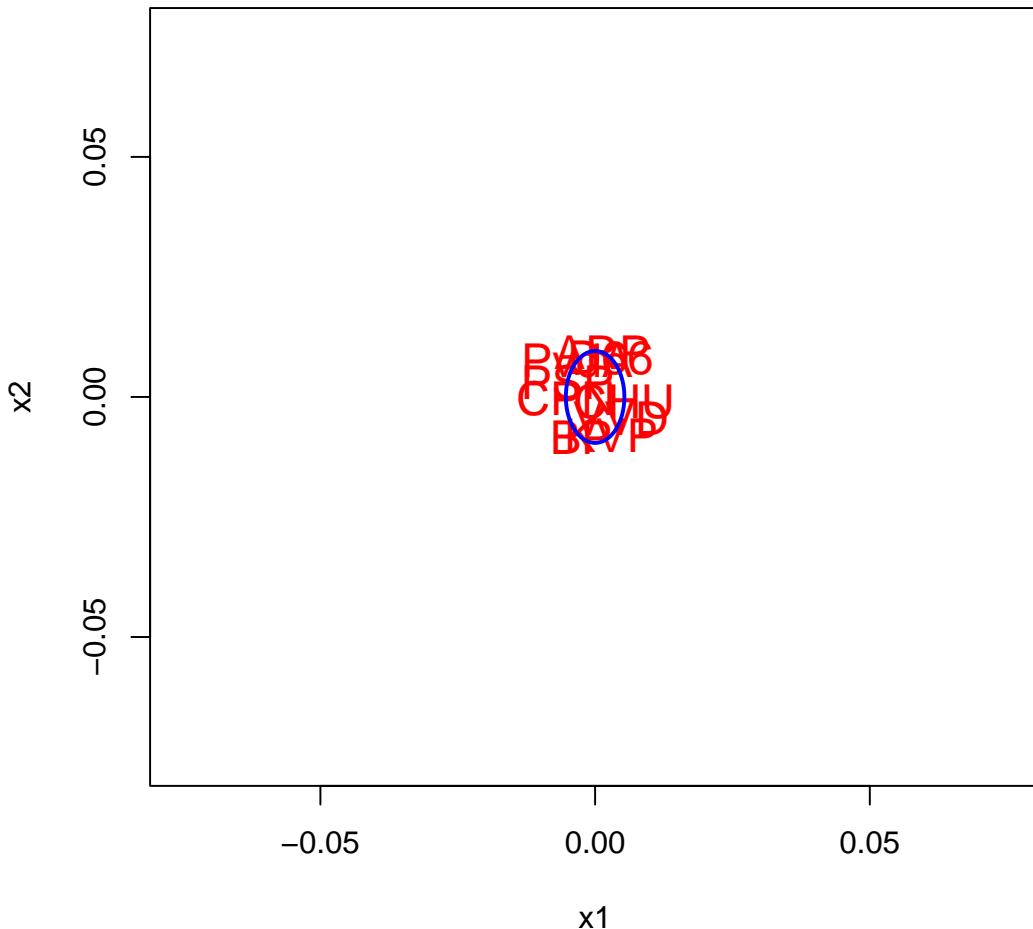
where, as before, θ is the largest eigenvalue of V .

$$\theta \lambda_{\max}^2 (Y - Y_{\text{old}}) = V(\tilde{Z} - Y_{\text{old}})\Lambda^2$$

(abadir_magnus_05, p 283)

Normalize the rows of

$$Y_{\text{old}} + \frac{1}{\theta \lambda_{\max}^2} V(Z\Lambda^{-1} - Y_{\text{old}})\Lambda^2$$



0.0053277, 0.0095667

2

0.415335

15.6.2 Dual

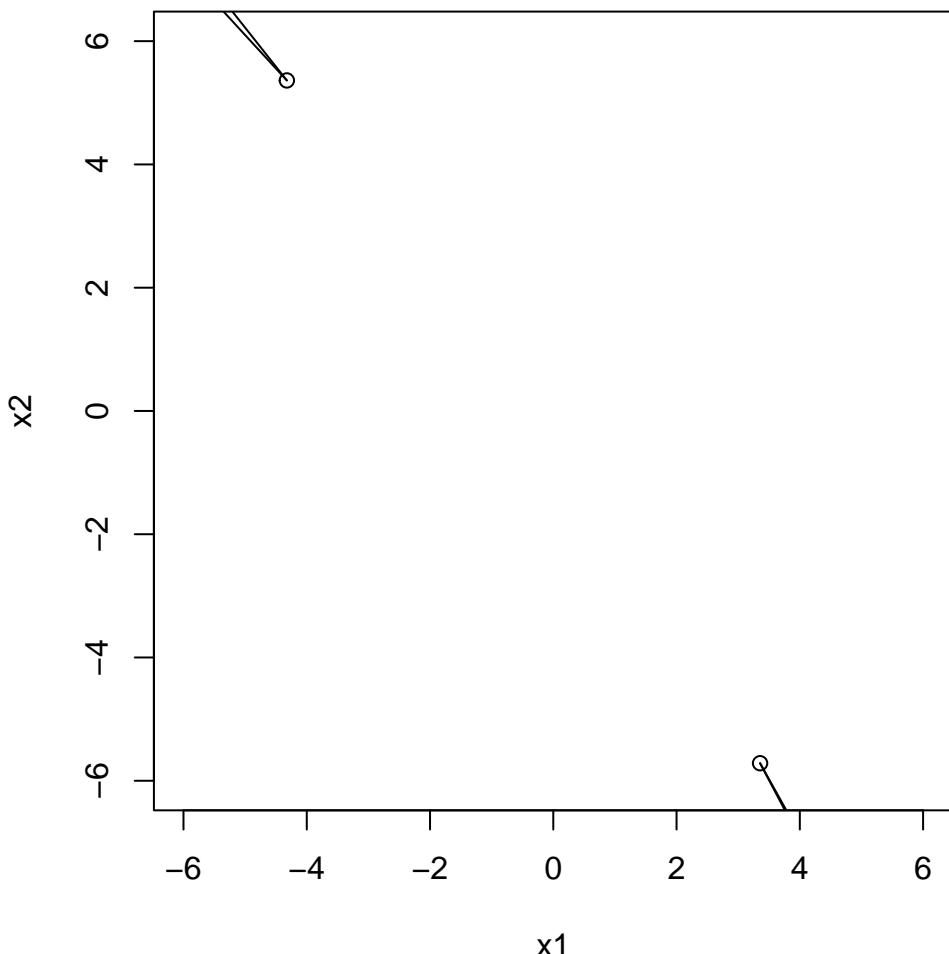
We will only develop a dual method for ellipses in two dimensions, because there is no easy characterization in terms of distances in higher dimensions (that I know of). But in two dimensions the famous pin-and-string construction uses the fact that for all points on the ellipse the sum of the distances to

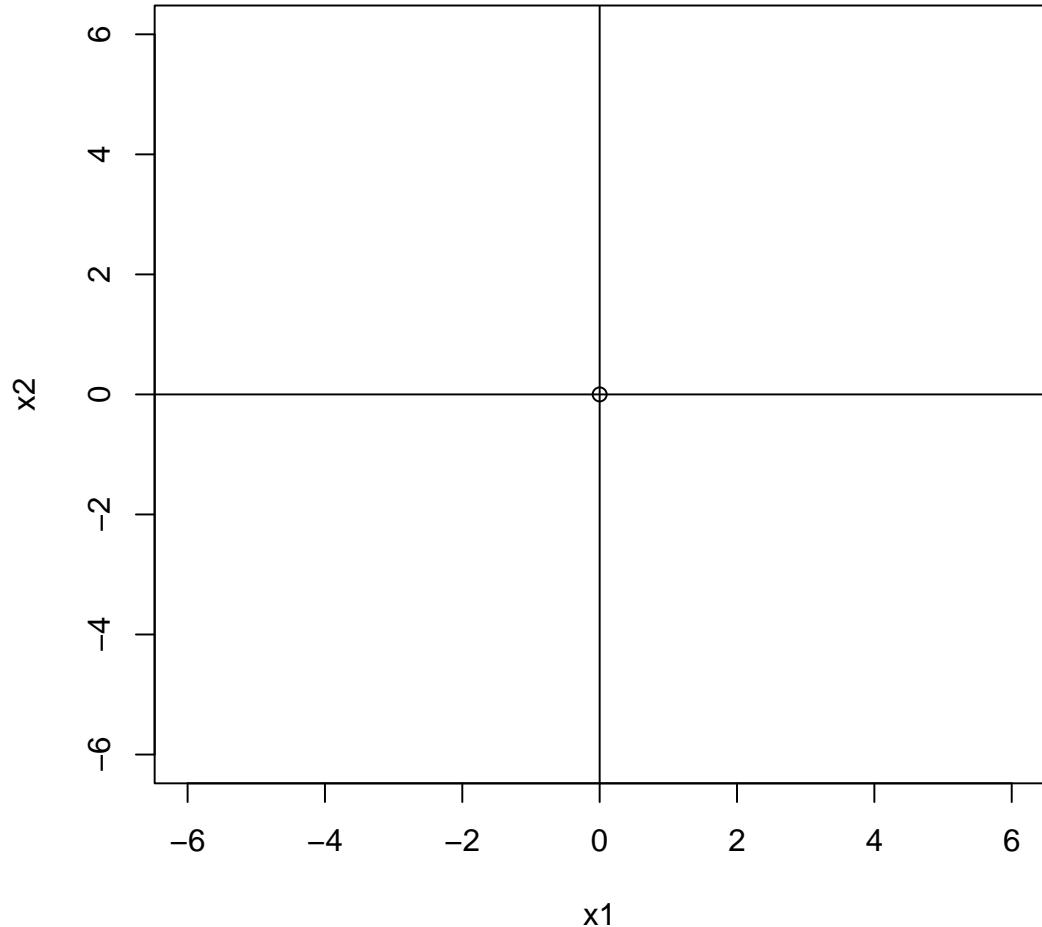
two focal points is constant. Thus our dual method now adds two points to the n points in the configuration, chooses the weights for the $2n$ components of stress to be the penalty parameter w , and finds the $2n$ unknown dissimilarities between the two focal points and the n points on the ellipse to add up to a constant.

This means we have to minimize $\text{tr}(\Delta - D)'(\Delta - D)$ over Δ satisfying $\Delta e = \gamma e$, where for the time being Δ and D are $n \times 2$ submatrices. The Lagrangian is $\text{tr}(\Delta - D)'(\Delta - D) - 2\mu'(\Delta e - \gamma e)$, and thus we must have $\Delta = D + \mu e'$. Taking row sums gives $\gamma e = De + p\mu$ and thus $\mu = \frac{1}{2}(\gamma e - De)$. This implies $\Delta - D = \mu e' = \frac{1}{2}(\gamma E_{np} - DE_{pp})$, and to minimize loss over γ we choose $\gamma = \frac{1}{n}e'_n De_p$. This gives

$$\Delta = DJ + \frac{e'De}{2n}ee'.$$

Thus we take D , transform its n rows to deviations from the mean, and then add the overall mean to all elements.





hyperbola: difference of distances constant $|d((x_i, x_2), (f_1, 0)) - d((x_i, x_2), (g_1, 0))| = c$
 parabola: equal distance from the focus point and the directrix (horizontal axis) $d((x_i, x_2), (f_1, f_2)) = d((x_1, x_2), d(x_1, 0))$

15.7 Distance Bounds

De Leeuw (2017d) De Leeuw (2017c) De Leeuw (2017b)

15.8 Localized MDS

15.9 MDS as MVA

Q methodology

<http://qmethod.org>

Stephenson (1953)

De Leeuw and Meulman (1986) J. J. Meulman (1986) J. J. Meulman (1992)

15.10 Horseshoes

Chapter 16

Asymmetry in MDS

16.1 Conditional Rankings

Two sets (= unfolding), one set (much tighter) solution Young_75

16.2 Confusion Matrices

Choice theory

16.3 The Slide Vector

Chapter 17

Individual Differences

This chapter deals with the situation in which we observe more than one set of dissimilarities. We need an extra index $k = 1, \dots, m$ for Δ_k, W_k , and for X_k . The definition of stress becomes

$$\sigma(X_1, \dots, X_m) := \sum_{k=1}^m \sum_{1 \leq i < j \leq n} w_{ijk} (\delta_{ijk} - d_{ij}(X_k))^2$$

For this particular definition of stress the minimization over X_1, \dots, X_m simply means solving m separate MDS problems, one for each k . Thus it does not bring anything new. In order to make it interesting we have to constrain the X_k in some way or other, preferable one in which the different X_k are linked, so they have something in common and something in which they differ.

MDS with linking constraints on the configurations is known in the psychometric literature as MDS with individual differences. This does not imply that index k necessarily refers to individuals, it can refer to replications, points in time, points of view, experimental conditions, and so on. The essential component is that we have m sets of dissimilarities between the same n objects. In order not to prejudge where the m different sets of dissimilarities come from, we shall refer to them with the neutral term *slices*, just as the dissimilarities are defined on pairs of neutral *objects*.

17.1 Replications

The first constraint that comes to mind is $X_k = X$ for all $k = 1, \dots, m$. Thus the configuration is the same for all slices.

17.2 INDSCAL/PARAFAC

17.3 IDIOSCAL/TUCKALS

Chapter 18

Nominal MDS

So far we have assumed there was some direct information about dissimilarities between objects. The information could be either numerical or ordinal, but we have not talked about nominal or categorical information yet. By nominal information we mean that the n objects are partitioned in K categories, where we usually assume that K is much smaller than n .

Where does MDS come in ? We sort of think that objects in the same category are more similar than those in different categories. But this requirement can be formalized in different ways.

We discuss some of the ways in which we can express the nominal information in terms of distances and apply techniques in the smacof family to computed optimal least squares configurations.

For historical and other reasons this topic is of great interest to me. Some of my first rambling red reports (De Leeuw (1968a), De Leeuw (1969)) were on the analysis of relational or categorical data. The first one discussed mainly the quantification methods of Guttman (1941), the second one explored using the topological notion of separation. My dissertation (De Leeuw (1973a)) systematized that early research. Making it available to a wider audience was the main motivation for starting the Gifi project (Gifi (1990), De Leeuw (2020)).

18.1 Binary Dissimilarities

Suppose we have an equivalence relation \simeq on n objects \mathcal{O} . Let $d_{ij} = 0$ if $i \simeq j$ and $d_{ij} = 1$ if $i \not\simeq j$. Suppose the quotient set \mathcal{O}/\simeq has K equivalence classes. Thus the objects can be ordered in such a way that Δ has K zero matrices with size $n_1 \times n_1, \dots, n_K \times n_K$ in the diagonal blocks and ones everywhere else. The n_k are the sizes of the equivalence classes, and they add up to n .

[x] canonical projection

18.2 Indicator matrix

Q technique

```
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 2 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 3 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 4 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 5 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 6 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 7 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 8 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 9 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 10 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
## 11 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
## 12 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
## 13 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
## 14 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
## 15 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
## 16 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
## 17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
## 18 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
## 19 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
## 20 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
```

Table 18.1: Example Indicator Matrix

	A	B	C
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0
10	1	0	0
11	0	1	0
12	0	1	0
13	0	1	0
14	0	1	0
15	0	1	0
16	0	0	1
17	0	0	1
18	0	0	1
19	0	0	1
20	0	0	1

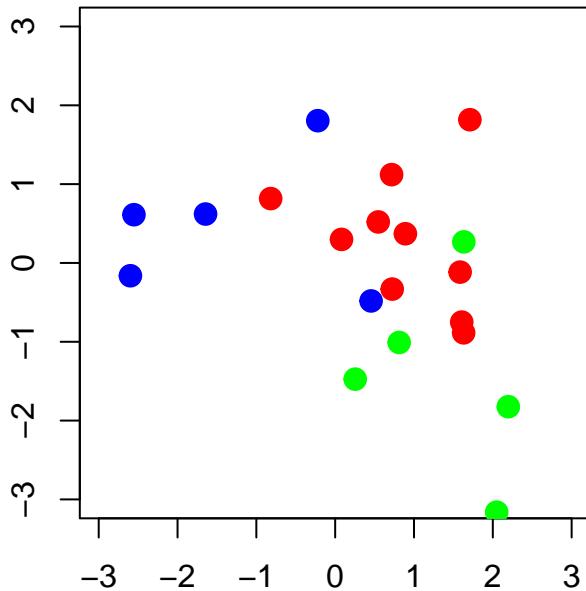


Figure 18.1: Example Object Scores

18.3 Unfolding Indicator Matrices

all within category distances smaller than all between category distances (for each variable separately) as in MCA: joint persons and categories, primary approach to ties

These order constraints are rather strict. They do not only imply that the category clouds are separated, they also mean the clouds must be small and rather far apart.

Think of the situation where the two categories are balls in \mathbb{R}^p with centers x and y and radius r . The largest within-category distance is $2r$. The smallest between-category distance is $\max(0, d(x, y) - 2r)$. Thus all within-category distances are all smaller than all between-category distances if and only if $d(x, y) \geq 4r$.

We can make the requirements less strict by

For all k

$$\max_{i \in I_k, j \in I_k} d(x_i, x_j) \leq \min_{i \in I_k, j \in I \setminus I_k} d(x_i, x_j).$$

$$\max(kk) \leq \min(k\bar{k})$$

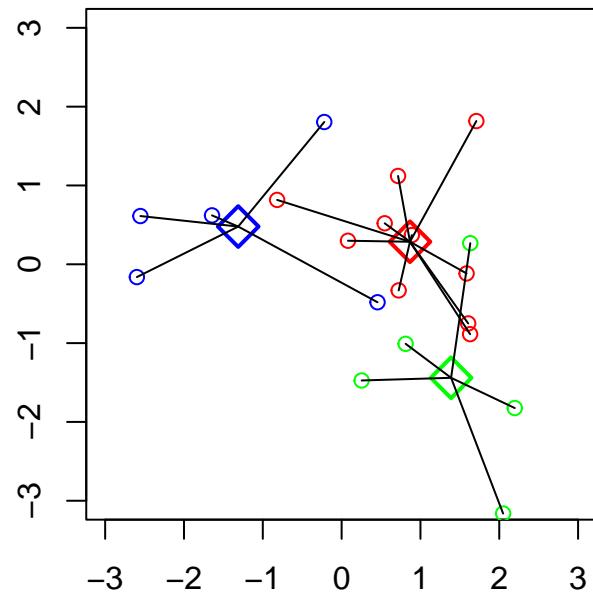


Figure 18.2: Distance Based MCA

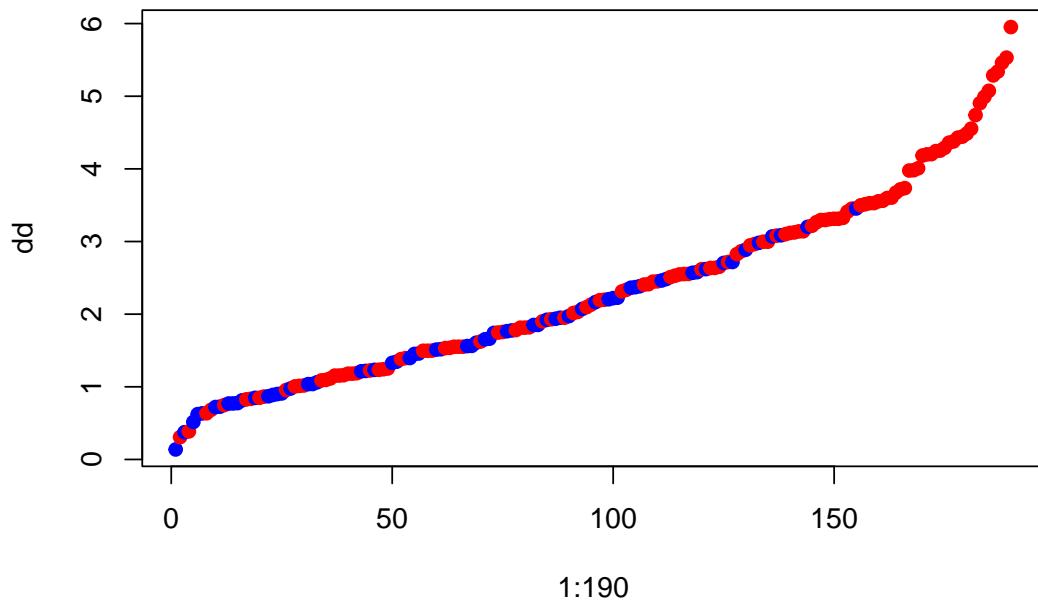


Figure 18.3: Within and Between Distances

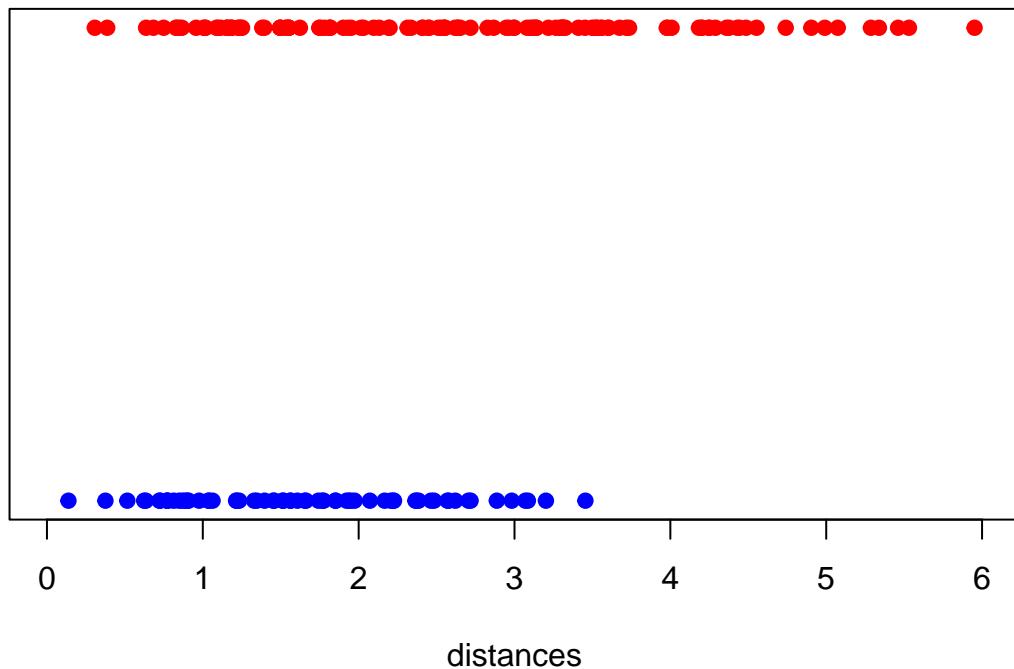


Figure 18.4: Within and Between Distances

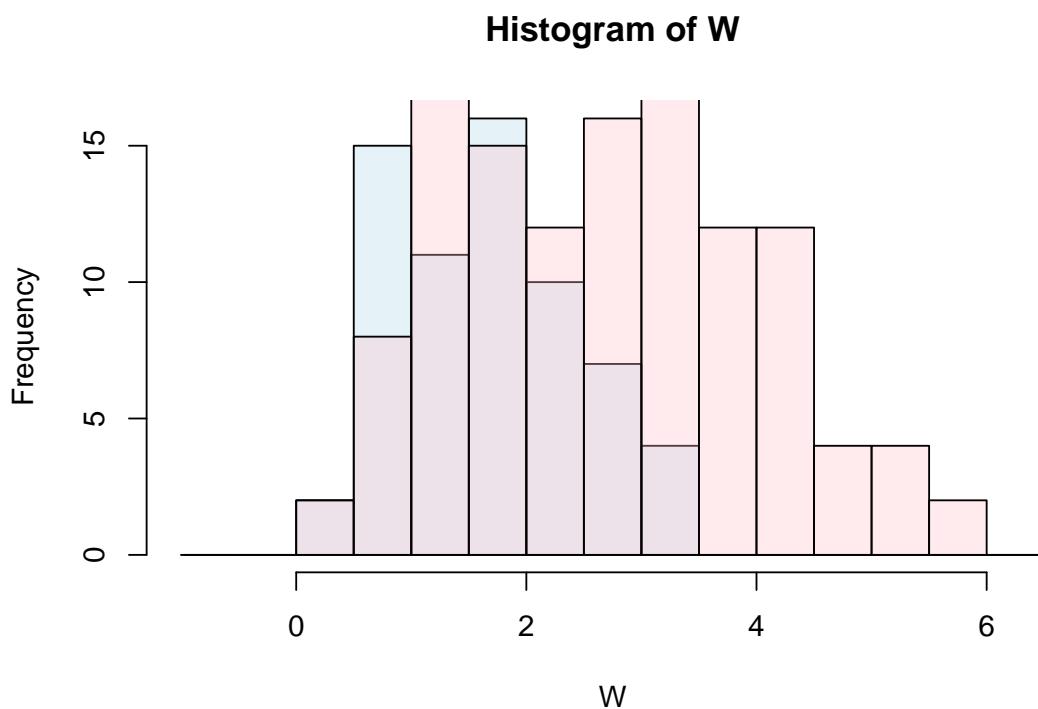


Figure 18.5: Within and Between Distances

$$\max_k \max(kk) \leq \min_{i \neq j} \min(ij)$$

the within category distances of category 1 are less than the smallest between-category distance

18.4 Linear Separation

line perpendicular to line connecting category points separates categories all closer to their star center than to other star centers: primary monotone regression over all rows of g

suppose the star centers are y and z . The plane is $(x - \frac{1}{2}(y + z))'(y - z) = 0$
If u is in the y category we must have $(u - \frac{1}{2}(y + z))'(y - z) \geq 0$

Just in terms of distances

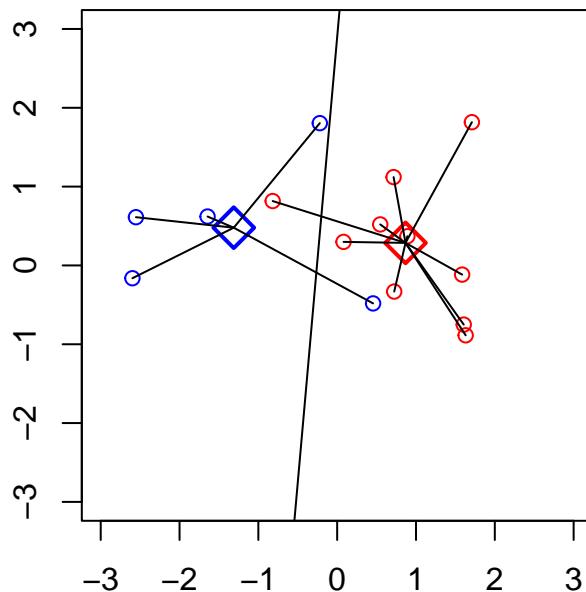


Figure 18.6: Distance Based MCA

What if the star centers are on a straight line

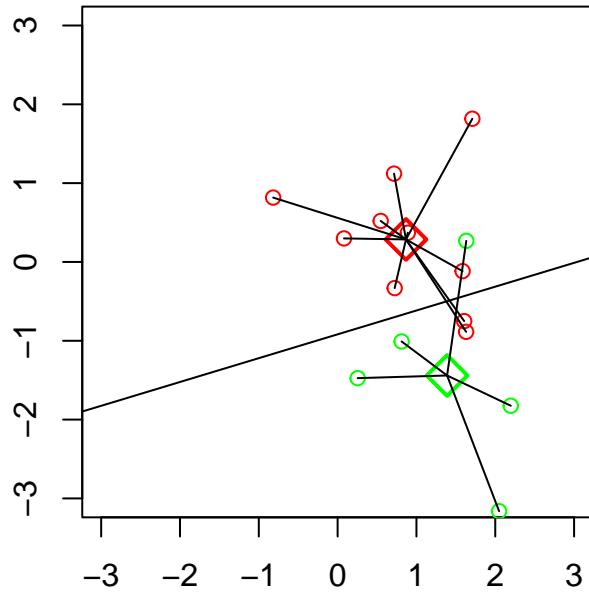


Figure 18.7: Distance Based MCA

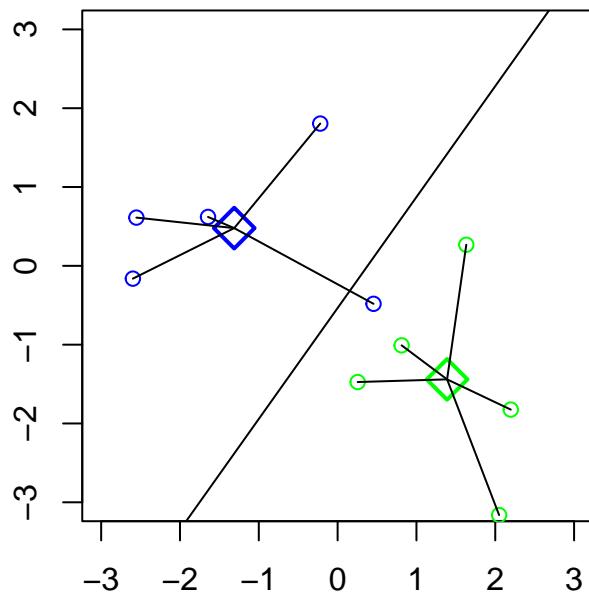


Figure 18.8: Distance Based MCA

18.5 Circular Separation

monotone regression on each column of \mathbf{g}

k within balls must be disjoint \Rightarrow they can be separated by straight lines

18.6 Multidimensional Scalogram Analysis

Guttman's MSA: Inner points, outer points

Lingoes (1968b) Lingoes (1968a) Guttman (1967)

Chapter 19

Sstress and strain

19.1 sstress

Takane, Young, and De Leeuw (1977) gave σ_2 the name *sstress*. Thus sstress is defined as

$$\sigma_2(X) := \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij}^2 - d_{ij}^2(X))^2$$

On the space of configurations sstress is mathematically much better behaved as stress. It is a non-negative multivariate polynomial of degree four, actually a sum-of-squares or SOS polynomial (ref).

It is everywhere infinitely many times differentiable everywhere and, in principle at least, we can compute all real-valued configurations where the derivatives of sstress vanish by algebraic methods. That includes all local minima, and thus also the global minimum. Unfortunately in almost all MDS applications the number of variables is too large to apply the usual algebraic methods.

nonnegative polynomials of degree four – general algebra

19.1.1 sstress and stress

Clearly configurations with small stress will tend to have small sstress, and vice versa.

weighting residuals – fitting large dissimilarities

$$\sum d_{ij}^2(\delta_{ij}^2 - d_{ij}^2) = 0$$

We can write sstress as

$$\sigma_2(X) = \sum_{1 \leq i < j \leq n} \sum w_{ij} (\delta_{ij} + d_{ij}(X))^2 (\delta_{ij} - d_{ij}(X))^2. \quad (19.1)$$

Thus if the $d_{ij}(X)$ provide a good fit to the δ_{ij} we have the approximation

$$\sigma_2(X) \approx 4 \sum_{1 \leq i < j \leq n} \sum w_{ij} \delta_{ij}^2 (\delta_{ij} - d_{ij}(X))^2 \quad (19.2)$$

Since $\mathcal{D}f(\delta) = 2\delta$ in this case, (19.2) also follows from (20.3).

Now

$$\frac{\sigma_2(X)}{\sigma(X)} = \frac{\sum \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij} + d_{ij}(X))^2 (\delta_{ij} - d_{ij}(X))^2}{\sum \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij} - d_{ij}(X))^2}. \quad (19.3)$$

This is a weighted average of the quantities $(\delta_{ij} + d_{ij}(X))^2$, and thus

$$4\{\min(\delta_{ij} + d_{ij}(X))\}^2 \sigma(X) \leq \sigma_2(X) \leq 4\{\max(\delta_{ij} + d_{ij}(X))\}^2 \sigma(X). \quad (19.4)$$

19.1.2 Decomposition

$$\begin{aligned} \sigma_2(X) &= 1 - 2\rho_2(X) + \eta_2^2(X) \\ \sigma_2(\alpha X) &= 1 - 2\alpha^2 \rho_2(X) + \alpha^4 \eta_2^2(X) \end{aligned}$$

At a minimum $\rho_2(X) = \eta_2^2(X)$ and thus $\sigma(X) = 1 - \eta_2^2(X)$, which implies $\eta_2(X) \leq 1$.

Thus minimizing σ_2 means maximizing ρ_2 over $\eta_2(X) \leq 1$, which is the same thing as minimizing η_2 over $\rho_2(X) \geq 1$. Both are reverse convex problems.

$$\eta_2(X) = \sqrt{\sum \sum w_{ij} d_{ij}^4(X)} \geq \frac{1}{\sqrt{\sum \sum w_{ij} d_{ij}^4(Y)}} \sum \sum w_{ij} d_{ij}^2(Y) d_{ij}^2(X)$$

Thus maximizing ρ_2 with $\rho_2(X) = \text{tr } X' B_0 X$ over $\text{tr } X' B_2(Y) X \leq 1$ for all Y .

19.1.3 Full-dimensional sstress

Theorem: The set of squared Euclidean distance matrices between n points $\mathfrak{D} := \{D \mid D = D^{(2)}(X)\}$ is a closed convex cone.

Proof. It suffices to observe that $\alpha D^{(2)}(X) = D^{(2)}(\sqrt{(\alpha)}X)$ and $D^{(2)}(X) + D^{(2)}(Y) = D^{(2)}(X \mid Y)$. Alternatively, \mathfrak{D} is the intersection of two convex cones, and thus convex. The first cone are the hollow non-negative symmetric matrices and the second cone are the symmetric matrices D for which $-JDJ \gtrsim 0$. \square

Corollary:

$$\min_{D \in \mathfrak{D}} \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij}^2 - d_{ij})^2$$

is a convex problem with a unique minimum.

$$D \in \mathfrak{D}, \Delta - D \in \mathfrak{D}^o \text{tr } W \times D(\Delta - D) = 0$$

Since $d_{ij}^2 = c_{ii} + c_{jj} - 2c_{ij}$ it follows that σ_2 is a convex quadratic in C and that minimizing σ_2 over $C \gtrsim 0$ is a convex problem, just as minimizing σ_1 over $C \gtrsim 0$ is.

$$\begin{aligned} & \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij}^2 - \text{tr } A_{ij} C)^2 \\ \mathcal{D}\sigma_2(C) = & -2 \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij}^2 - \text{tr } A_{ij} C) A_{ij} \end{aligned}$$

$$B_2(C) := \sum_{1 \leq i < j \leq n} w_{ij} d_{ij}^2(C) A_{ij}$$

Weights !! Polar cone !

$$\begin{aligned} C &\gtrsim 0, \\ V_2 - B_2(C) &\gtrsim 0, \\ \text{tr } C(V_2 - B_2(C)) &= 0. \end{aligned} \tag{19.5}$$

gower2 rank

19.1.4 Minimizing sstress

19.1.4.1 ALSCAL

The first published paper on sstress minimization was Takane, Young, and De Leeuw (1977). There are some historical precursors, but they are mostly in internal memos, and they usually did not come with software. The ALSCAL program (F. W. Young, Takane, and Lewyckyj (1978b)) was widely distributed through SPSS and SAS and is still used regularly in various areas of research.

In this section of the book we will discuss basic ALSCAL, i.e. the sstress version of basic MDS scaling. As usual, we generalize to weighted least squares, but for now we ignore the individual differences and the non-metric parts.

It is also worth noting that in the original version of ALSCAL, from Takane, Young, and De Leeuw (1977), the configuration is fitted by an alternating least squares algorithm that changes all p coordinates of each point simultaneously, and then cycles through the points. The minimization over coordinates is done with a safeguarded Newton-Raphson method. At the time I forcefully objected to this. Sstress is a p -dimensional quartic, and because p is usually small finding the global minimum with algebraic methods is at least conceivable. But in terms of simplicity, it is much better to change a single coordinate at the time, meaning that one cycle consists of finding the global minimum of np univariate quadratics. There is some acknowledgement of this

in section 5 of Takane, Young, and De Leeuw (1977), but the paper is quite verbose and it can easily be overlooked. My understanding, based on F. W. Young, Takane, and Lewyckyj (1978a), is that later versions of ALSCAL did indeed adopt one-dimensional cyclic coordinate descent (CCD). And this is what we will discuss here.

First note that sstress can be decomposed in the same way as stress. We have

$$\sigma_2(X) = 1 - 2\rho_2(X) + \eta_2^2(X) \quad (19.6)$$

with

$$\begin{aligned} \rho_2(X) &:= \sum_{1 \leq i < j \leq n} w_{ij} \delta_{ij}^2 d_{ij}^2(X), \\ \eta_2^2(X) &:= \sum_{1 \leq i < j \leq n} w_{ij} d_{ij}^4(X). \end{aligned} \quad (19.7)$$

Both ρ_2 and η_2^2 are convex, ρ_2 is quadratic and η_2^2 is quartic.

In CCD we replace x_{ks} by $\tilde{x}_{ks} := x_{ks} + \epsilon$. Or, in matrices, $\tilde{X} := X + \epsilon e_k e_s'$. Only the $d_{ij}^2(\tilde{X})$ with $i = k$ or $j = k$ differ from the corresponding $d_{ij}(X)$. All these $d_{ik}^2(\tilde{X})$ are now just a function of ϵ and thus we write

$$d_{ik}^2(\epsilon) := d_{ik}^2(\tilde{X}) = d_{ik}^2(X) - 2\epsilon u_i + \epsilon^2. \quad (19.8)$$

where $u_i := x_{is} - x_{ks}$. Also

$$d_{ik}^4(\epsilon) = d_{ij}^4(X) - 4\epsilon u_i d_{ik}^2(X) + 2\epsilon^2(d_{ik}^2(X) + 2u_i^2) - 4\epsilon^3 u_i + \epsilon^4. \quad (19.9)$$

Combining (19.8) and (19.9) with (19.6) gives

$$\rho_2(\epsilon) = \rho_2(0) - 2\epsilon \sum_{i=1}^n w_{ik} \delta_{ik}^2 u_i + \epsilon^2 \sum_{i=1}^n w_{ik} \delta_{ik}^2 \quad (19.10)$$

and

$$\eta_2^2(\epsilon) = \eta_2^2(0) - 4\epsilon \sum_{i=1}^n w_{ik} d_{ik}^2(X) u_i + 2\epsilon^2 \sum_{i=1}^n w_{ik} (d_{ik}^2(X) + 2u_i^2) - 4\epsilon^3 \sum_{i=1}^n w_{ik} u_i + \epsilon^4 \sum_{i=1}^n w_{ik} \quad (19.11)$$

and finally

$$\sigma_2(\epsilon) = \sigma_2(0) - 4\epsilon \sum_{i=1}^n w_{ik} (d_{ik}^2(X) - \delta_{ik}^2) u_i + 2\epsilon^2 \sum_{i=1}^n w_{ik} ((d_{ik}^2(X) - \delta_{ik}^2) + 2u_i^2) - 4\epsilon^3 \sum_{i=1}^n w_{ik} u_i + \epsilon^4 \sum_{i=1}^n w_{ik} \quad (19.12)$$

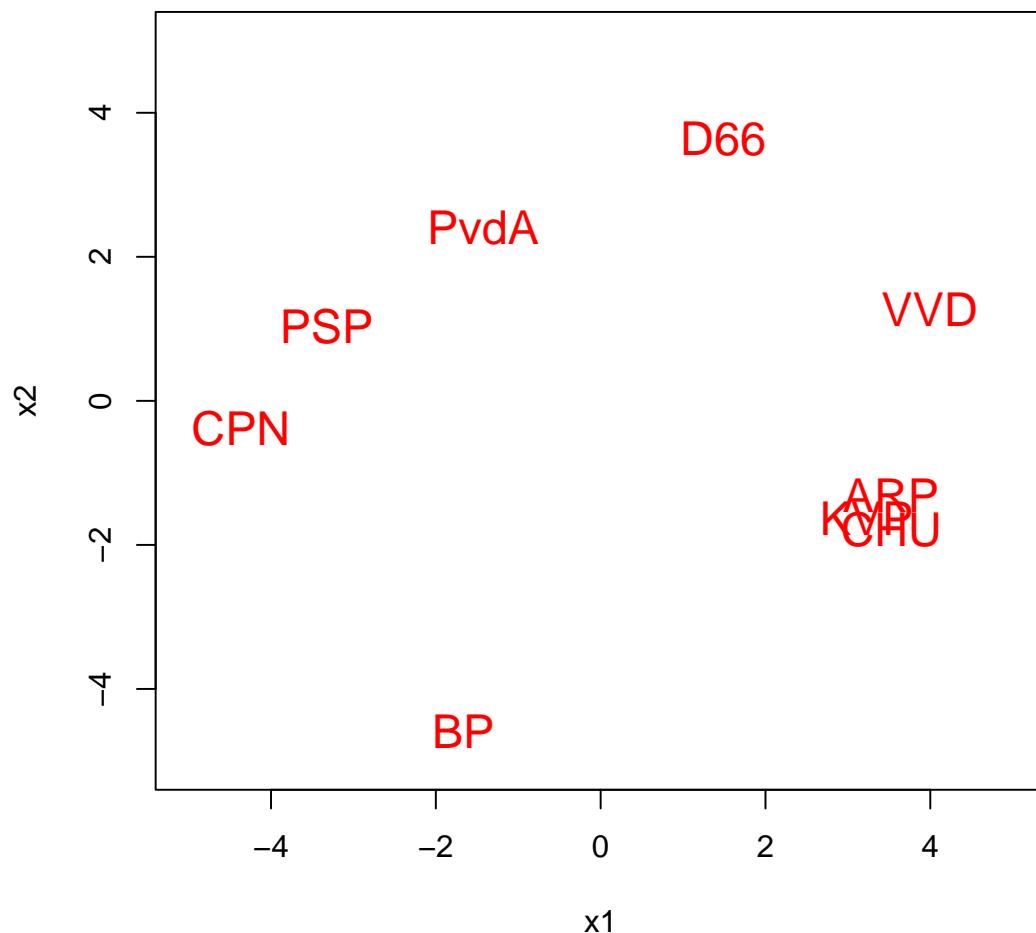
Convex, DC, derivative

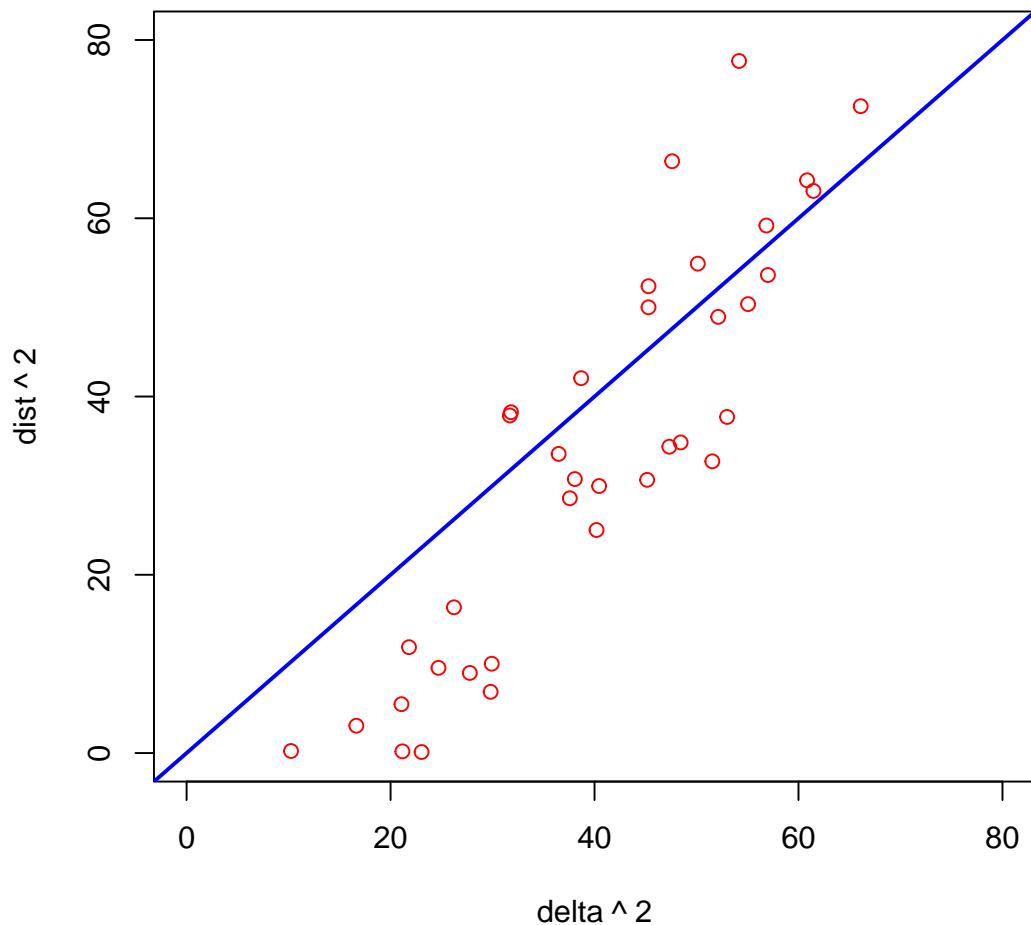
Jeffrey

Code

Example

Query: is σ_2 a convex quartic in ϵ ? Does $\mathcal{D}\sigma_2(\epsilon) = 0$ always have a single root.





1.7957662×10^4 after 55 iterations

Try global minimum over points (p-dimensional quartic)

19.1.4.2 Majorization

De Leeuw (1975b)

De Leeuw, Groenen, and Pietersz (2016)

takane

brown

Functions of Squared Distances

$$\begin{aligned}
f(X) &= F(D^2(X)) \\
\mathcal{D}f(X) &= 2 \sum \sum \mathcal{D}_{ij} F(D^2(X)) A_{ij} X \\
f(C) &= F(D^2(C)) \\
\mathcal{D}f(C) &= \sum \sum \mathcal{D}_{ij} F(D^2(C)) A_{ij} \\
\mathcal{D}^2 f(C) &= \sum \sum \mathcal{D}_{ij,kl} F(D^2(C)) A_{ij}
\end{aligned}$$

19.1.4.3 SOS

Normal Fourth degree tensor

$$\begin{aligned}
&\sum \sum w_{ij} (\delta_{ij} - x' A_{ij} x)^2 \\
&\sum_k \sum_l \left\{ \sum \sum w_{ij} \delta_{ij} A_{ij} \right\} x_k x_l \\
&\sum_k \sum_l \sum_p \sum_q \left\{ \sum \sum w_{ij} \delta_{ij} \{A_{ij} \otimes A_{ij}\} \right\}_{klpq} x_k x_l x_p x_q
\end{aligned}$$

19.1.4.4 Duality

!! David Gao

19.1.4.5 ALS Unfolding

In the heady days around 1968, when the Department of Data Theory was founded in Leiden, the focus was very much on unfolding. Coombs had just visited and the non-metric revolution was starting up. Alternating least squares was in the air. I was supposed to work on a program for metric unfolding, starting from the ideas of Ross and Cliff (1964) and the machinery provoked by Torgerson's classical scaling.

The idea, mainly due to John van de Geer, was to complete the $n \times m$ matrix of off-diagonal dissimilarities to a symmetric matrix of order $n + m$, starting

with initial estimates of the distances in the two diagonal blocks. Then apply Torgerson, and use the results to improve the estimates of the distances in the diagonal blocks, then use Torgerson again, and so on. Alternating least squares with imputation of the diagonal blocks. Multidimensional scaling of $n + m$ objects, with zero weights for the diagonal blocks. But it only worked to a certain point. After decreasing stress for a while and approaching convergence the loss started to increase. We were deflated and gave up the approach, without really being able to understand about why it did not work (De Leeuw (1968c)).

In hindsight, it is clear what was wrong. We imputed the diagonal blocks minimizing stress, and then adjusted the configuration using strain. Two different loss functions, which obviously violated the basic idea of alternating least squares and the guaranteed convergence of either of the two loss functions.

What is worth preserving from this approach is the initial estimate for the diagonal blocks, again due to John van de Geer. It cleverly uses the two triangle inequalities, assuming the dissimilarities are really distances. For $1 \leq i < j \leq n$

$$\delta_{ij} = \frac{1}{2} \left\{ \min_{k=1}^m (\delta_{ik} + \delta_{jk}) + \max_{k=1}^m |\delta_{ik} - \delta_{jk}| \right\},$$

and for $n + 1 \leq k < l \leq n + m$

$$\delta_{kl} = \frac{1}{2} \left\{ \min_{v=1}^n (\delta_{vk} + \delta_{vl}) + \max_{k=1}^m |\delta_{vk} - \delta_{vl}| \right\}.$$

Greenacre and Browne (1986)

augmentation

19.1.5 Bounds for sstress

19.2 strain

Classical scaling as formulated by Torgerson (1958) computes the dominant non-negative eigenvalues, with corresponding eigenvectors, of the Torgerson

transform of the squared dissimilarities. This is usually presented as an “ignore-errors” technique. It is clear what it does in the case of perfect fit of distances to dissimilarities, it is not so obvious how it measures approximation errors in the case of imperfect fit. Or, to put it differently, MDS lore has it that in classical scaling loss is defined on the scalar products, which are a transformation of the dissimilarity data, and not on the dissimilarities themselves. This is presented as somehow being a disadvantage (see, for example, Takane, Young, and De Leeuw (1977), Browne (1987)).

If we agree to use weights, then *strain* is defined straightforwardly as

$$\sigma_\tau(X) := \sum_{i=1}^n \sum_{j=1}^n w_{ij} (\tau_{ij}(\Delta^{(2)}) - x_i' x_j)^2. \quad (19.13)$$

Note that this summation includes the diagonal elements, so in general we cannot expect W to be hollow. In fact, @eq:straindef adds diagonal elements only once, while the off-diagonal elements are added twice. This observation is the basis of the excellent paper by Bailey and Gower (1990).

Because of the weights, minimizing strain in this form does not lead to an eigenvalue problem, unless there is a non-negative vector u such that $w_{ij} = u_i u_j$. In that case

$$\sigma_\tau(X) := \text{tr } (U\tau(\Delta^{(2)})U - UX X' U)^2, \quad (19.14)$$

where $U = \text{diag}(u)$, and we can find UX , and thus X , by eigen decomposition of $U\tau_{ij}(\Delta^{(2)})U$.

We can use our general results on unweighting, as in section @ref{minunweight}, to get rid of the weights, but this leads to a sequence of eigenvalue problems. Nevertheless, if the weights are important, this is an option.

19.2.1 Unweighted

For column-centered configurations

$$J(\Delta^{(2)} - D^{(2)}(X))J = -2(\tau(\Delta^{(2)}) - XX')$$

Thus if all weights are equal to one then

$$4\sigma_\tau(X) = \text{tr } J(\Delta^{(2)} - D^{(2)}(X))J(\Delta^{(2)} - D^{(2)}(X)),$$

which shows that strain is a matrix-weighted version of sstress. It also shows (De Leeuw and Heiser (1982), theorem 21) that

$$\sigma_\tau(X) \leq \frac{1}{4}\sigma_2(X).$$

19.2.2 Using Additivity

$$\sigma_\tau(X) = \min_{\alpha} \sum_{1 \leq i < j \leq n} \sum w_{ij} (\delta_{ij}^2 - (\alpha_i + \alpha_j) - 2x'_i x_j))^2$$

$$\sigma_2(X) = \min_{\alpha \geq 0} \min_{\text{diag } X X' = I} \sum_{1 \leq i < j \leq n} \sum w_{ij} (\delta_{ij}^2 - (\alpha_i^2 + \alpha_j^2 - 2\alpha_i \alpha_j x'_i x_j))^2$$

projection

If there are no weights, or if we unweight the weighted loss function

Chapter 20

fstress and rstress

20.1 fstress

Fstress is a straightforward generalization of stress. Suppose f is any non-decreasing real-valued function, and define

$$\sigma_f(X) := \sum_{1 \leq i < j \leq n} \sum w_{ij} (f(\delta_{ij}) - f(d_{ij}(X)))^2 \quad (20.1)$$

We discuss various specific examples in this chapter, such as the square and the logarithm, but let's first mention some general results.

20.1.1 Use of Weights

Suppose the $d_{ij}(X)$ are close to the δ_{ij} , so that we have a good fit and a low stress. Then the approximation

$$f(d_{ij}(X)) \approx f(\delta_{ij}) + \mathcal{D}f(\delta_{ij})(d_{ij}(X) - \delta_{ij}) \quad (20.2)$$

will be close. Thus

$$\sigma_f(X) \approx \sum_{1 \leq i < j \leq n} \sum w_{ij} (\mathcal{D}f(\delta_{ij}))^2 (\delta_{ij} - d_{ij}(X))^2. \quad (20.3)$$

Thus we can approximately minimize fstress by minimizing stress with weights $w_{ij}(\mathcal{D}f(\delta_{ij}))^2$. If the fit is good, we can expect to be close. If the fit is perfect, the approximation is perfect too. Note that we do not assume that f is increasing, i.e. that $f' \geq 0$.

20.1.2 Convexity

$$f(g(\lambda x + (1 - \lambda)y)) \leq f(\lambda g(x) + (1 - \lambda)g(y)) \leq \lambda f(g(x)) + (1 - \lambda)f(g(y))$$

Thus if g is convex (for instance distance) and f is convex and increasing then $f \circ g$ is convex (and thus stress is DC). Unfortunately a concave f is more interesting.

20.2 rStress

$$\sigma_r(X) := \sum_{1 \leq j < i \leq n} w_{ij}(\delta_{ij}^r - d_{ij}^r(X))^2. \quad (20.4)$$

In definition (20.4) we approximate the r -th power of the dissimilarities by the r -th power of the distances. Alternatively, we could have defined

$$\sigma_r(X) := \sum_{1 \leq j < i \leq n} w_{ij}(\delta_{ij} - d_{ij}^r(X))^2 \quad (20.5)$$

In definition (20.4) we are still approximating the dissimilarities by the distances, as in basic MDS, but we are defining errors of approximation as the differences between the r -th powers. I am not sure which of the two formulations is the more natural one. I am sure, however, that for metric and known dissimilarities the two formulations are basically the same, because we can just define dissimilarities in (20.5) as the r -th power of the ones in (20.4). And in the ordinal case the two formulations are the same as well, because the rank orders of the unpowered and powered dissimilarities are the same.

20.2.1 Using Weights

$$d_{ij}^r(X) - \delta_{ij}^r = (d_{ij}^r(X) + \delta_{ij}^r)(d_{ij}^r(X) - \delta_{ij}^r)$$

If $\delta_{ij} \approx d_{ij}(X)$ then

If r is a power of 2 ==>

20.2.2 Minimizing rstress

rstress, qstress, power stress

Groenen and De Leeuw (2010) De Leeuw (2014b) De Leeuw, Groenen, and Mair (2016b) De Leeuw, Groenen, and Mair (2016d) De Leeuw, Groenen, and Mair (2016a)

20.3 mstress

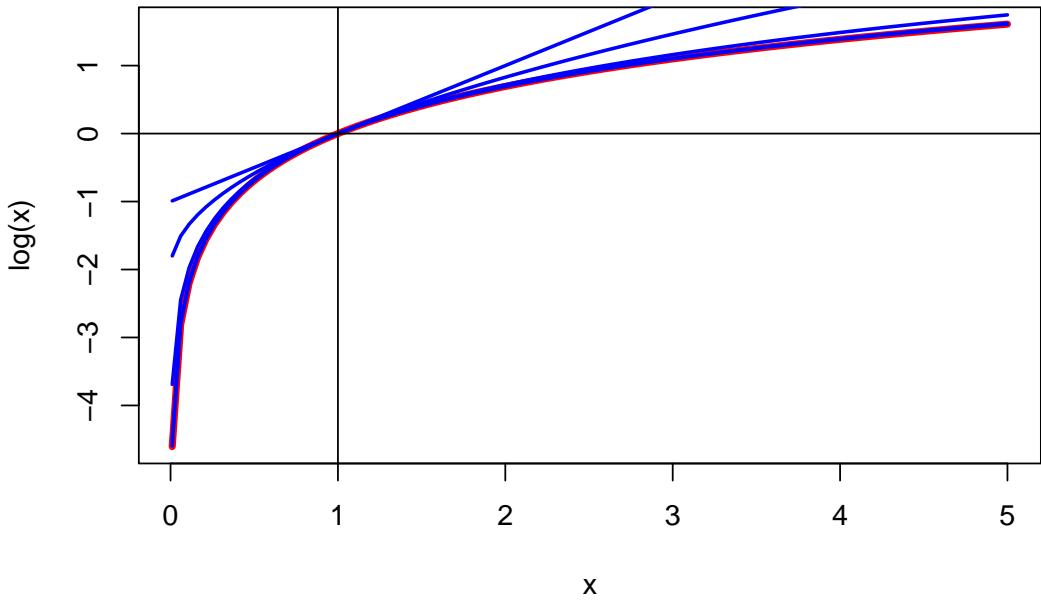
The loss function used by Ramsay (1977) in his MULTISCAL program for MDS can be written as

$$\sigma_0(X) := \sum_{1 \leq i < j \leq n} w_{ij} (\log \delta_{ij} - \log(d_{ij}(X)))^2. \quad (20.6)$$

To justify the notation σ_0 we define f_r , for all $x > 0$ and $r < 1$, by $f_r(x) := r^{-1} \frac{x^r - 1}{r}$. f_r is concave for all r , it majorizes the log because $f_r(x) \geq \log(x)$ for all x , with equality iff $x = 1$, and

$$\lim_{r \rightarrow 0} \frac{x^r - 1}{r} = \log x. \quad (20.7)$$

We have drawn the logarithm, in red, and f_r for r equal to 0.001, 0.01, 0.1, 0.5, 1 over the interval [.01, 5] in the figure that follows.



For $r = .001$ and $r = .01$ the logarithm and f_r are practically indistinguishable, and even for $r = .1$ we have an approximation which is probably good enough (in the given range) for most practical purposes.

Using the approximation of the logarithm with small r gives

$$\begin{aligned}\sigma_0(X) &\approx \sum_{1 \leq i < j \leq n} w_{ij} \left(\delta_{ij} - \frac{d_{ij}^r(X) - 1}{r} \right)^2 \\ &= r^{-2} \sum_{1 \leq i < j \leq w_{ij}} ((r\delta_{ij} + 1) - d_{ij}^r(X))^2.\end{aligned}\tag{20.8}$$

If r is really small both $r\delta_{ij} + 1$ and $d_{ij}^r(X)$ will be very close to one, which will make minimization of the approximation difficult. A simple suggestion is to start with the SMACOF solution for $r = 1$, then use that solution for $r = 1$ as a starting point for $r = \frac{1}{2}$, and so on.

Alternative ?

$$\log d_{ij}(X) - \log \delta_{ij} \leq \frac{\left\{ \frac{d_{ij}(X)}{\delta_{ij}} \right\}^r - 1}{r} = \frac{d_{ij}^r(X) - \delta_{ij}^r}{r \delta_{ij}^{r-1}}$$

$$\log d_{ij}(X) - \log d_{ij}(Y) \leq \frac{\left\{ \frac{d_{ij}(X)}{d_{ij}(Y)} \right\}^r - 1}{r} = \frac{d_{ij}^r(X) - d_{ij}^r(Y)}{rd_{ij}^r(Y)}$$

20.4 astress

robust MDS (zho@u_xu_li_19) LAR (Heiser (1988))

$$\begin{aligned}\sigma_{11}(X) &= \sum_{1 \leq i < j \leq n} \sum w_{ij} |\delta_{ij} - d_{ij}(X)| \\ |(\delta_{ij} - d_{ij}(X)) + \epsilon| &\leq \frac{1}{2} \frac{((\delta_{ij} - d_{ij}(X)) + \epsilon)^2 + ((\delta_{ij} - d_{ij}(Y)) + \epsilon)^2}{|(\delta_{ij} - d_{ij}(Y)) + \epsilon|} \\ \sigma_{rs}(X) &= \sum_{1 \leq i < j \leq n} \sum w_{ij} |\delta_{ij}^r - d_{ij}^r(X)|^s\end{aligned}$$

20.5 pstress

The p in pstress stands for panic. We define

$$\sigma(X) := \sum_{(i < j) \leq (k < l)} \sum w_{ijkl} (\delta_{ij} - d_{ij}(X)) (\delta_{kl} - d_{kl}(X)),$$

where $(i < j) \leq (k < l)$ means that index pair (i, j) is lexicographically not larger than pair (k, l) .

Covariances/variances

How many of these weights w_{ijkl} are there ?

$$\frac{1}{2} \binom{n}{2} \left(\binom{n}{2} + 1 \right) = \frac{1}{8} n(n-1)(n^2-n+2)$$

No reason to panic. Again, majorization comes to the rescue (Groenen, Giaquinto, and Kiers (2003)). Suppose there is a $K > 0$ and a hollow, symmetric, non-negative Ω such that

$$\sum_{(i < j) \leq (k < l)} \sum w_{ijkl} z_{ij} z_{kl} \leq K \sum_{1 \leq i < j \leq n} \omega_{ij} z_{ij}^2.$$

Often the form of the weights w_{ijkl} will suggest how to choose K . In the worst case scenario we choose $\Omega = E - I$ and compute K by the power method. If all w_{ijkl} are equal to one, then ref becomes

$$\frac{1}{2} \left\{ \sum_{1 \leq i < j \leq n} z_{ij} \right\}^2 + \sum_{1 \leq i < j \leq n} z_{ij}^2 \leq K \sum_{1 \leq i < j \leq n} \omega_{ij} z_{ij}^2.$$

$$\sigma(X) \leq \sigma(Y) + 2 \sum \theta_{ij} (d_{ij}(Y) - d_{ij}(X)) + K \sum \omega_{ij} (d_{ij}(Y) - d_{ij}(X))^2$$

with (modify slightly !!)

$$\theta_{ij} = \sum_{1 \leq k < l \leq n} w_{ijkl} (\delta_{kl} - d_k l Y)$$

Chapter 21

Alternative Least Squares Loss

21.1 Sammon's MDS

21.2 McGee's Work

21.3 Shepard's Nonmetric MDS

21.4 Guttman's Nonmetric MDS

21.5 Positive Orthant Nonmetric MDS

!! Richard Johnson 1973

!! Guttman Absolute Value

!! Hartmann

21.6 Role Reversal

Kruskal, arithmetic with dissimilarities

$$\sigma(X) = \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij} - P_r(d_{ij}(X)))^2$$

Chapter 22

Inverse Multidimensional Scaling

In MDS we start with dissimilarities and we find a configuration that locally minimizes stress. We know that the equation we have to solve is $B(X)X = VX$. MDS maps dissimilarities into configurations by finding one (or, ideally, all) solutions of this equation. In Inverse MDS (IMDS) we start with the same equation $B(X)X = VX$, but now we find all dissimilarity matrices for which a given configuration is a local optimum, or at least a stationary point. Thus we solve for Δ for given X , and we study the inverse of the MDS map.

Inverse MDS was first described in De Leeuw and Groenen (1997), R code was provided in De Leeuw (2012), and some elaborations are in De Leeuw, Groenen, and Mair (2016c). This chapter leans heavily on De Leeuw, Groenen, and Mair (2016c), but we have reformulated some results and pruned some of the examples.

In studying the IMDS mapping we limit ourselves, unless explicitly stated otherwise, to configurations X that are *regular*, in the sense that $d_{ij}(X) > 0$ for all $i \neq j$. This can be done without loss of generality. If some of the distances are zero, then the corresponding IMDS problem can be reduced to a regular problem with a smaller number of points (De Leeuw, Groenen, and Mair (2016e)). Also, an $n \times p$ configuration X is *normalized* if it is column-centered and has rank p . For such X there exist $n \times (n - p - 1)$ centered orthonormal matrix K such that $K'X = 0$. In IMDS we will always assume that X is both regular and normalized. We also assume, unless it is explicitly

stated otherwise, that all off-diagonal weights w_{ij} are non-zero.

In De Leeuw and Groenen (1997) two different basic IMDS versions are discussed. Both versions start with the stationary equation $B(X)X = VX$. The first finds all W and Δ for which a given X is stationary. The second finds all Δ for a given X and W for which X is stationary. In this chapter we only look at the second form of IMDS. As we shall see, the first version turns out introduce too many unknowns in W and Δ to be useful. The second form also reflects the point of view that Δ are the data, while the W are part of the definition of the loss function.

22.1 Basic IMDS

Suppose X is a regular and normalized configuration satisfying the stationary equation $(V - B(X))X = 0$. Our first IMDS step is to describe the set $\mathfrak{D}(X)$ of all Δ for which X is stationary.

Lemma 22.1. *Suppose X is an $n \times p$ matrix of rank $p < n$. Suppose K is an $n \times (n - p)$ orthonormal matrix with $K'X = 0$. Then a symmetric matrix A satisfies $AX = 0$ if and only if there is a symmetric S such that $A = KSK'$. If $\text{rank}(K'AK) = r$ then S can be chosen to be of order r .*

::: {.proof} Suppose $X = LT$ with L an orthonormal basis for the column space of X and T non-singular. Write A as

$$A = [L \quad K] \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} L' \\ K' \end{bmatrix}. \quad (22.1)$$

Then $AX = LA_{11}T + KA_{21} = 0$ which is true if and only if $A_{11} = 0$ and $A_{21} = 0$, and by symmetry $A_{12} = 0$. Thus $A = KA_{22}K'$.

```
\begin{theorem}
\label{thm:keyresult} </strong>
\begin{equation}
\delta_{ij}=d_{ij}(X)\left(1-\frac{k_i'Sk_j}{w_{ij}}\right).
\label{eq:invalldelta}
\end{equation}

```

```
\end{equation}
\end{theorem}
```

::: {.proof}

By lemma XXX we have $(V - B(X))X = 0$ if and only if there is a symmetric S such that
:::

Thus $\mathfrak{D}(X)$ is a non-empty affine space, a translation of a linear subspace.

If $\Delta_1, \dots, \Delta_m$ are in $\mathfrak{D}(X)$, then so is the affine subspace

The next result is corollary 6.3 in @deleeuw_groenen_A_97.

```
\begin{corollary}
```

\label{cor:unnamed-chunk-12}

::: {.proof}

If $p=n-1$ then S in theorem \ref{thm:keyresult} is of order zero.

:::

For any two elements of $\mathfrak{D}(X)$, one cannot be elementwise larger (or smaller).

```
\begin{corollary}
```

\label{cor:dominate}

::: {.proof}

With obvious notation $\mathbf{tr}\ X'(B_1(X) - B_2(X))X = 0$, which can be written as

```
\begin{equation}
```

$$\sum_{1 \leq i < j \leq n} w_{ij} (\delta_{1ij} - \delta_{2ij}) d_{ij}$$

\label{eq:invcompare}

```
\end{equation}
```

and thus $\Delta_1 \leq \Delta_2$ implies $\Delta_1 = \Delta_2$.

...

Non-negative Dissimilarities

From equation \eqref{eq:invaliddelta} it follows δ_{ij} is a decreasing

Convex cone, affine convex cone

```
\begin{lemma}
```

\label{lem:goldman}

... { .proof }

See @goldman_56, corollary 1B.

...

If C satisfies the conditions of lemma \ref{lem:goldman} then it is a bound

There are, of course, affine combinations Δ with negative elements. We

```
\begin{theorem}
```

\label{thm:posresult}

```
\begin{equation}
```

$\mathbf{low} \leq \mathbf{low}$.

```
\end{equation}
```

Thus $\Delta(W, X) \cap \Delta_+$ is a convex polyhedron, closed under non-negative

...

... { .proof }

Follows easily from the representation in theorem \ref{thm:keyresult} displayed

...

Of course the minimum of $\sigma(X, W, \Delta)$ over $\Delta \in \Delta(W, X) \cap \Delta_+$

```
\begin{theorem}
<span class="theorem" id="thm:bounded"><strong>\label{thm:bounded} </strong></span>$\backslash
\end{theorem}
```

::: {.proof}

This is corollary 3.2 in @deleeuw_groenen_A_97, but the proof given there is incorrec

```
\begin{equation}
0=\text{tr}\{ X'UX=2\mathop{\sum\sum}_{1\leq i < j \leq n} u_{ij}d_{ij}^2(X)\} .
\end{equation}
```

This implies $U=0$, and the recession cone is the zero vector.

:::

We can compute the vertices of $\Delta(W,X) \cap \Delta_+$ using the complete description.

There is also a brute-force method of converting H to V that is somewhat wasteful, bu

Zero Weights and/or Distances

If a distance is zero then the corresponding element of $B(X)$ must be zero as well.

If, for example, X is

22.2 [1]

22.3 [1,] -0.5

22.4 [2,] -0.5

22.5 [3,] 0.5

22.6 [4,] 0.5

and the weights are all one,

then \$V-B\$ must be a linear combination of the three matrices, say \$P_{11}\$, \$P_{12}\$,

22.7 [,1] [,2] [,3] [,4]

22.8 [1,] 1 -1 1 -1

22.9 [2,] -1 1 -1 1

22.10 [3,] 1 -1 1 -1

22.11 [4,] -1 1 -1 1

22.12. [,1] [,2] [,3] [,4]

251

22.12 [,1] [,2] [,3] [,4]

22.13 [1,] 1 -1 -1 1

22.14 [2,] -1 1 1 -1

22.15 [3,] -1 1 1 -1

22.16 [4,] 1 -1 -1 1

22.17 [,1] [,2] [,3] [,4]

22.18 [1,] -2 2 0 0

22.19 [2,] 2 -2 0 0

22.20 [3,] 0 0 2 -2

22.21 [4,] 0 0 -2 2

and \$B\$ is \$V\$ minus the linear combination. For any \$B\$ computed this way we have \$B = b_{12} = b_{34} = 0\$ if and only if \$B = V - \alpha P_{11} + (1-\alpha) P_{22}\$.

Examples

First Example

As our first example we take \mathbf{X} equal to four points in the corners of a square:

22.22 [1] [,2]

22.23 [1,] -0.5 -0.5

22.24 [2,] -0.5 0.5

22.25 [3,] 0.5 0.5

22.26 [4,] 0.5 -0.5

with distances

22.27 1 2 3

22.28 2 1.000000

22.29 3 1.414214 1.000000

22.30 4 1.000000 1.414214 1.000000

and \mathbf{K} is the vector

22.31 [1] -0.5 0.5 -0.5 0.5

For unit weights we have $\Delta \in \Delta(\mathbf{X}, \mathbf{W})$ if and only if

$\Delta = D(X) \setminus \{W - \lambda kk' \}$
 for some real λ . This means that $\Delta \in \Delta(X, W) \cap \Delta_+$ if and only if
 $\Delta_1 := 2\sqrt{2} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$
 $\Delta_2 := 2 \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$.
 Thus $\Delta(X, W) \cap \Delta_+$ are the convex combinations
 $\Delta(\alpha) := \alpha \Delta_1 + (1-\alpha) \Delta_2 =$
 $\begin{bmatrix} 0 & 2(1-\alpha) & 2\alpha\sqrt{2} & 2(1-\alpha) \\ 2(1-\alpha) & 0 & 2(1-\alpha) & 2(1-\alpha) \\ 2\alpha\sqrt{2} & 2(1-\alpha) & 0 & 2(1-\alpha) \\ 2(1-\alpha) & 2(1-\alpha) & 2(1-\alpha) & 0 \end{bmatrix}$
 This can be thought of as the distances between points on a (generally non-Euclidean) circle of radius $\sqrt{2}/2 \approx 0.585786$.

The distances are certainly Euclidean if Pythagoras is satisfied, i.e. if the square of the distance between two points x, y is given by

$\frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$
 $L := \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{\sqrt{2}}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{\sqrt{2}}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{\sqrt{2}}{2} & \frac{1}{2} \end{bmatrix}$

The diagonal elements of $L'E_1L$ are $1, -1, -1, 1$ and those of $L'E_2L$ are $2, 0, 0, -2$.

We see that

$\Delta(\alpha)$ is two-dimensional Euclidean for $\alpha = \frac{1}{2}$, one-dimensional Euclidean for $\alpha = 0, 1$.

On the unit interval stress is the quadratic $32(\alpha - \frac{1}{2})^2$, which attains its minimum at $\alpha = \frac{1}{2}$.

Second Example

We next give another small example with four equally spaced points on the line

22.32 [, 1]

22.33 [1,] -0.6708204

22.34 [2,] -0.2236068

22.35 [3,] 0.2236068

22.36 [4,] 0.6708204

and $D(X)$ is

22.37 1 2 3

22.38 2 3.464102

22.39 3 4.472136 2.828427

22.40 4 6.324555 4.472136 3.464102

For S we use the basis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$, the three matrices

22.41 1 2 3**22.42 2 4.330127****22.43 3 5.590170 2.121320****22.44 4 4.743416 5.590170 4.330127****22.45 1 2 3****22.46 2 3.983717****22.47 3 3.801316 4.101219****22.48 4 6.640783 3.801316 3.983717****22.49 1 2 3****22.50 2 1.914908****22.51 3 5.472136 2.828427****22.52 4 6.324555 3.472136 5.013295**

Both `scdd()` from `rcdd` and `enumerate.vertices()` from `vertexenum` find the same

22.52. 4 6.324555 3.472136 5.013295

257

22.53 1 2 3

22.54 2 20.784610

22.55 3 0.000000 5.656854

22.56 4 0.000000 13.416408 0.000000

22.57 1 2 3

22.58 2 13.856406

22.59 3 4.472136 0.000000

22.60 4 0.000000 13.416408 0.000000

22.61 1 2 3

22.62 2 20.78461

22.63 3 0.00000 22.62742

22.64 4 0.00000 0.00000 20.78461

22.65 1 2 3

22.66 2 0.000000

22.67 3 13.416408 5.656854

22.68 4 0.000000 0.000000 20.784610

22.69 1 2 3

22.70 2 0.000000

22.81. [1] 460.00000 248.00000 1072.00000 460.00000 248.00000 36.44444 112.00000259

**22.81 [1] 460.00000 248.00000 1072.00000
460.00000 248.00000 36.44444 112.00000**

and we know that 1072 is the maximum of stress over $\Delta \in \Delta(X, W) \cap \Delta_+$.

In general the vanishing of the stationary equations does not imply that X corresponds to an extreme point.

Third Example

The number of extreme points of the polytope $\Delta(W, X) \cap \Delta_+$ grows very quickly, so it can be difficult to determine exactly how many extreme points there are.

@deleeuw_groenen_A_97 analyzed this example and found 42 extreme points. We repeat the analysis. We select $\binom{15}{6} = 5005$ sets of six linear equations from our 15 linear inequalities. We use our function `cleanUp()` to remove duplicates, which leaves 42 vertices, same number as @deleeuw_groenen_A_97.

- 22.82** [1] **24.00000 24.00000 24.00000 21.75000
12.00000 13.33333 6.66667 13.33333**
- 22.83** [9] **17.33333 17.33333 19.68000 13.33333
6.66667 17.33333 6.66667 21.75000**
- 22.84** [17] **6.66667 60.00000 24.00000 21.75000
19.68000 17.33333 21.75000 13.33333**
- 22.85** [25] **19.68000 17.33333 17.33333
21.75000 17.33333 17.33333 13.33333
6.66667**
- 22.86** [33] **21.75000 17.33333 19.68000
13.33333 17.33333 19.68000 19.68000
17.33333**
- 22.87** [41] **6.66667 17.33333**

and their maximum is 60.

If we perform the calculations more efficiently in `rcdd`, using rational arithmetic, the result is different:

The fact that we get different numbers of vertices with different methods is something that can happen when solving linear systems. A system \$Ax \leq b\$ satisfying the \$n \times m\$ system \$Ax \leq b\$ is an extreme point if and only if

22.88. [,1] [,2]

261

rank \$m\$. It turns out all the additional vertices found by `rcdd` and `vertexenum` do

Fourth Example

This is an unfolding example with \$n=3+3\$ points, configuration

22.88 [,1] [,2]

22.89 [1,] 0.0000000 0.0000000

22.90 [2,] 0.0000000 0.0000000

22.91 [3,] 0.0000000 -0.8164966

22.92 [4,] 0.0000000 0.4082483

22.93 [5,] 0.7071068 0.0000000

22.94 [6,] -0.7071068 0.4082483

and weight matrix

22.95 [1] [2] [3] [4] [5] [6]

22.96 [1,] 0 0 0 1 1 1

22.97 [2,] 0 0 0 1 1 1

22.98 [3,] 0 0 0 1 1 1

22.99 [4,] 1 1 1 0 0 0

22.100 [5,] 1 1 1 0 0 0

22.101 [6,] 1 1 1 0 0 0

Note that row-points one and two in \$X\$ are equal, and thus $d_{12}(X)=0$. These are diagonal blocks, because they are not part of the MDS problem.

Using our brute force method, we find the two edges

22.102 4 5 6

22.103 1 0.0000000 1.414214 1.632993

22.104 2 0.8164966 0.000000 0.000000

22.105 3 1.2247449 1.080123 1.414214

22.106 4 5 6

22.107 1 0.8164966 0.000000 0.000000

22.108 2 0.0000000 1.414214 1.632993

22.109 3 1.2247449 1.080123 1.414214

and the off-diagonal blocks for which $\$X\$$ is an unfolding solution are convex combinations of the diagonal blocks.

MDS Sensitivity

Suppose $\$X\$$ is a solution to the MDS problem with dissimilarities $\$\\Delta\$$, found by

For typical MDS examples there is no hope of computing all vertices of $\$\\mathfrak{D}(\mathbb{R}^n) \times \\mathfrak{D}(\\mathbb{R}^m)\$$. There are $n=9$ objects, to be scaled in $p=2$ dimensions. We have $\frac{n(n-1)}{2}=36$ dissimilarities and $\frac{m(m+1)}{2}=21$ variables. It suffices to consider that there are 5567902560 ways to choose a set of 36 points in \mathbb{R}^2 .

What we can do, however, is to optimize linear (or quadratic functions) over $\$\\mathfrak{D}(\mathbb{R}^n) \times \\mathfrak{D}(\\mathbb{R}^m)\$$. Define an easily manageable LP (or QP) problem. As an example, not necessarily δ_{ij} to the largest possible δ_{ij} turns out to be quite large.

The data are

22.110. *KVP PVDA VVD ARP CHU CPN PSP BP D66* 265

**22.110 KVP PvdA VVD ARP CHU CPN
PSP BP D66**

**22.111 KVP 0.00 5.63 5.27 4.60 4.80 7.54 6.73
7.18 6.17**

**22.112 PvdA 5.63 0.00 6.72 5.64 6.22 5.12
4.59 7.22 5.47**

**22.113 VVD 5.27 6.72 0.00 5.46 4.97 8.13 7.55
6.90 4.67**

**22.114 ARP 4.60 5.64 5.46 0.00 3.20 7.84 6.73
7.28 6.13**

**22.115 CHU 4.80 6.22 4.97 3.20 0.00 7.80 7.08
6.96 6.04**

**22.116 CPN 7.54 5.12 8.13 7.84 7.80 0.00 4.08
6.34 7.42**

**22.117 PSP 6.73 4.59 7.55 6.73 7.08 4.08 0.00
6.88 6.36**

**22.118 BP 7.18 7.22 6.90 7.28 6.96 6.34 6.88
0.00 7.36**

**22.119 D66 6.17 5.47 4.67 6.13 6.04 7.42 6.36
7.36 0.00**

The optimal configuration found by `smacof` is

22.120 [1] [2]

22.121 [1,] 1.78 3.579

22.122 [2,] -1.46 2.297

22.123 [3,] 3.42 -2.776

22.124 [4,] 3.30 1.837

22.125 [5,] 3.84 0.308

22.126 [6,] -5.09 -0.044

22.127 [7,] -3.79 2.132

22.128 [8,] -2.86 -4.318

22.129 [9,] 0.86 -3.015

\begin{figure}

{\centering \includegraphics{_main_files/figure-latex/poldistconfplot-1}}

}

\caption{De Gruijter Configuration}\label{fig:poldistconfplot}\\
\end{figure}

The maximum and minimum dissimilarities in $\Delta(X, W) \cap \Delta_+$ are

22.129. [9,] 0.86 -3.015

267

22.130	KVP PvdA VVD ARP CHU CPN PSP BP D66
22.131	KVP 0.0 32.6 9.7 9.3 24.6 24.3 11.7 18.9 15.2
22.132	PvdA 32.6 0.0 25.7 36.9 20.7 34.1 36.1 29.7 22.2
22.133	VVD 9.7 25.7 0.0 17.4 32.0 34.4 22.3 23.5 20.4
22.134	ARP 9.3 36.9 17.4 0.0 28.0 33.7 25.6 28.9 23.0
22.135	CHU 24.6 20.7 32.0 28.0 0.0 20.8 31.9 33.3 26.3
22.136	CPN 24.3 34.1 34.4 33.7 20.8 0.0 24.9 30.7 31.8
22.137	PSP 11.7 36.1 22.3 25.6 31.9 24.9 0.0 23.7 27.7
22.138	BP 18.9 29.7 23.5 28.9 33.3 30.7 23.7 0.0 35.0
22.139	D66 15.2 22.2 20.4 23.0 26.3 31.8 27.7 35.0 0.0

22.139. D66 15.2 22.2 20.4 23.0 26.3 31.8 27.7 35.0 0.0 269

22.140	KVP PvdA VVD ARP CHU CPN PSP BP D66
22.141	KVP 0.00 3.48 0.00 0.00 2.34 0.00 0.00 0.00 0.00
22.142	PvdA 3.48 0.00 0.00 0.00 0.00 0.00 0.00 0.93 0.00 0.00
22.143	VVD 0.00 0.00 0.00 0.00 0.83 0.00 0.00 0.00 0.00
22.144	ARP 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
22.145	CHU 2.34 0.00 0.83 0.00 0.00 0.00 0.00 0.00 0.00
22.146	CPN 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
22.147	PSP 0.00 0.93 0.00 0.00 0.00 0.00 0.00 0.00 0.00
22.148	BP 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
22.149	D66 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Second Order Inverse MDS

As we mentioned many times before, stationary values are not necessarily local minima
\$\$

$\Delta_H(W, X) := \{\Delta \mid \mathcal{D}^2 \sigma(X) \gtrsim 0\}.$

\$\$

We can now study $\Delta(W, X) \cap \Delta_H(W, X)$ or $\Delta(W, X) \cap \Delta_H(W, X) \cap \Delta$.

`\begin{theorem}`

`\label{thm:invconvset} `
`\end{theorem}`

`::: {.proof}`

In MDS the Hessian is $2(V - H(x, W, \Delta))$, where

`\begin{equation}`

$H(x, W, \Delta) := \mathop{\sum \sum}_{1 \leq i < j \leq n} w_{ij} \frac{\delta_{ij}}{d_{ij}(x)}$
`\end{equation}`

Here we use $x := \mathbf{vec}(X)$, and both A_{ij} and V are direct sums of p components

`:::`

In example 3 the smallest eigenvalues of the Hessian at the 42 vertices are

- 22.150** [1] 0.00000 0.00000 0.00000 -5.86238
0.00000 -2.96688 0.00000
- 22.151** [8] -2.96688 -4.47894 -4.47894 -
5.71312 -2.96688 0.00000 -5.00453
- 22.152** [15] 0.00000 -5.86238 0.00000 -
12.00000 0.00000 -5.86238 -5.71312
- 22.153** [22] -5.00453 -5.86238 -2.96688 -
5.71312 -4.47894 -5.00453 -5.86238
- 22.154** [29] -5.00453 -5.00453 -2.96688
0.00000 -5.86238 -4.47894 -5.71312
- 22.155** [36] -2.96688 -5.00453 -5.71312 -
5.71312 -4.47894 0.00000 -4.47894

and thus there are at most 11 vertices corresponding with local minima.

The next step is to refine the polyhedral approximation to $\Delta(W, X) \cap D$ those vertices for which this smallest eigenvalue is negative. Thus, if the ei

$$\begin{aligned} & \text{\begin{equation}} \\ & \mathop{\sum\sum}_{1 \leq i < j \leq n} \zeta_{ij} (w_{ij} - k_i' S_k_j) \geq 0, \\ & \text{\end{equation}} \end{aligned}$$

where

\$\$

$$\zeta_{ij} := y' \left(A_{ij} - \frac{A_{ij} x x' A_{ij}}{x' A_{ij} x} \right) y.$$

It is clear, however, that adding a substantial number of linear inequalities

```
22.155. [36] -2.96688 -5.00453 -5.71312 -5.71312 -4.47894 0.00000 -4.47894273
```

```
## Inverse FDS
```

A configuration \$X\$ is a ***full dimensional scaling*** or ***FDS*** solution (@deleeuw_groen

```
```{theorem< label = "invfull"}
$\Delta \in \Delta_F(W, X)$ if and only if there is a positive semi-definite S such that
\begin{equation}
\delta_{ij} = d_{ij}(X) \left(1 - \frac{1}{w_{ij}} k_i' S k_j\right),
\end{equation}
Thus $\Delta_F(W, X)$ is a non-polyhedral convex set, closed under linear combinations
```

*Proof.* Of course  $\Delta_F(W, X) \subseteq \Delta(W, X)$ . Thus  $V - B(X) = KSK'$  for some  $S$ , and  $XX'$  and  $V - B(X)$  must both be positive semi-definite, with complementary null spaces.  $\square$

We reanalyze our second example, with the four points equally spaced on the line, requiring a positive semi-definite  $S$ . We start with the original 7 vertices, for which the minimum eigenvalues of  $S$  are

```
[1] -4.000 -2.899 -1.708 -3.333 8.000 -1.708 -2.899
```

If the minimum eigenvalue is negative, with eigenvector  $y$ , we add the constraints  $y'Sy \geq 0$ . This leads to 11 vertices with minimum eigenvalues

```
[1] 8.0000 -0.0355 0.0000 -0.7911 -0.3834 -0.3834 -0.0355 -0.0853 -0.0853
[10] -0.7911 0.0000
```

We see that the negative eigenvalues are getting smaller. Repeating the procedure of adding constraints based on negative eigenvalues three more times gives 19, 37, and 79 vertices, with corresponding minimum eigenvalues

```
[1] 8.000000 -0.000021 0.000000 -0.209852 -0.202322 -0.085642 -0.085642
[8] -0.000021 -0.106132 -0.018912 -0.008051 -0.023967 -0.008051 -0.023967
[15] -0.106132 -0.018912 -0.209852 -0.202322 0.000000
```

```

[1] 8.000000 -0.000021 0.000000 -0.056872 -0.053562 -0.045307 -0.062028
[8] -0.020853 -0.020853 -0.000021 -0.028793 -0.004490 -0.001925 -0.006410
[15] -0.001925 -0.006410 -0.028793 -0.004490 -0.000005 -0.002104 -0.021934
[22] -0.024355 -0.021934 -0.024355 -0.000005 -0.002104 -0.004975 -0.005596
[29] -0.005596 -0.004975 -0.002318 -0.002318 -0.056872 -0.053562 -0.045307
[36] -0.062028 0.000000

[1] 8.000000 -0.000021 0.000000 -0.015026 -0.013698 -0.010861 -0.017838
[8] -0.013830 -0.013426 -0.012084 -0.013998 -0.018140 -0.005180 -0.005180
[15] -0.000021 -0.007568 -0.001096 -0.000471 -0.001662 -0.000471 -0.001662
[22] -0.007568 -0.001096 -0.000005 -0.000001 -0.000538 -0.000488 -0.005588
[29] -0.005886 -0.005588 -0.005886 -0.000005 -0.000001 -0.000538 -0.000488
[36] -0.001278 -0.001355 -0.001355 -0.001278 -0.000001 -0.000649 -0.000594
[43] -0.005244 -0.005378 -0.005244 -0.005378 -0.000001 -0.000649 -0.000594
[50] -0.006838 -0.006293 -0.001150 -0.001210 -0.000492 -0.000514 -0.000566
[57] -0.001545 -0.001444 -0.000492 -0.000514 -0.000566 -0.001545 -0.001444
[64] -0.006838 -0.006293 -0.001150 -0.001210 -0.000001 -0.000001 -0.015026
[71] -0.013698 -0.010861 -0.017838 -0.013830 -0.013426 -0.012084 -0.013998
[78] -0.018140 0.000000

```

It should be noted that the last step already takes an uncomfortable number of minutes to compute. Although the number of vertices goes up quickly, the diameter of the polygon (the maximum distance between two vertices) slowly goes down and will eventually converge to the diameter of  $\Delta_F(W, X) \cap \Delta_+$ . Diameters in subsequent steps are

```
[1] 5.657 5.086 5.065 5.061 5.060
```

## 22.156 Multiple Solutions

If  $X_1, \dots, X_s$  are configurations with the same number of points, then the intersection  $\{\bigcap_{r=1}^s \Delta(W, X_r)\} \cap \Delta_+$  is again a polygon, i.e. a closed and bounded convex polyhedron (which may be empty). If  $\Delta$  is in this intersection then  $X_1, \dots, X_s$  are all solutions of the stationary equations for this  $\Delta$  and  $W$ .

Let's look at the case of two configurations  $X_1$  and  $X_2$ . We must find vectors  $t_1$  and  $t_2$  such that

$$d_1 \circ (e - w^\dagger \circ G_1 t_1) = d_2 \circ (e - w^\dagger \circ G_2 t_2).$$

If  $H_1 := \text{diag}(d_1 \circ w^\dagger)G_1$  and  $H_2 := \text{diag}(d_2 \circ w^\dagger)G_2$  then this can be written as

$$\begin{bmatrix} H_1 & -H_2 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = d_1 - d_2$$

This is a system of linear equations in  $t_1$  and  $t_2$ . If it is solvable we can intersect it with the convex sets  $G_1 t_1 \leq w$  and  $G_2 t_2 \leq w$  to find the non-negative dissimilarity matrices  $\Delta$  for which both  $X_1$  and  $X_2$  are stationary.

```
$delta1
[,1]
[1,] 1.464102
[2,] 1.464102
[3,] 1.464102
[4,] 1.464102
[5,] 1.464102
[6,] 1.464102
##
$delta2
[,1]
[1,] 1.464102
[2,] 1.464102
[3,] 1.464102
[4,] 1.464102
[5,] 1.464102
[6,] 1.464102
##
$res
[1] 1.024137e-15
##
$rank
[1] 2
```

As a real simple example, suppose  $X$  and  $Y$  are four by two. They both have three points in the corners of an equilateral triangle, and one point in

the centroid of the other three. In  $X$  the fourth point is in the middle, in  $Y$  the first point. The only solution to the linear equations is the matrix with all dissimilarities equal.

For a much more complicated example we can choose the De Gruijter data. We use `smacof` to find two stationary points in two dimensions. The matrices  $G_1$  and  $G_2$  are  $36 \times 21$ , and thus  $H := (H_1 \mid -H_2)$  is  $36 \times 42$ . The solutions of the linear system  $Ht = d_1 - d_2$  are of the form  $t_0 - Lt$ , with  $t_0$  an arbitrary solution and  $L$  a  $42 \times 6$  basis for the space of  $Ht = 0$ . To find the non-negative solutions we can use the H representation  $Lv \leq t_0$ , and then compute the V representation, realizing of course that we can choose 6 rows from 42 rows in 5245786 ways.

## 22.157 Minimizing iStress

The IMDS approach can also be used to construct an alternative MDS loss function. We call it *iStress*, defined as

$$\sigma_t(X, W, \Delta) := \min_{\tilde{\Delta} \in \Delta(W, X) \cap \Delta_+} \sum_{1 \leq i < j \leq n} w_{ij} (\delta_{ij} - \tilde{\delta}_{ij})^2.$$

Minimizing iStress means minimizing the projection distance between the observed dissimilarity matrix and the moving convex set of non-negative dissimilarity matrices for which  $X$  satisfies the stationary equations. The convex set is moving, because it depends on  $X$ . For each  $X$  we have to solve the IMDS problem of finding  $\Delta(X, W) \cap \Delta_+$ , and then solve the quadratic programming problem that computes the projection.

**Theorem 22.1.**  $\min_X \sigma_t(X, W, \Delta) = 0$  and the minimum is attained at all  $X$  with  $(V - B(X))X = 0$ .

*Proof.* If  $X$  is a stationary point of stress then  $\Delta \in \mathfrak{D}(X) \cap \mathfrak{D}_+$  and thus iStress is zero. Conversely, if iStress is zero then  $\Delta \in \mathfrak{D}(X) \cap \mathfrak{D}_+$  and  $X$  is a stationary point of stress.  $\square$

Minimizing iStress may not be a actual practical MDS method, but it has some conceptual interest, because it provides another way of looking at the relationship of MDS and IMDS.

We use the De Gruijter data for an example of iStress minimization. We use `optim` from base R, and the `quadprog` package of Turlach and Weingessel (2013). Two different solutions are presented, the first with iterations starting at the classical MDS solution, the second starting at the `smacof` solution. In both cases iStress converges to zero, i.e. the configurations converge to a solution of the stationary equations for stress, and in the `smacof` case the initial solution already has stress equal to zero.

```
initial value 106.624678
iter 10 value 0.205734
iter 20 value 0.025163
iter 30 value 0.018427
iter 40 value 0.000116
iter 50 value 0.000007
iter 60 value 0.000000
final value 0.000000
converged

initial value 0.000000
iter 10 value 0.000000
iter 20 value 0.000000
final value 0.000000
converged
```

## 22.158 Order three

We will now consider, in some detail, MDS of a dissimilarity matrix of order three.. The plan is as follows. Our MDS problem is of order three, i.e. there are only three objects, and three dissimilarities between them. Suppose, for simplicity, that all weights are one. At a stationary point of we have  $B(X)X = 3X$ . Thus the  $B$  matrix has an eigenvalue equal to 3 (in addition to having an eigenvalue equal to zero).

!! Not just eval 3, but also evecs X

If there are only three objects we can look at the set of doubly-centered matrices of order three with an eigenvalue equal to three. If three is the

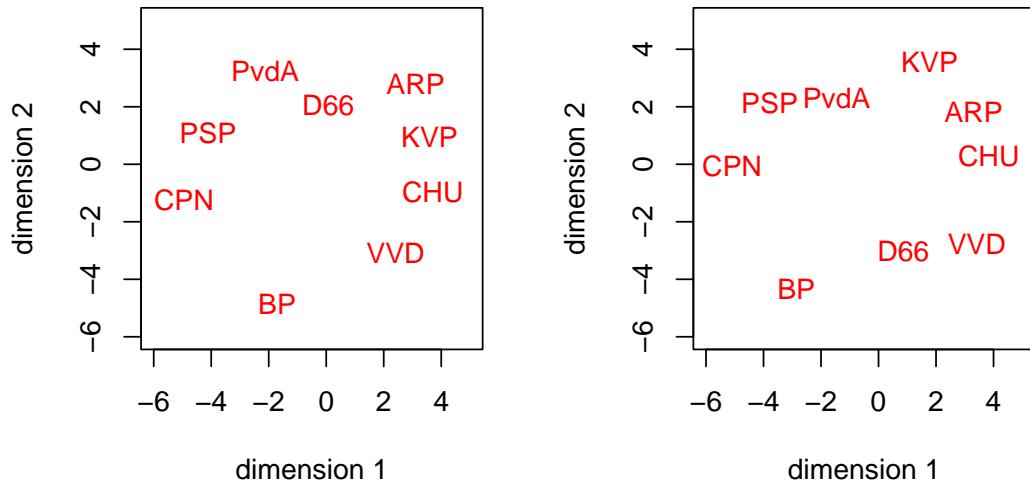


Figure 22.1: De Gruijter Minimim iStress Configuration

largest eigenvalue then the local minimum is global, if it is the second largest eigenvalue then it may or may not be global. If  $B$  has two eigenvalues equal to three, then  $B = 3J$  and the two eigenvectors give the unique global minimum in two dimensions (an equilateral triangle).

A symmetric doubly-centered matrix of order three is a linear combination  $B = \alpha A_{12} + \beta A_{13} + \gamma A_{23}$  of three diff matrices (25.2.3), with  $\alpha, \beta, \gamma$  all non-negative. Thus

$$B = \begin{bmatrix} \alpha + \beta & -\alpha & -\beta \\ -\alpha & \alpha + \gamma & -\gamma \\ -\beta & -\gamma & \beta + \gamma \end{bmatrix}. \quad (22.2)$$

$B$  has one eigenvalue equal to zero, and two real non-negative eigenvalues  $\lambda_1$  and  $\lambda_2$ . The characteristic equation is  $f(\lambda) := \lambda(\lambda^2 - 2\tau\lambda + 3\eta) = 0$ , where  $\tau := \alpha + \beta + \gamma$  and  $\eta := \alpha\beta + \alpha\gamma + \beta\gamma$ .

Thus we have at least one eigenvalue equal to three if  $f(3) = 0$ , i.e.  $2\tau - \eta = 3$ . Both eigenvalues are equal to three if  $\eta = \tau = 3$ , which means  $\alpha = \beta = \gamma = 1$  and  $B = 3J$ .

The two eigenvalues are  $\tau \pm \sqrt{\tau^2 - 3\eta}$ . Note that  $\tau^2 - 3\eta \geq 0$  with equality if and only if  $\tau = \eta = 3$  if and only if  $\alpha = \beta = \gamma = 1$ . This easily follows from  $\tau^2 - 3\eta = \alpha^2 + \beta^2 + \gamma^2 - \eta$ , while  $\eta = \alpha\beta + \alpha\gamma + \beta\gamma \leq \alpha^2 + \beta^2 + \gamma^2$

by applying AM/GM three times. Also note that if two out of the three of  $\alpha, \beta, \gamma$  are zero then  $\eta = 0$  and thus  $\lambda_1 = 2\tau$  and  $\lambda_2 = 0$ .

Since  $\lambda_1 \geq \lambda_2$  we have the two possibilities  $\lambda_2 \leq \lambda_1 = 3$  and  $3 \geq \lambda_1 \geq \lambda_2 = 3$ . Now  $\lambda_1 = 3$  iff  $\sqrt{\tau^2 - 3\eta} = 3 - \tau$  iff  $\tau \leq 3$  and  $2\tau - \eta = 3$ . And  $\lambda_2 = 3$  iff  $\sqrt{\tau^2 - 3\eta} = \tau - 3$  iff  $\tau \geq 3$  and  $2\tau - \eta = 3$ . It also follows that it is necessary for  $\lambda = 3$  that  $\tau \geq \frac{3}{2}$ .

If we have  $(\tau, \eta)$  we can solve for  $\alpha, \beta$  and  $\gamma$  by

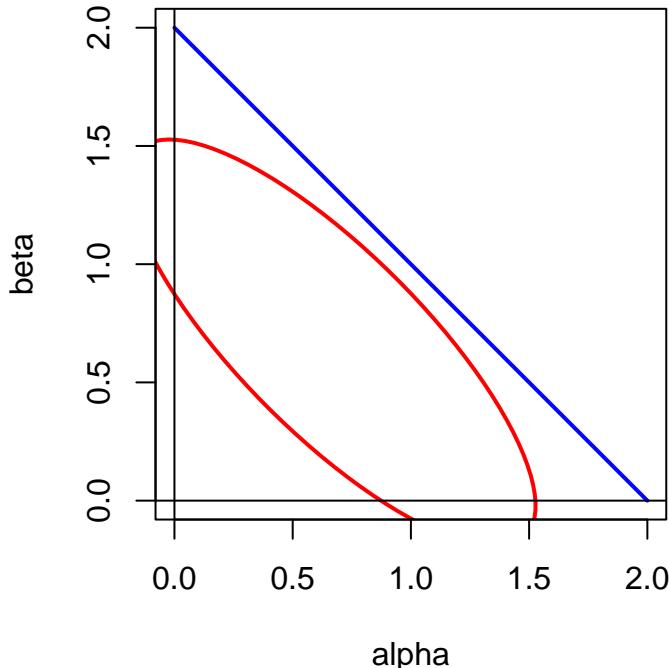
$$\alpha + \beta + \gamma = \tau, \alpha^2 + \beta^2 + \gamma^2 = \omega$$

Thus the set of  $(\alpha, \beta, \gamma)$  corresponding with  $\tau, \eta$  is the intersection of a sphere and an equilateral triangle in the non-negative orthant of  $\mathbb{R}^3$ .

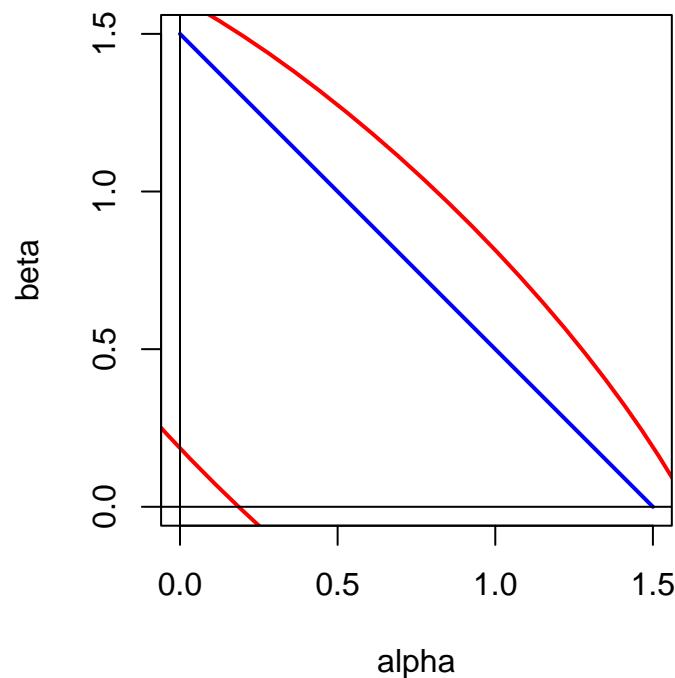
$$\gamma = \tau - \alpha - \beta\alpha^2 + \beta^2 - \tau\alpha - \tau\beta + \alpha\beta = -\eta$$

ellipse with center  $\frac{1}{3}\tau e$  and radius  $\frac{1}{3}(\tau - 3)^2$

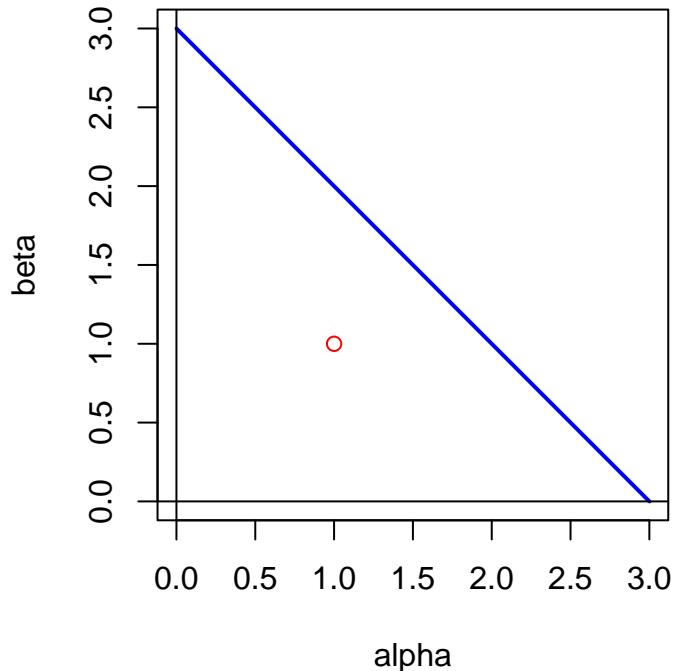
`imdsSolver(2)`



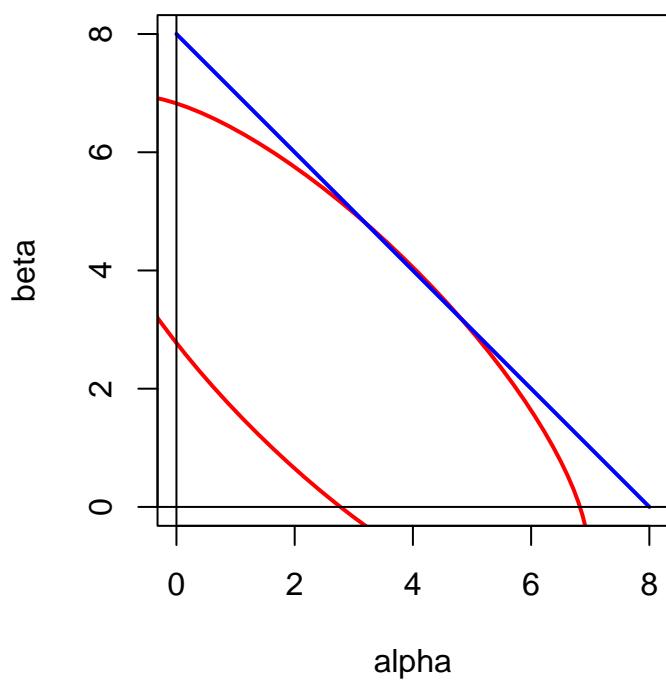
```
imdsSolver(3/2)
```



```
imdsSolver(3)
points (1, 1, col = "RED")
```



```
imdsSolver(8)
```



## 22.159 Order Four

$$B = \alpha A_{12} + \beta A_{13} + \gamma A_{14} + \delta A_{23} + \epsilon A_{24} + \phi A_{34}$$

$$B = \begin{bmatrix} \alpha + \beta + \gamma & -\alpha & -\beta & -\gamma \\ -\alpha & \alpha + \delta + \epsilon & -\delta & -\epsilon \\ -\beta & -\delta & \beta + \delta + \phi & -\phi \\ -\gamma & -\epsilon & -\phi & \gamma + \epsilon + \phi \end{bmatrix}.$$

The characteristic equation is

$$f(\lambda) = \lambda(\lambda^3 - 2\tau\lambda^2 + (3\eta + (\alpha\phi + \beta\epsilon + \gamma\delta))\lambda - Y).$$

$$Z = \frac{1}{2} \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}.$$

# **Chapter 23**

## **Stability of MDS Solutions**

**23.1 Null Distribution**

**23.2 Pseudo-confidence Ellipsoids**

**23.3 A Pseudo-Bootstrap**



# Chapter 24

## In Search of Global Minima

We have already discussed the problem of finding the global minimum, instead of merely one or more local minima, in chapters 13 and 12. In the uni-dimensional case the basic MDS problem becomes combinatorial, we have to minimize over all  $n!$  permutations of  $\iota_n$ , and there usually are very many local minima, all of them strict. The case in which all  $\delta_{ij}$  are the same shows there can be  $n!$  local minima, all global. All these minima are strict and isolated, and thus a small perturbation of the equal-dissimilarity case still has  $n!$  local minima (Pliner (1996)). In the full-dimensional case there is only one minimum, which is by definition the global minimum. For  $1 < p < n - 1$  we can expect to be somewhere between these two extremes, with in addition the possibility that some of the critical points are saddle points and not local minima. But note that if all dissimilarities are equal all permutations of the points in the global minimum configuration also give global minima, which means that even in higher dimensional cases we may have  $n!$  local minima.

One way to protect against non-global minima is to start with a really good initial configuration. Generally, the Torgerson and Guttman initial configurations are helpful, and so are the first  $p$  principal components of the full-dimensional solution. Another important tool is to stop iterations using the size of the gradient (or the difference between  $X$  and  $\mathfrak{G}(X)$ ), with a cut-off value of at least  $1e - 7$ , but preferably  $1e - 10$  or even  $1e - 15$ . Earlier implementations of smacof may have stopped too soon if the target local minimum is in a flat region of the configuration space, or if the iterations stray too close to a saddle point and must flex their muscle to get away from

it.

In this chapter we will discuss some methods to find the global minimum, or, more modestly, to move from one local minimum to another better local minimum. The global optimization battlefield is in constant flux. New methods for general or specific global optimization problems seem to be invented every day, all struggling with the curse of dimensionality. The field is riddled with the remains of methods that died in infancy. So I am not saying I will discuss the best, or even the most promising, global optimization methods for MDS. I simply choose the ones I like best, and the ones that fit nicely into the smacof framework.

## 24.1 Random Starts

The simplest way to get an idea about the local minima of stress in any specific example is to run smacof with multiple random initial configurations. The implementation is simple. Put the smacof runs in a loop from 1 to  $N$ , start each run with a random initial configuration, and collect the results in some data structure. It is true that our analysis will take  $N$  times as long to finish, but just start up your PC and go and do something else while it runs. It seems to me that this ought to be standard practice for actual MDS applications. Not only do we find the best local minimum in terms of stress, but we get valuable information about the stability of our result. If we take, for example,  $N = 1000$  and we find the same local minimum in all runs, we can be reasonably confident that we have found the global minimum. A small change in the dissimilarities will probably find the same global minimum. If we find multiple local minima, all with about the same frequency and with stress values that are close, then a small change may very well switch to another local minimum with the smallest stress value.

There is some freedom in the choice of method to generate random initial configurations. In the examples in this chapter we sample from the standard multivariate normal, but since we know that at local minima  $\eta(X) \leq 1$  it may be more appropriate to sample from the unit ball in  $\mathbb{R}^{n \times p}$  (Harman and Lacko (2010)). In fact, since we also know that at a local minimum  $\rho(X) = \eta^2(X)$ , we may project our intial configurations on that surface.

metric - nonmetric

Table 24.1: Local Minima in Ekman Example

no	minimum	frequency
1	0.0086066	824
2	0.0182714	136
3	0.0300985	19
4	0.0309922	7
5	0.0317395	8
6	0.0331013	4
7	0.0337281	1
8	0.0379281	1

## 24.1.1 Examples

### 24.1.1.1 Ekman

We use the Ekman data to illustrate this. We use 1000 random starts, and we stop iterating when the decrease in stress is less than 1e-15. We find 8 local minima, listed in table 24.1.

The results are encouraging, because they indicate that, at least in the Ekman example, the lower the local minimum, the more attractive it is for the smacof iterations. The lowest local minima have the largest regions of attraction. Specifically, we find what is presumably the global minimum in more than 80% of the cases. Nevertheless, if our clients were so unwise to start their MDS analysis with a random initial configuration then about 20% of them will get the wrong answer.

In the example we also see a number of local minima, found only in a small number of cases, whose stress values are very close to each other. It seems likely that others of roughly the same stress level will be found if we continue sampling additional random initial configurations.

The configurations corresponding with the two dominant local minima are in 24.3 and 24.4. if we compare them we see that the color circle has two separate circular segments. In the second local minimum the order of the colors on one of these two segments is reversed.

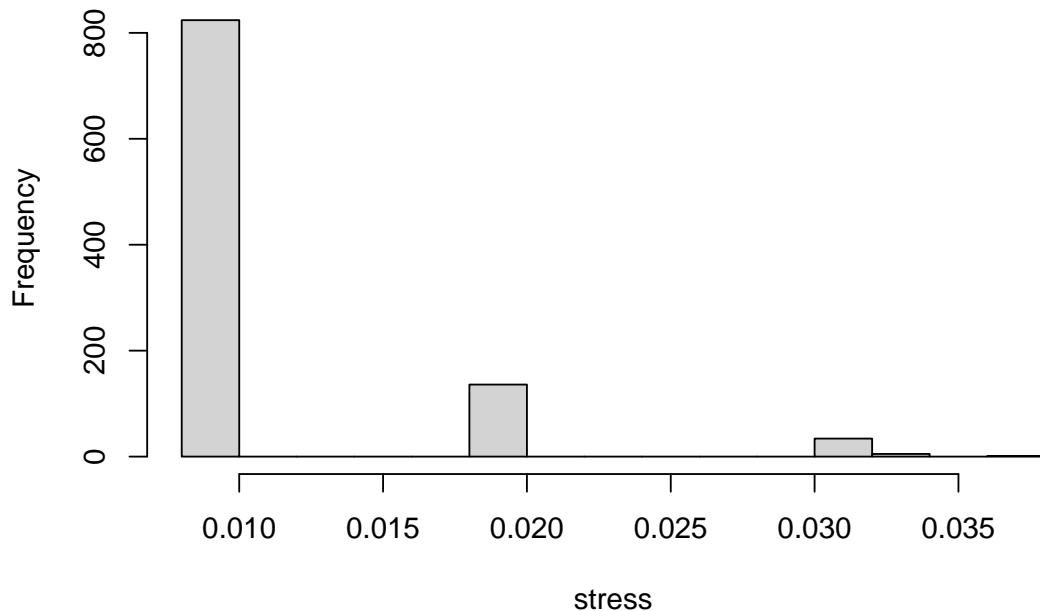


Figure 24.1: Ekman Stress Values

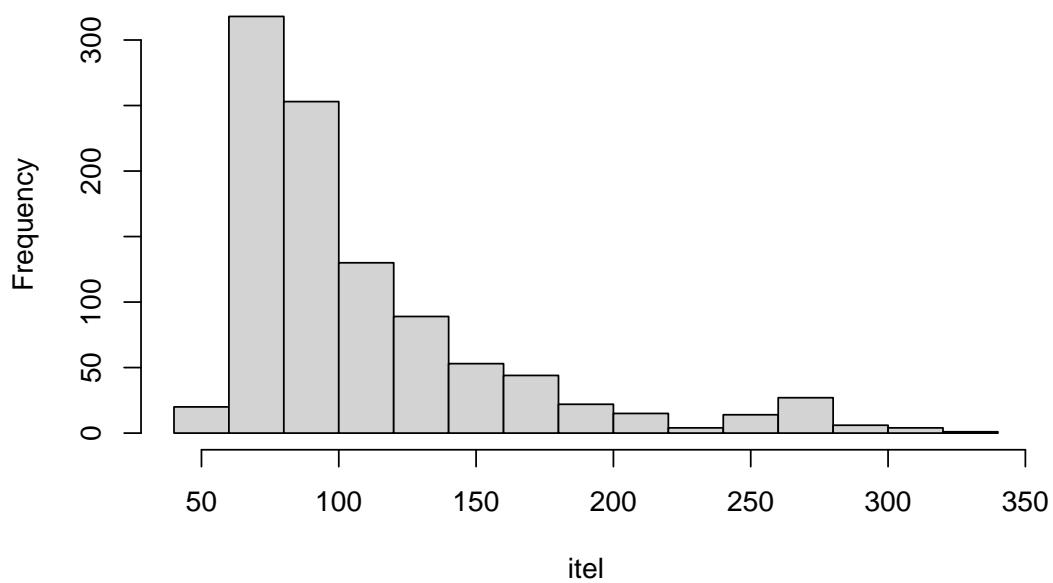


Figure 24.2: Ekman Iteration Counts

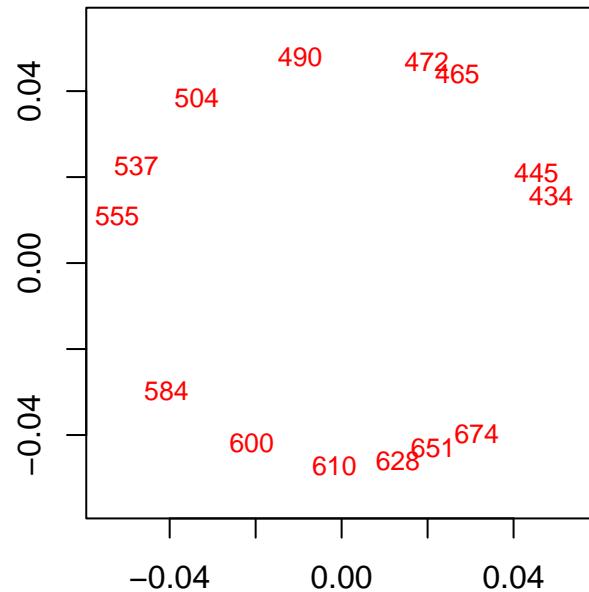


Figure 24.3: Global (?) Minimum

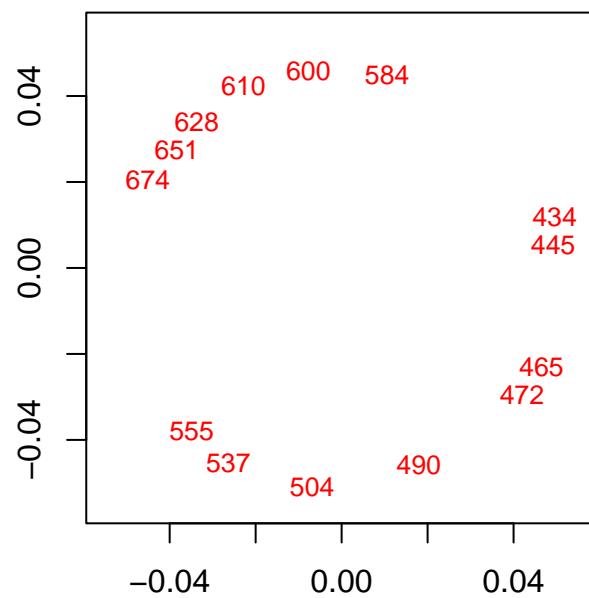


Figure 24.4: Largest (?) Non-global Minimum

### 24.1.1.2 De Grujter

The Ekman example is somewhat atypical, because it has an exceptionally good fit. We perform the same computations for the De Grujter example, still using a cut-off at 1e-15, but now allowing for up to 10,000 iterations.

The data in this example are averages of preference rankings for nine Dutch political parties by 100 students. Due to the heterogeneity of the population there is a considerable regression to the mean. Typically this would suggest splitting the students into more homogeneous groups (which is what De Grujter (1967) did), and/or using a form of non-metric scaling (which is what De Grujter did as well).

Our 1000 runs produce 21 local minima, in table @ref{tab:grijterlocalminima}, with stress values that are close to each other. In this case it is easy to imagine that more runs will produce many more local minima, with low frequencies, mainly permuting the political parties on the horseshoe. In other words, the situation is somewhat like the case in which all dissimilarities between the nine objects are equal, in which case we have  $9! = 3.6288 \times 10^5$  local minima, all with the same stress value.

Another comparison may be useful. The Ekman example is like a matrix with two dominant eigenvalues, the De Grujter example is like a matrix with all eigenvalues approximately equal to each other. Since smacof is somewhat like the power method, we expect poor convergence in the De Grujter example, and histogram 24.6 shows exactly that. There is even one random start from which there is no converged in 10,000 iterations. In the Ekman example the frequency of the local minima seems closely related to the stress value at the local minimum, in the De Grujter example the frequency of the local minima seems more random. In the Ekman case we can be pretty sure we have found the global minimum, in the De Grujter case we are far from sure.

```
x1 <- hgruijter[[14]]$x
x3 <- hgruijter[[1]]$x
```

Table 24.2: Local Minima in De Gruijter Example

no	minimum	frequency
1	0.0222149	155
2	0.0222262	85
3	0.0222631	156
4	0.0223017	248
5	0.0232310	130
6	0.0244955	88
7	0.0245003	35
8	0.0246216	2
9	0.0254775	26
10	0.0254982	18
11	0.0261988	18
12	0.0273897	1
13	0.0274695	7
14	0.0297550	4
15	0.0303408	4
16	0.0315674	1
17	0.0315679	1
18	0.0356327	14
19	0.0362333	3
20	0.0364538	3
21	0.0375689	1

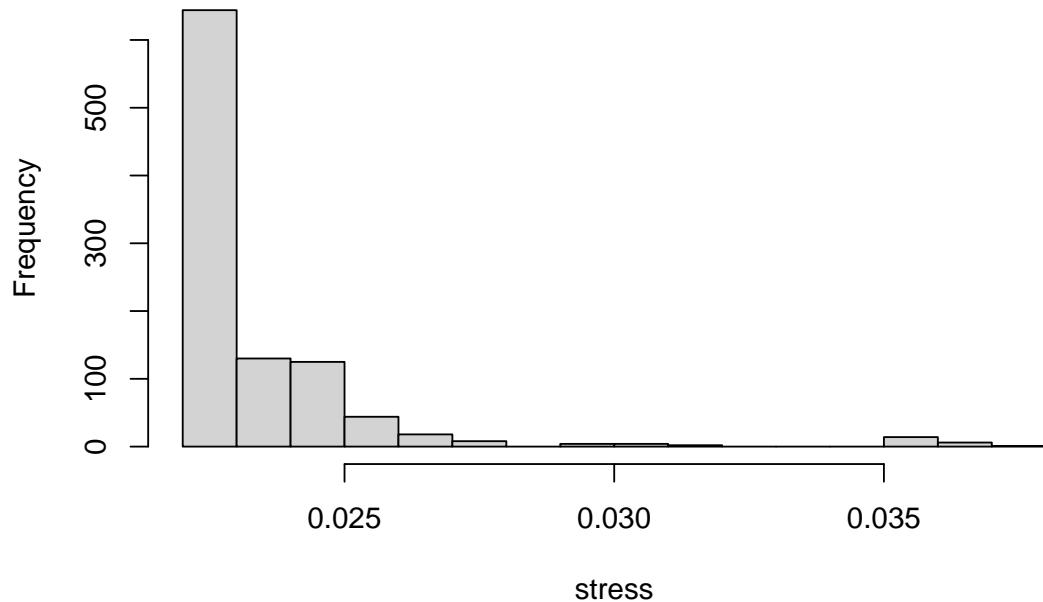


Figure 24.5: De Gruijter Stress Values

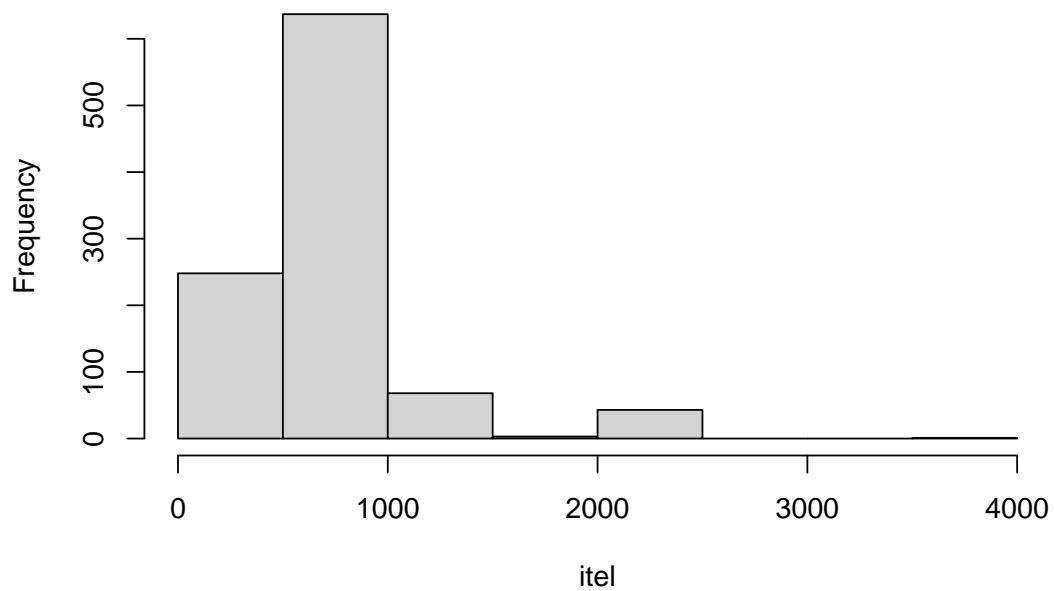


Figure 24.6: De Gruijter Iteration Count

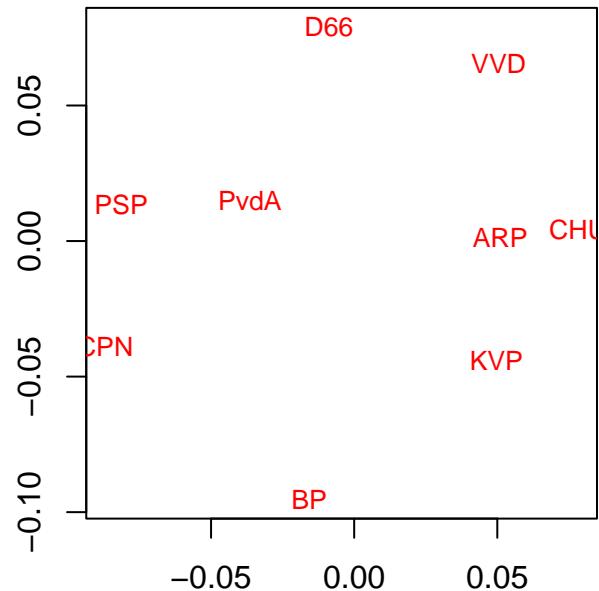


Figure 24.7: Largest Local Minimum

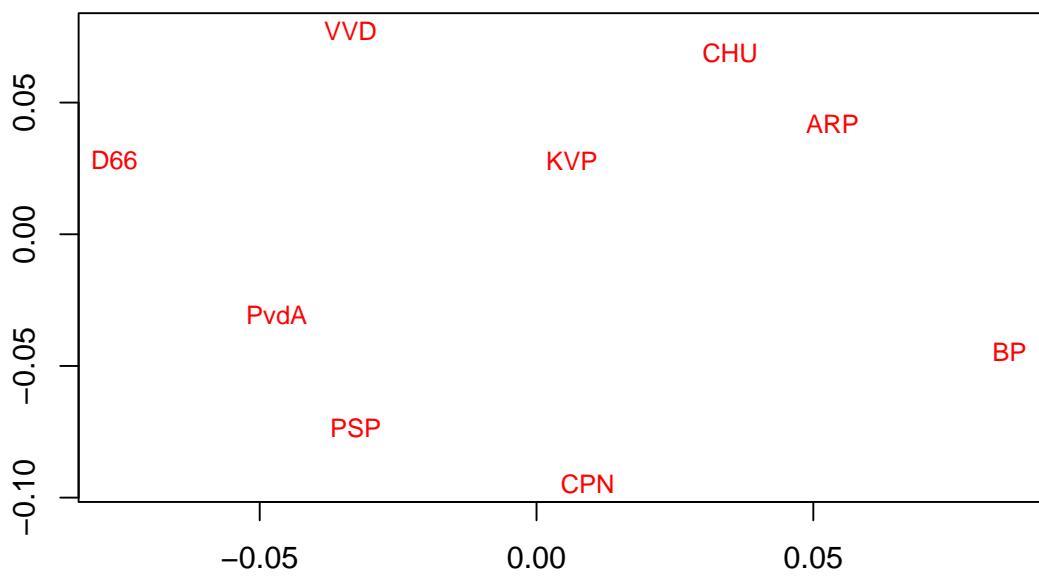


Figure 24.8: Another Local Minimum

## 24.2 Tunneling, Filling, Annealing, etc.

### 24.3 Cutting Planes

In cutting plane methods we approximate a non-polyhedral compact convex set  $\mathcal{C}$  by a sequence  $\mathcal{P}_n$  of convex polyhedra. Approximation is from the outside, i.e.  $\mathcal{C} \subset \mathcal{P}_n$ , strictly monotonic, i.e.  $\mathcal{P}_{n+1} \subset \mathcal{P}_n$ , and convergent, i.e.  $\lim_{n \rightarrow \infty} \mathcal{P}_n = \mathcal{C}$ . Under suitable conditions the maximum/minimum  $f_n^*$  of a function  $f$  on  $\mathcal{P}_n$  will converge to  $f^*$ , the maximum/minimum of  $f$  on  $\mathcal{C}$ .

If we are maximizing a convex function on  $\mathcal{C}$  then the maximum on the approximating polyhedron  $\mathcal{P}_n$  will be at one of the vertices  $x^*$ . If  $x^* \notin \mathcal{C}$  we cut it off by finding a hyperplane that separates  $x^*$  and  $\mathcal{C}$ . We can, for example, project  $x^*$  on  $\mathcal{C}$  and use the tangent hyperplane to  $\mathcal{C}$  in the projection  $\hat{x}$ . Define  $\mathcal{P}_{n+1} \cap \{x \mid a'x \leq b\}$ , with  $a'x = b$  the separating hyperplane that has  $a'x \leq b$  for each  $x \in \mathcal{C}$  and  $a'x^* > b$ .

The basic MDS problem can be reformulated as maximization of  $\rho(x)$  on the unit ball  $\eta^2(x) = x'x \leq 1$ . See sections 3.2 and 3.3 for the reformulation tools. The cutting plane method in the case of a ball is particularly simple, at least conceptually. If  $x^* \notin \mathcal{C}$  then its projection on the ball is  $\hat{x} = x^*/\|x^*\|$  and the tangent hyperplane in  $\hat{x}$  is  $\hat{x}'x \leq 1$ .

Computationally, however, matters are not so simple. The polyhedron  $\mathcal{P}_n$  is described by an increasing number of linear inequalities. Finding all vertices requires a non-trivial effort, and in the general case the method seems practical only for small or moderately small examples. In an actual implementation we would have to have a scheme for dropping or not adding redundant inequalities (that are implied by earlier inequalities) and a scheme for dropping inequalities generating vertices that can never be the global maximum. Such strategies will depend on the nature of the function  $f$  and on the convex set  $\mathcal{S}$ .

#### 24.3.1 On the Circle

For the circle the cutting plane method can be made very simple. Suppose we start with  $n$  distinct inner points  $x_1, \dots, x_n$  on the unit circle  $\mathcal{S}$ , ordered

clockwise so that  $x_1$  is at the top of the circle (high noon). Let  $\mathcal{Q}_n$  be their convex hull. Then  $\mathcal{Q}_n \subset \mathcal{S}$ . The tangent lines at  $x_i$  and  $x_{i+1}$  intersect outside the circle in an outer point  $y_i$ , with  $y_n$  the intersection of the tangents at  $x_n$  and  $x_1$ . Let  $\mathcal{P}_n$  be the convex hull of the  $y_i$ . Then  $\mathcal{Q}_n \subset \mathcal{S} \subset \mathcal{P}_n$  and thus

$$\max_{i=1}^n \rho(x_i) = \max_{x \in \mathcal{Q}_n} \rho(x) \leq \max_{x \in \mathcal{S}} \rho(x) \leq \max_{x \in \mathcal{P}_n} \rho(x) = \max_{i=1}^n \rho(y_i).$$

Thus we have easily computable lower and upper bounds for the global maximum of  $\rho$  on  $\mathcal{S}$ . The next step is to add the  $n$  projections  $y_i/\|y_i\|$  to the  $n$  inner points to have a new set of  $2n$  inner points. Then compute the corresponding  $2n$  outer points, and so on. After  $k$  steps we have  $2^k k_0$  inner and outer points, where  $k_0$  is the number of inner points we started with.

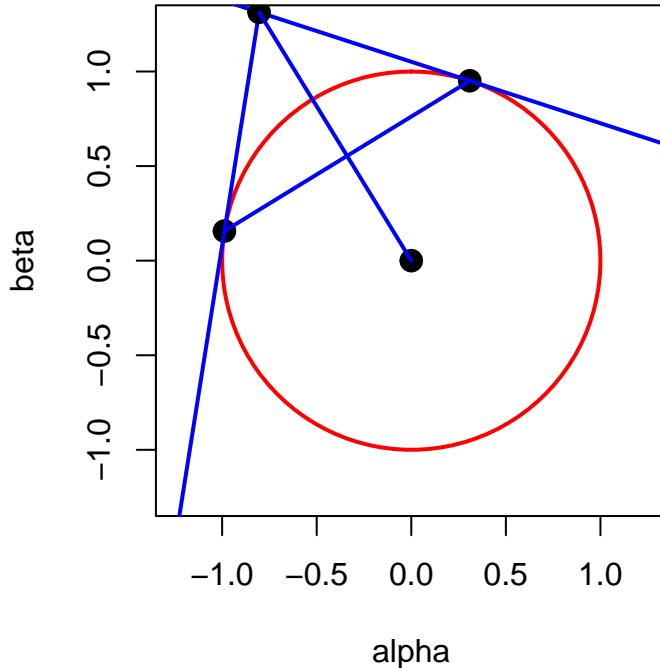


Figure 24.9: Circle Segment

As figure 24.9 shows, the outer point corresponding with two consecutive points on the circle lies on the perpendicular bisector of the line connecting the points. Using non-consecutive points will produce tangent lines which intersect farther away from the circle, and which consequently leads to a larger convex hull, and a worse approximation.

The next three figures illustrate the first iterations of the algorithm. We always start with  $n$  inner points equally distributed on the unit circle, in this case  $n = 4$ . The circle is in red, the convex hulls of the outer and inner points are in blue.

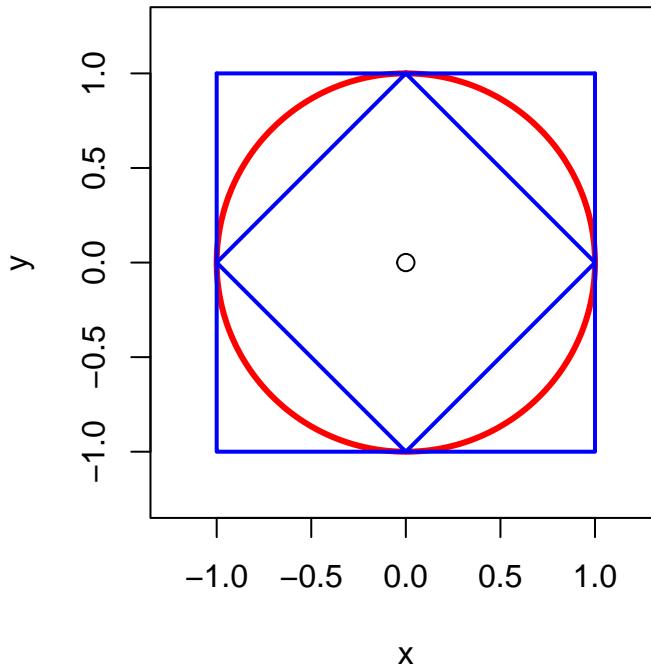


Figure 24.10: Starting Point

We see rapid convergence of the convex hulls to the circle. The figures also suggest an improvement of the method. Suppose  $\rho_0$  is a lower bound of the global minimum  $\rho_*$  equal to the largest  $\rho$  value of the inner points. Suppose an outer point has a  $\rho$  value less than or equal to  $\rho_0$ , consider the triangle with the outer point and the two inner points. All three vertices have a  $\rho$  value less than or equal to  $\rho_0$ , and because  $\rho$  is convex so have all points in the triangle, including a segment of the circle. Thus we can *phantom* that segment of the circle and create no new inner points there. If  $\rho_0$  get closer to  $\rho_*$  more and more segments of the circle are eliminated, which will presumably lead to faster computation. It seems advantageous to start with a value of  $\rho_0$  that is as large as possible, for example by using the value the smacof algorithm converges to. We can then use the inner and outer points to check if the  $\rho$  value is a global minimum.

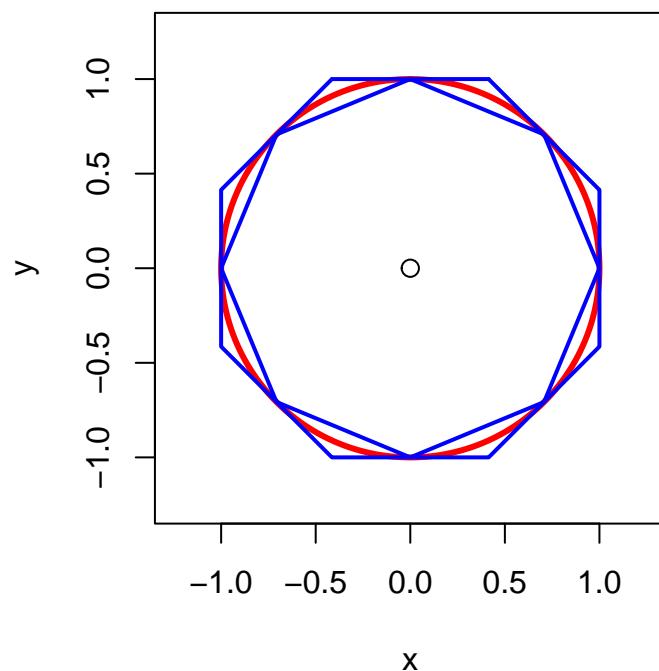


Figure 24.11: After First Iteration

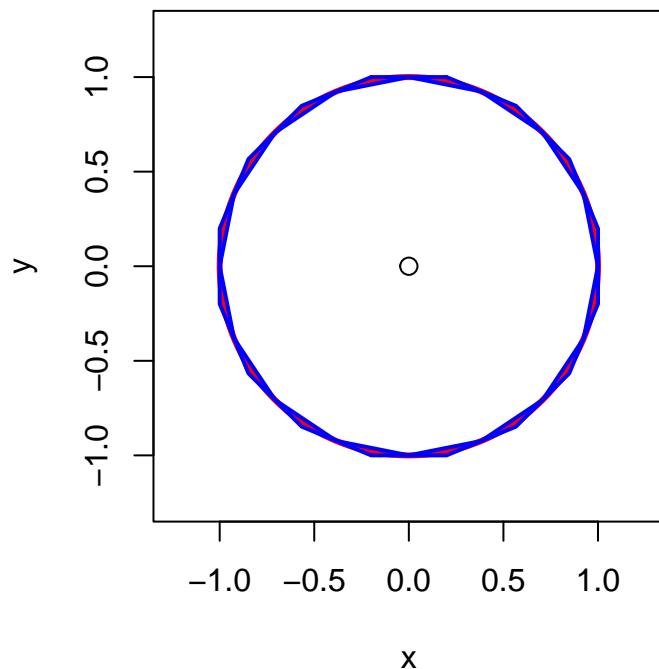


Figure 24.12: After Second Iteration

### 24.3.2 Cauchy Step Size

The standard smacof update of  $X$  (update method  $\text{up}_A$  of section 7.1) is  $\mathfrak{G}(X) = V^+B(X)X$ . The relaxed update is  $X(\lambda) := \lambda\mathfrak{G}(X) + (1 - \lambda)X$ . We usually choose  $\lambda = 2$ , which gives update method  $\text{up}_B$  of section 7.1.

The Cauchy or steepest descent update is  $X(\hat{\lambda})$ , with

$$\hat{\lambda} = \operatorname{argmin}_{\lambda \geq 0} \sigma(X(\lambda)). \quad (24.1)$$

There are some examples of the use of  $\hat{\lambda}$  in De Leeuw and Heiser (1980), but there the optimal step-size is computed by using constrained smacof iterations, which may actually take us to just a local minimum along the line. In this example we use our circle methodology to compute the global minimum.

Now  $\operatorname{tr} X'V\mathfrak{G}(X) = \rho(X)$  and thus if  $\operatorname{tr} X'VX = 1$  then  $Y = \mathfrak{G}(X) - \rho(X)X$  satisfies  $\operatorname{tr} X'VY = 0$  and  $\eta^2(Y) = \eta^2(\mathfrak{G}(X)) - \rho^2(X)$ . Normalize  $Y$  such that  $\operatorname{tr} Y'VY = 1$ , and now maximize  $\rho$  over  $\alpha$  and  $\beta$ , i.e. in configuration space maximize  $\rho(\alpha X + \beta Y)$ , where  $\alpha^2 + \beta^2 = 1$ .

In the example we choose  $X$  to be a  $10 \times 2$  matrix filled with random standard normals, and we start with 10 inner points on the circle. The iterations until convergence are as follows.

<code>## itel</code>	1	vertices	10	innermax	0.96683250	outermax	1.01451883
<code>## itel</code>	2	vertices	20	innermax	0.96683250	outermax	0.97744943
<code>## itel</code>	3	vertices	40	innermax	0.96683250	outermax	0.97008052
<code>## itel</code>	4	vertices	80	innermax	0.96709009	outermax	0.96794626
<code>## itel</code>	5	vertices	160	innermax	0.96720000	outermax	0.96739328
<code>## itel</code>	6	vertices	320	innermax	0.96720681	outermax	0.96726518
<code>## itel</code>	7	vertices	640	innermax	0.96721856	outermax	0.96722816
<code>## itel</code>	8	vertices	1280	innermax	0.96721856	outermax	0.96722140
<code>## itel</code>	9	vertices	2560	innermax	0.96721856	outermax	0.96721949
<code>## itel</code>	10	vertices	5120	innermax	0.96721876	outermax	0.96721890
<code>## itel</code>	11	vertices	10240	innermax	0.96721876	outermax	0.96721880
<code>## itel</code>	12	vertices	20480	innermax	0.96721876	outermax	0.96721877
<code>## itel</code>	13	vertices	40960	innermax	0.96721876	outermax	0.96721877

The optimal  $\alpha$  and  $\beta$  are -0.5464817, -0.837471. In configuration space this translates to 1.7919402  $X +$ -2.6770034  $\mathfrak{G}(X)$ .

### 24.3.3 Balls

In dimension  $p > 2$ , where  $\rho$  must be maximized on the unit ball in  $\mathbb{R}^p$ , matters are not so simple any more. There is no single compelling natural ordering of the points on the sphere or hypersphere, and thus we have to improvise more. We would like to maintain both upper and lower bounds for the global minimum that both keep improving in every iteration.

#### 24.3.3.1 Outer Approximation

Let's first discuss a possible initial set of inner and outer points that are more or less regularly spaced inside or outside the unit sphere. For the inner points we can use the vertices of the cross-polytope (or the  $\ell_1$ -ball), which is the set of all  $x$  in  $\mathbb{R}^n$  with  $\sum_{i=1}^n |x_i| \leq 1$ . If  $|x_i| \leq 1$  then  $x_i^2 \leq |x_i|$  with equality iff  $x_i \in \{-1, 0, 1\}$ . Thus  $x'x \leq \sum_{i=1}^n |x_i|$  and  $x'x = \sum_{i=1}^n |x_i| = 1$  iff exactly one of  $x_i$  is  $\pm 1$ , i.e. there are  $2^n$  inner points on the sphere.

For the outer points we choose the vertices of  $\max_{i=1}^n |x_i| \leq 1$ , or equivalently  $-1 \leq x_i \leq +1$  for all  $i$ . Thus there are  $2^n$  vertices which have  $x_i = \pm 1$  for all  $i$ .

## 24.4 Distance Smoothing

$$d_{ij}(X, \epsilon) := \sqrt{d_{ij}^2(X) + \epsilon^2}$$

$$d_{ij}^\epsilon(X) := \sqrt{d_{ij}^2(X) + \epsilon^2}$$

$$\mathcal{D}d_{ij}^\epsilon(X) = \frac{1}{d_{ij}^\epsilon(X)} A_{ij} X$$

$$\nabla \sigma_\epsilon(X) = 2(V - B_\epsilon(X))X$$

$$\mathcal{D}^2 \sigma_\epsilon(X) =$$

**Theorem 24.1.** *If  $\epsilon \geq \max_{i,j} \delta_{ij}$  then  $B(X_\epsilon) \lesssim V$  and thus  $\mathcal{D}^2(X_\epsilon) \gtrsim 0$  for all  $X$  and  $\sigma_\epsilon$  is convex.*

$$\begin{aligned}\mathcal{D}_1 d_{ij}(X, \epsilon) &= \frac{1}{d_{ij}(X, \epsilon)} A_{ij} X \\ \nabla \sigma_\epsilon(X) &= 2 \left( V - \sum_{1 \leq i < j \leq n} w_{ij} \frac{\delta_{ij}}{d_{ij}(X_\epsilon)} A_{ij} \right) X \\ \nabla^2 \sigma_\epsilon(X) &= 2V - \sum_{1 \leq i < j \leq n} w_{ij} \frac{\delta_{ij}}{d_{ij}(X_\epsilon)} A_{ij} X\end{aligned}$$

## 24.5 Penalizing Dimensions

In Shepard (1962a) and Shepard (1962b) an NMDS technique is developed that minimizes a loss function over configurations in full dimensionality  $n - 1$ . In that sense the technique is similar to FDS. Shepard's iterative process, however, aims to maintain monotonicity between distances and dissimilarities and at the same time concentrate as much of the variation as possible in a small number of dimensions (De Leeuw (2017f)).

Let us explore the idea of concentrating variation in  $p < n - 1$  dimensions, but use an approach which is quite different from the one used by Shepard. We remain in the FDS framework, but we aim for solutions in  $p < n - 1$  dimensions by penalizing  $n - p$  dimensions of the full configuration, using the classical Courant quadratic penalty function.

Partition a full configuration  $Z = [X \mid Y]$ , with  $X$  of dimension  $n \times p$  and  $Y$  of dimension  $n \times (n - p)$ . Then

$$\sigma(Z) = 1 - \mathbf{tr} X' B(Z) X - \mathbf{tr} Y' B(Z) Y + \frac{1}{2} \mathbf{tr} X' V X + \frac{1}{2} \mathbf{tr} Y' V Y. \quad (24.2)$$

Also define the *penalty term*

$$\tau(Y) = \frac{1}{2} \mathbf{tr} Y' V Y, \quad (24.3)$$

and *penalized stress*

$$\pi(Z, \lambda) = \sigma(Z) + \lambda \tau(Y). \quad (24.4)$$

Our proposed method is to minimize penalized stress over  $Z$  for a sequence of values  $0 = \lambda_1 < \lambda_2 < \dots < \lambda_m$ . For  $\lambda = 0$  this is simply the FDS problem, for

which we know we can compute the global minimum. For fixed  $0 < \lambda < +\infty$  this is a Penalized FDS or PFDS problem. PFDS problems with increasing values of  $\lambda$  generate a *trajectory*  $Z(\lambda)$  in configuration space.

The general theory of exterior penalty functions, which we review in section XX of this paper, shows that increasing  $\lambda$  leads to an increasing sequence of stress values  $\sigma$  and a decreasing sequence of penalty terms  $\tau$ . If  $\lambda \rightarrow +\infty$  we approximate the global minimum of the FDS problem with  $Z$  of the form  $Z = [X \mid 0]$ , i.e. of the pMDS problem. This assumes we do actually compute the global minimum for each value of  $\lambda$ , which we hope we can do because we start at the FDS global minimum, and we slowly increase  $\lambda$ . There is also a local version of the exterior penalty result, which implies that  $\lambda \rightarrow \infty$  takes us to a local minimum of pMDS, so there is always the possibility of taking the wrong trajectory to a local minimum of pMDS.

### 24.5.1 Local Minima

The stationary equations of the PFDS problem are solutions to the equations

$$(V - B(Z))X = 0, \quad (24.5)$$

$$((1 + \lambda)V - B(Z))Y = 0. \quad (24.6)$$

We can easily relate stationary points and local minima of the FDS and PFDS problem.

**Theorem 24.2.** 1: *If  $X$  is a stationary point of the pMDS problem then  $Z = [X \mid 0]$  is a stationary point of the PFDS problem, no matter what  $\lambda$  is.*

2: *If  $Z = [X \mid 0]$  is a local minimum of the PFDS problem then  $X$  is a local minimum of pMDS and  $(1 + \lambda)V - B(X) \gtrsim 0$ , or  $\lambda \geq \|V^+B(X)\|_\infty - 1$ , with  $\|\bullet\|_\infty$  the spectral radius (largest eigenvalue).*

*Proof.* Part 1 follows by simple substitution in the stationary equations.

Part 2 follows from the expansion for  $Z = [X + \epsilon P \mid \epsilon Q]$ .

$$\begin{aligned} \pi(Z) &= \pi(X) + \epsilon \operatorname{tr} P' D \sigma(X) + \\ &\quad + \frac{1}{2} \epsilon^2 D^2 \sigma(X)(P, P) + \frac{1}{2} \epsilon^2 \operatorname{tr} Q' ((1 + \lambda)V - B(X))Q + o(\epsilon^2). \end{aligned} \quad (24.7)$$

At a local minimum we must have  $\mathcal{D}\sigma(X) = 0$  and  $\mathcal{D}^2\sigma(X)(P, P) \gtrsim 0$ , which are the necessary conditions for a local minimum of pMDS. We also must have  $((1 + \lambda)V - B(X)) \gtrsim 0$ .  $\square$

Note that the conditions in part 2 of theorem 24.2 are also sufficient for PFDS to have a local minimum at  $[X \mid 0]$ , provided we eliminate translational and rotational indeterminacy by a suitable reparametrization, as in De Leeuw (1993).

### 24.5.2 Algorithm

The smacof algorithm for penalized stress is a small modification of the unpenalized FDS algorithm (ref). We start our iterations for  $\lambda_j$  with the solution for  $\lambda_{j-1}$  (the starting solution for  $\lambda_1 = 0$  can be completely arbitrary). The update rules for fixed  $\lambda$  are

$$X^{(k+1)} = V^+ B(Z^{(k)}) X^{(k)}, \quad (24.8)$$

$$Y^{(k+1)} = \frac{1}{1 + \lambda} V^+ B(Z^{(k)}) Y^{(k)}. \quad (24.9)$$

Thus we compute the FDS update  $Z^{(k+1)} = V^+ B(Z^{(k)}) Z^{(k)}$  and then divide the last  $n - p$  columns by  $1 + \lambda$ .

Code is in the appendix. Let us analyze a number of examples.

### 24.5.3 Examples

This section has a number of two-dimensional and a number of one-dimensional examples. The one-dimensional examples are of interest, because of the documented large number of local minima of stress in the one-dimensional case, and the fact that for small and medium  $n$  exact solutions are available (for example, De Leeuw (2005c)). By default we use `seq(0, 1, length = 101)` for  $\lambda$  in most examples, but for some of them we dig a bit deeper and use longer sequences with smaller increments.

If for some value of  $\lambda$  the penalty term drops below the small cutoff  $\gamma$ , for example  $10^{-10}$ , then there is no need to try larger values of  $\lambda$ , because they

will just repeat the same result. We hope that result is the global minimum of the 2MDS problem.

The output for each example is a table in which we give, the minimum value of stress, the value of the penalty term at the minimum, the value of  $\lambda$ , and the number of iterations needed for convergence. Typically we print for the first three, the last three, and some regularly spaced intermediate values of  $\lambda$ . Remember that the stress values increase with increasing  $\lambda$ , and the penalty values decrease.

For two-dimensional examples we plot all two-dimensional configurations, after rotating to optimum match (using the function `matchMe()` from the appendix). We connect corresponding points for different values of  $\lambda$ . Points corresponding to the highest value of  $\lambda$  are labeled and have a different plot symbol. For one-dimensional examples we put `1:n` on the horizontal axes and plot the single dimension on the vertical axis, again connecting corresponding points. We label the points corresponding with the highest value of  $\lambda$ , and draw horizontal lines through them to more clearly show their order on the dimension.

The appendix also has code for the function `checkUni()`, which we have used to check the solutions in the one dimensional case are indeed local minima. The function checks the necessary condition for a local minimum  $x = V^+u$ , with

$$u_i = \sum_{j=1}^n w_{ij} \delta_{ij} \operatorname{sign}(x_i - x_j).$$

It should be emphasized that all examples are just meant to study convergence of penalized FDS. There is no interpretation of the MDS results

#### 24.5.3.1 Chi Squares

In this example, of order 10, the  $\delta_{ij}$  are independent draws from a chi-square distribution with two degrees of freedom. There is no structure in this example, everything is random.

```
itel 198 lambda 0.000000 stress 0.175144 penalty 0.321138
itel 5 lambda 0.010000 stress 0.175156 penalty 0.027580
itel 3 lambda 0.020000 stress 0.175187 penalty 0.025895
```

```

itel 1 lambda 0.100000 stress 0.175914 penalty 0.015172
itel 1 lambda 0.200000 stress 0.177666 penalty 0.004941
itel 4 lambda 0.300000 stress 0.178912 penalty 0.000088
itel 6 lambda 0.310000 stress 0.178933 penalty 0.000020
itel 20 lambda 0.320000 stress 0.178939 penalty 0.000000

```

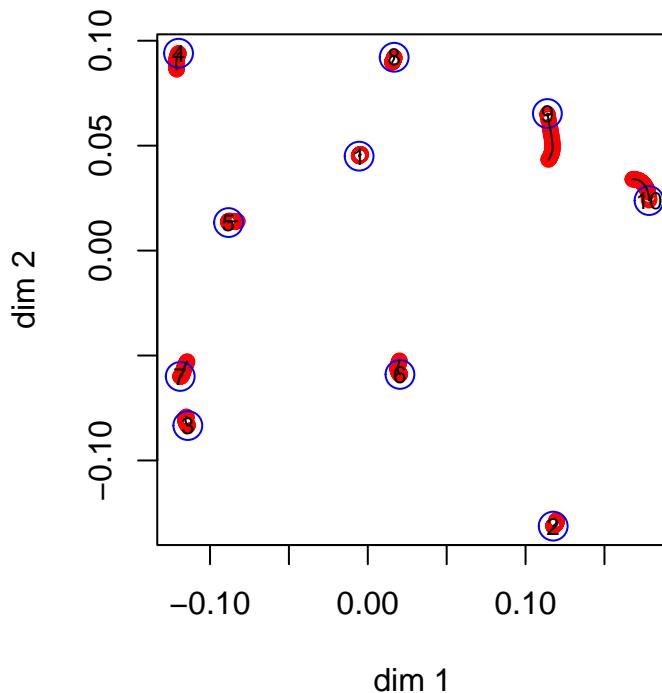


Figure 24.13: 10 Chi Squares

It seems that in this example the first two dimensions of FDS are already close to optimal for 2MDS. This is because the Gower rank of the dissimilarities is only three (or maybe four, the fourth singular value of the FDS solution  $Z$  is very small).

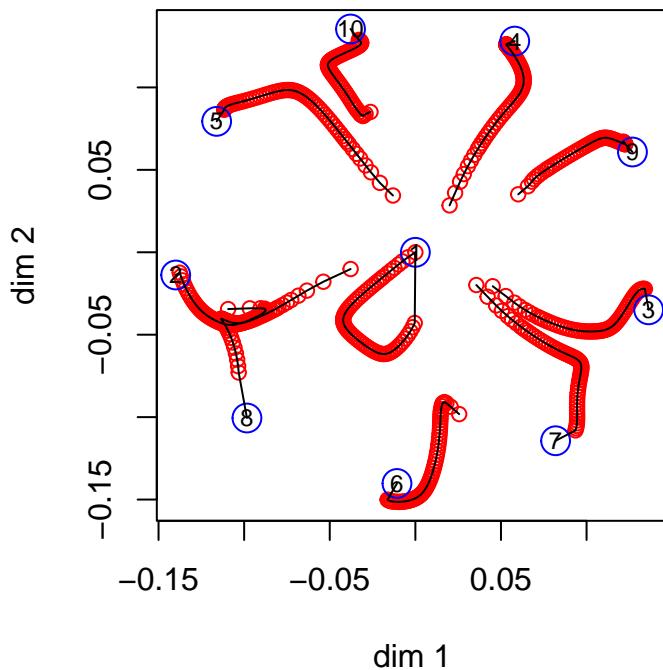
### 24.5.3.2 Regular Simplex

The regular simplex has all dissimilarities equal to one. We use an example with  $n = 10$ , for which the global minimum (as far as we know) of pMDS with  $p = 2$  is a configuration with nine points equally spaced on a circle and one point in the center.

```

itel 1 lambda 0.000000 stress 0.000000 penalty 0.400000
itel 7 lambda 0.010000 stress 0.000101 penalty 0.375653
itel 5 lambda 0.020000 stress 0.000422 penalty 0.360483
itel 1 lambda 0.100000 stress 0.008849 penalty 0.258090
itel 1 lambda 0.200000 stress 0.032243 penalty 0.149427
itel 1 lambda 0.300000 stress 0.059088 penalty 0.079592
itel 1 lambda 0.400000 stress 0.079534 penalty 0.043250
itel 1 lambda 0.500000 stress 0.095361 penalty 0.020895
itel 1 lambda 0.600000 stress 0.105667 penalty 0.006921
itel 1 lambda 0.610000 stress 0.106337 penalty 0.005862
itel 1 lambda 0.620000 stress 0.106951 penalty 0.004879
itel 105 lambda 0.630000 stress 0.109880 penalty 0.000000

```



Next, we look at the regular simplex with  $n = 4$ , for which the global minimum has four points equally spaced on a circle (i.e. in the corners of a square). We use `seq(0, 1, length = 101)` for the  $\lambda$  sequence.

```

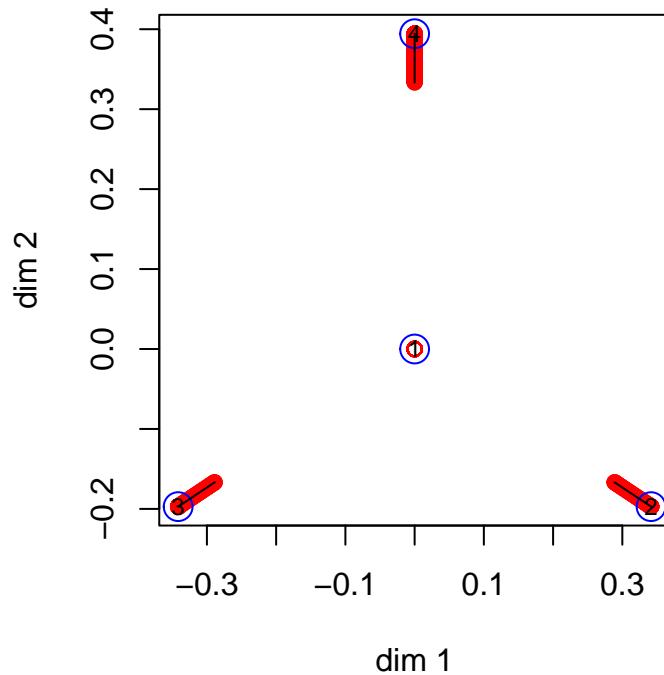
itel 1 lambda 0.000000 stress 0.000000 penalty 0.250000

```

```

itel 2 lambda 0.010000 stress 0.000035 penalty 0.162295
itel 1 lambda 0.020000 stress 0.000122 penalty 0.158591
itel 1 lambda 0.100000 stress 0.003808 penalty 0.122344
itel 1 lambda 0.200000 stress 0.014089 penalty 0.084111
itel 1 lambda 0.300000 stress 0.028044 penalty 0.053366
itel 1 lambda 0.400000 stress 0.043331 penalty 0.028965
itel 1 lambda 0.500000 stress 0.056851 penalty 0.011482
itel 1 lambda 0.600000 stress 0.064718 penalty 0.002471
itel 1 lambda 0.700000 stress 0.066799 penalty 0.000203
itel 1 lambda 0.800000 stress 0.066982 penalty 0.000005
itel 1 lambda 0.820000 stress 0.066985 penalty 0.000002
itel 1 lambda 0.830000 stress 0.066986 penalty 0.000001
itel 1 lambda 0.840000 stress 0.066986 penalty 0.000001

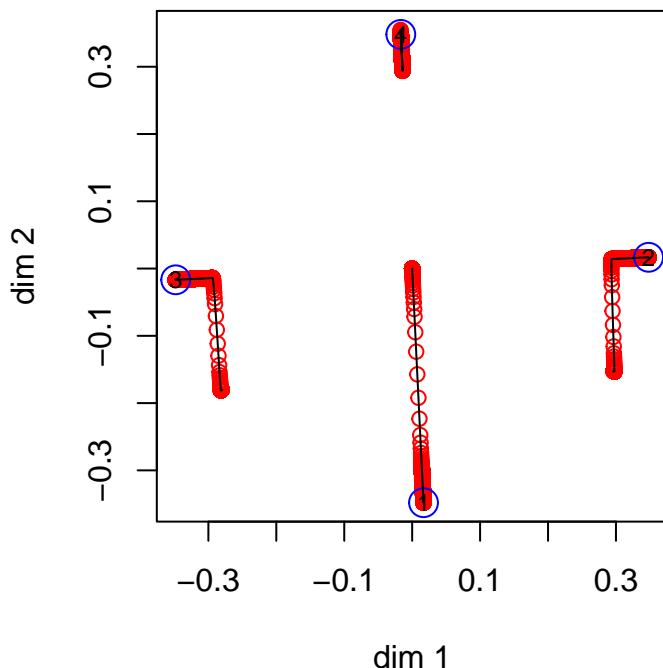
```



The solution converges to an equilateral triangle with the fourth point in the centroid. This is a local minimum. What basically happens is that the first two dimensions of the FDS solution are too close to the local minimum. Or, what amounts to the same thing, the Gower rank is too large (it is  $n - 1$  for a regular simplex) , there is too much variation in the higher dimensions, and

as a consequence the first two dimensions of FDS are a bad 2MDS solution. We try to repair this by refining the trajectory, using `seq(0, 1, 10001)`.

```
itel 1 lambda 0.000000 stress 0.000000 penalty 0.250000
itel 2 lambda 0.000100 stress 0.000000 penalty 0.166622
itel 1 lambda 0.000200 stress 0.000000 penalty 0.166583
itel 1 lambda 0.202200 stress 0.028595 penalty 0.000001
itel 1 lambda 0.202300 stress 0.028595 penalty 0.000001
itel 1 lambda 0.202400 stress 0.028595 penalty 0.000001
```



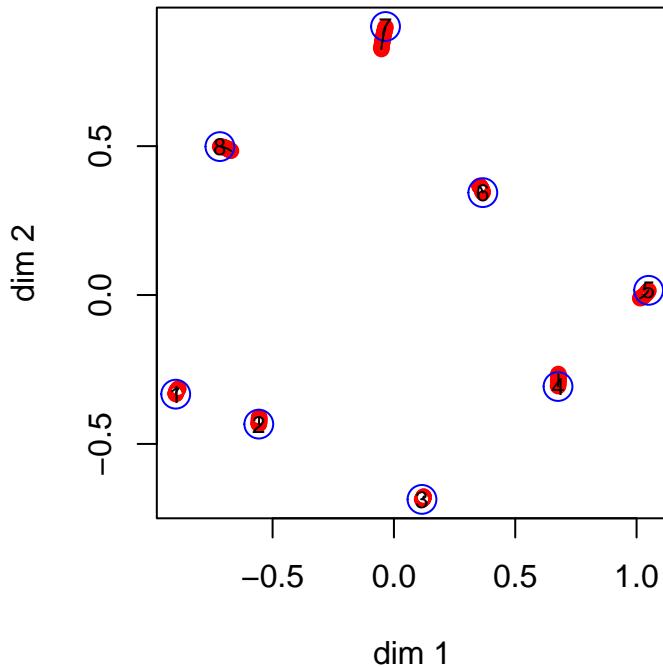
Now the trajectories move us from what starts out similar to an equilateral triangle to the corners of the square, and thus we do find the global minimum in this way. It is remarkable that we manage to find the square even when we start closer to the triangle with midpoint.

#### 24.5.3.3 Intelligence

These are correlations between eight intelligence tests, taken from the smacof package. We convert to dissimilarities by taking the negative logarithm of the

correlations. As in the chi-square example, the FDS and the 2MDS solution are very similar and the PMDS trajectories are short.

```
itel 2951 lambda 0.000000 stress 0.107184 penalty 7.988384
itel 7 lambda 0.010000 stress 0.107560 penalty 0.685654
itel 4 lambda 0.020000 stress 0.108528 penalty 0.628538
itel 3 lambda 0.030000 stress 0.110045 penalty 0.573208
itel 3 lambda 0.040000 stress 0.112449 penalty 0.510730
itel 2 lambda 0.050000 stress 0.114714 penalty 0.464650
itel 2 lambda 0.060000 stress 0.117623 penalty 0.415037
itel 2 lambda 0.070000 stress 0.121095 penalty 0.364536
itel 2 lambda 0.080000 stress 0.125010 penalty 0.315023
itel 2 lambda 0.090000 stress 0.129226 penalty 0.267831
itel 2 lambda 0.100000 stress 0.133589 penalty 0.223898
itel 2 lambda 0.110000 stress 0.137944 penalty 0.183868
itel 3 lambda 0.120000 stress 0.143921 penalty 0.133739
itel 2 lambda 0.130000 stress 0.147473 penalty 0.106166
itel 4 lambda 0.140000 stress 0.153215 penalty 0.064499
itel 4 lambda 0.150000 stress 0.157159 penalty 0.037735
itel 9 lambda 0.160000 stress 0.161434 penalty 0.010337
itel 72 lambda 0.170000 stress 0.163122 penalty 0.000000
```



The singular values of the FDS solution are 1.78e+00, 1.36e+00, 5.94e-01, 1.47e-01, 3.29e-03, 2.39e-16, 1.15e-16, 9.35e-18, which shows that the Gower rank is probably five, but approximately two.

#### 24.5.3.4 Countries

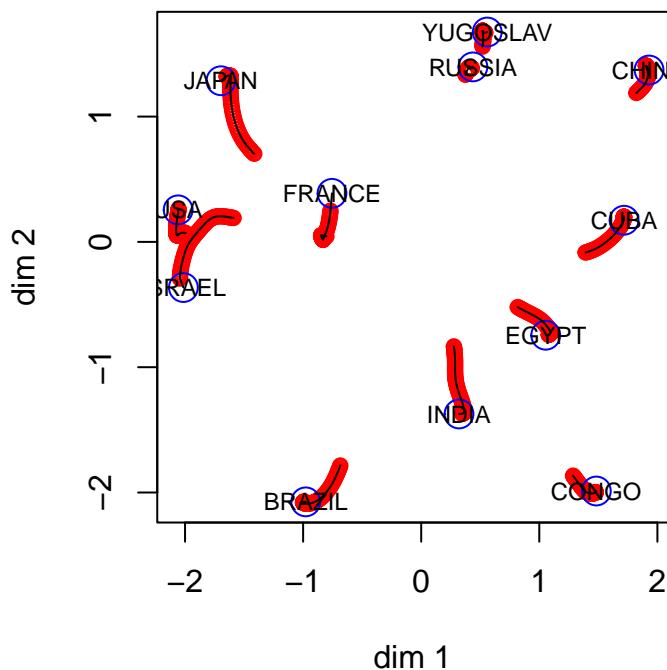
This is the wish dataset from the smacof package, with similarities between 12 countries. They are converted to dissimilarities by subtracting each of them from seven.

```
itel 1381 lambda 0.000000 stress 4.290534 penalty 98.617909
itel 4 lambda 0.010000 stress 4.301341 penalty 36.137074
itel 3 lambda 0.020000 stress 4.336243 penalty 34.389851
itel 1 lambda 0.100000 stress 5.187917 penalty 23.300775
itel 1 lambda 0.200000 stress 7.539228 penalty 11.543635
itel 1 lambda 0.300000 stress 9.901995 penalty 4.963372
itel 1 lambda 0.400000 stress 11.523357 penalty 1.859569
itel 1 lambda 0.500000 stress 12.391692 penalty 0.556411
```

```

itel 1 lambda 0.590000 stress 12.696493 penalty 0.080144
itel 1 lambda 0.600000 stress 12.708627 penalty 0.060113
itel 100 lambda 0.610000 stress 12.738355 penalty 0.000000

```



The singular values of the FDS solution are 4.20e+00, 3.71e+00, 2.67e+00, 1.80e+00, 1.33e+00, 6.64e-01, 5.97e-04, 6.75e-16, 4.70e-16, 2.38e-16, 1.14e-16, 1.98e-17, and the Gower rank is six or seven.

#### 24.5.3.5 Dutch Political Parties

In 1967 one hundred psychology students at Leiden University judged the similarity of nine Dutch political parties, using the complete method of triads (De Gruijter (1967)). Data were aggregated and converted to dissimilarities. We first print the matrix of dissimilarities.

```

KVP PvdA VVD ARP CHU CPN PSP BP D66
KVP +0.000 +0.209 +0.196 +0.171 +0.179 +0.281 +0.250 +0.267 +0.230
PvdA +0.209 +0.000 +0.250 +0.210 +0.231 +0.190 +0.171 +0.269 +0.204

```

```

VVD +0.196 +0.250 +0.000 +0.203 +0.185 +0.302 +0.281 +0.257 +0.174
ARP +0.171 +0.210 +0.203 +0.000 +0.119 +0.292 +0.250 +0.271 +0.228
CHU +0.179 +0.231 +0.185 +0.119 +0.000 +0.290 +0.263 +0.259 +0.225
CPN +0.281 +0.190 +0.302 +0.292 +0.290 +0.000 +0.152 +0.236 +0.276
PSP +0.250 +0.171 +0.281 +0.250 +0.263 +0.152 +0.000 +0.256 +0.237
BP +0.267 +0.269 +0.257 +0.271 +0.259 +0.236 +0.256 +0.000 +0.274
D66 +0.230 +0.204 +0.174 +0.228 +0.225 +0.276 +0.237 +0.274 +0.000

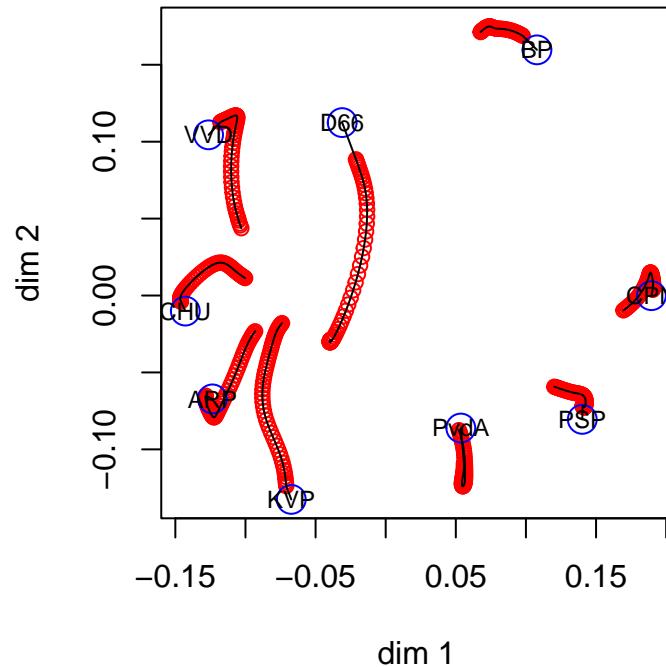
```

The trajectories from FDS to 2MDS show some clear movement, especially of the D'66 party, which was new at the time.

```

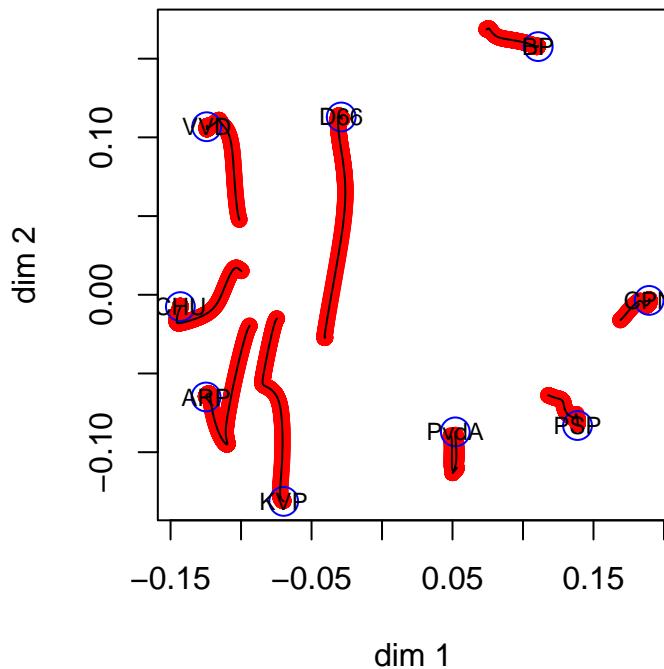
itel 223 lambda 0.000000 stress 0.000000 penalty 0.414526
itel 5 lambda 0.010000 stress 0.000061 penalty 0.196788
itel 2 lambda 0.020000 stress 0.000199 penalty 0.190472
itel 1 lambda 0.100000 stress 0.004399 penalty 0.136576
itel 1 lambda 0.200000 stress 0.015811 penalty 0.075466
itel 1 lambda 0.300000 stress 0.028235 penalty 0.036636
itel 1 lambda 0.400000 stress 0.038275 penalty 0.012608
itel 1 lambda 0.500000 stress 0.043644 penalty 0.002156
itel 1 lambda 0.520000 stress 0.044091 penalty 0.001324
itel 1 lambda 0.530000 stress 0.044253 penalty 0.001019
itel 277 lambda 0.540000 stress 0.044603 penalty 0.000000

```



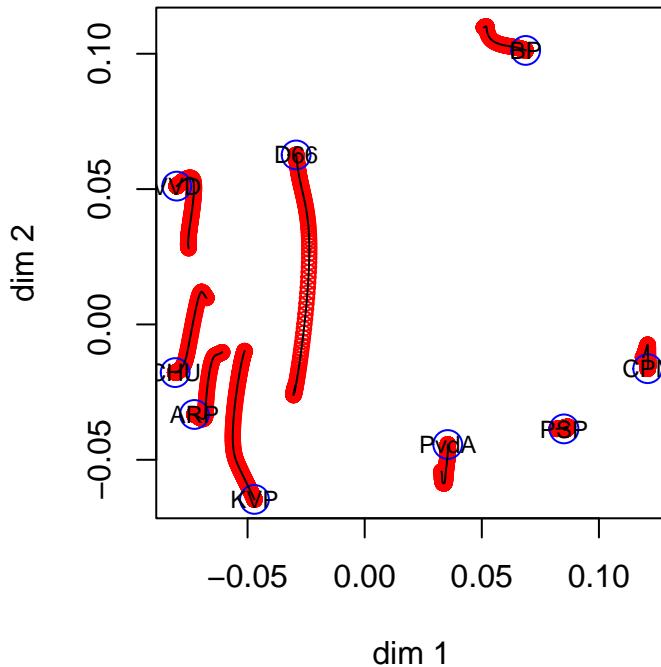
There seems to be some bifurcation going on at the end, so we repeat the analysis using `seq(0, 1, length = 1001)` for  $\lambda$ . The results turn out to be basically the same.

```
itel 223 lambda 0.000000 stress 0.000000 penalty 0.414526
itel 4 lambda 0.001000 stress 0.000001 penalty 0.204225
itel 2 lambda 0.002000 stress 0.000002 penalty 0.203535
itel 1 lambda 0.468000 stress 0.044604 penalty 0.000001
itel 1 lambda 0.469000 stress 0.044604 penalty 0.000000
itel 166 lambda 0.470000 stress 0.044603 penalty 0.000000
```



The singular values of the FDS solution are 2.95e-01, 2.10e-01, 1.89e-01, 1.34e-01, 1.16e-01, 1.06e-01, 8.61e-02, 7.06e-02, 3.98e-18, and the Gower rank is probably eight. This is mainly because these data, being averages, regress to the mean and thus have a substantial additive constant. If we repeat the analysis after subtracting .1 from all dissimilarities we get basically the same solution, but with somewhat smoother trajectories.

```
itel 511 lambda 0.000000 stress 0.000176 penalty 0.150789
itel 2 lambda 0.001000 stress 0.000176 penalty 0.037759
itel 2 lambda 0.002000 stress 0.000176 penalty 0.037619
itel 1 lambda 0.370000 stress 0.007642 penalty 0.000000
itel 1 lambda 0.371000 stress 0.007642 penalty 0.000000
itel 1 lambda 0.372000 stress 0.007642 penalty 0.000000
```



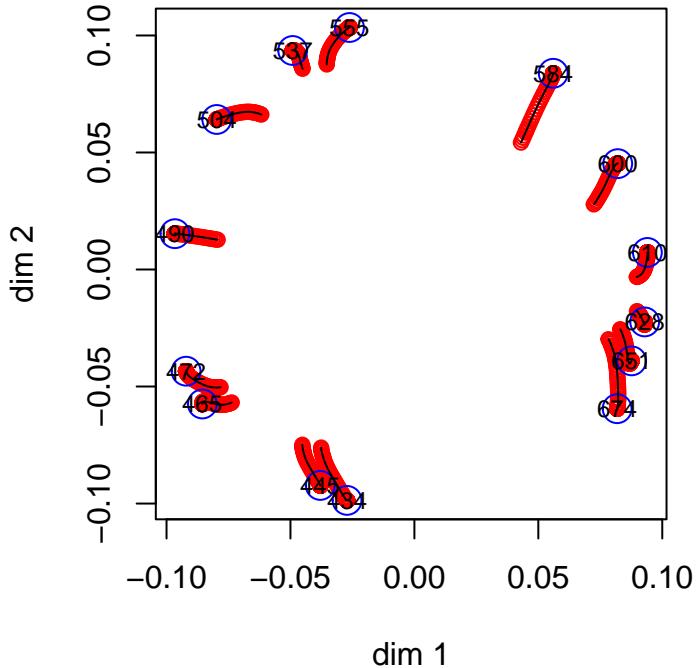
Now the singular values of the FDS solution are 2.05e-01, 1.34e-01, 1.11e-01, 6.03e-02, 3.11e-02, 3.97e-04, 1.18e-07, 4.55e-12, 8.75e-18, and the approximate Gower rank is more like five or six.

#### 24.5.3.6 Ekman

The next example analyzes dissimilarities between 14 colors, taken from Ekman (1954). The original similarities  $s_{ij}$ , averaged over 31 subjects, were transformed to dissimilarities by  $\delta_{ij} = 1 - s_{ij}$ .

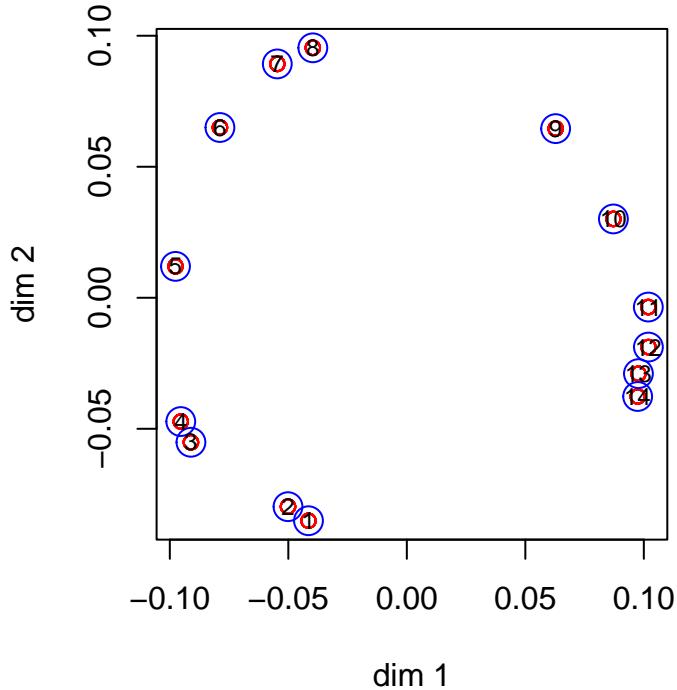
```
itel 1482 lambda 0.000000 stress 0.000088 penalty 0.426110
itel 5 lambda 0.010000 stress 0.000132 penalty 0.118988
itel 3 lambda 0.020000 stress 0.000253 penalty 0.112777
itel 1 lambda 0.100000 stress 0.003195 penalty 0.070791
itel 1 lambda 0.200000 stress 0.010778 penalty 0.024407
itel 1 lambda 0.300000 stress 0.016125 penalty 0.003230
itel 1 lambda 0.400000 stress 0.017142 penalty 0.000165
itel 4 lambda 0.500000 stress 0.017213 penalty 0.000000
```

```
itel 1 lambda 0.600000 stress 0.017213 penalty 0.000000
itel 1 lambda 0.610000 stress 0.017213 penalty 0.000000
itel 1 lambda 0.620000 stress 0.017213 penalty 0.000000
itel 1 lambda 0.630000 stress 0.017213 penalty 0.000000
```



If we transform the Ekman similarities by  $\delta_{ij} = (1 - s_{ij})^3$  then its is known (De Leeuw (2016c)) that the Gower rank is equal to two. Thus the FDS solution has rank 2, and the 2MDS solution is the global minimum.

```
itel 99 lambda 0.000000 stress 0.011025 penalty 0.433456
itel 1 lambda 0.010000 stress 0.011025 penalty 0.000000
itel 1 lambda 0.020000 stress 0.011025 penalty 0.000000
itel 1 lambda 0.100000 stress 0.011025 penalty 0.000000
itel 1 lambda 0.110000 stress 0.011025 penalty 0.000000
itel 1 lambda 0.120000 stress 0.011025 penalty 0.000000
itel 1 lambda 0.130000 stress 0.011025 penalty 0.000000
```

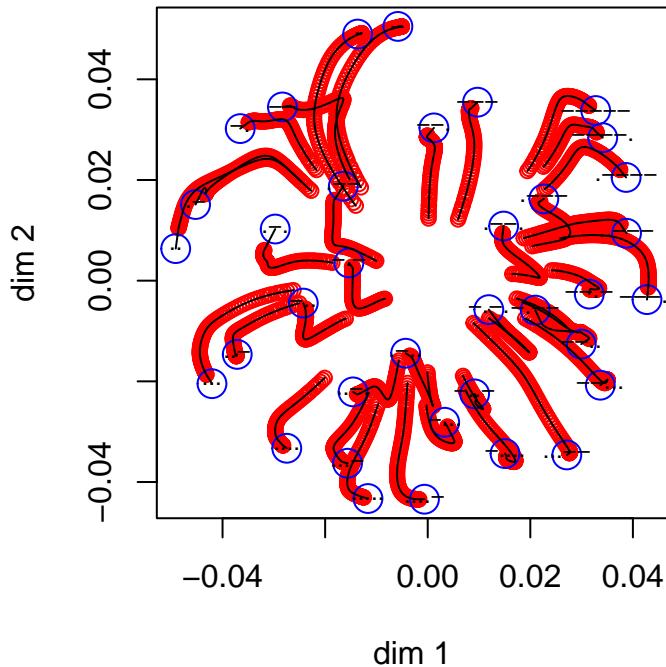


#### 24.5.3.7 Morse in Two

Next, we use dissimilarities between 36 Morse code signals (Rothkopf (1957)). We used the symmetrized version `morse` from the `smacof` package (De Leeuw and Mair (2009)).

```
itel 1461 lambda 0.000000 stress 0.000763 penalty 0.472254
itel 6 lambda 0.010000 stress 0.000858 penalty 0.322181
itel 4 lambda 0.020000 stress 0.001147 penalty 0.308335
itel 1 lambda 0.100000 stress 0.008576 penalty 0.216089
itel 1 lambda 0.200000 stress 0.028903 penalty 0.119364
itel 1 lambda 0.300000 stress 0.051285 penalty 0.060060
itel 1 lambda 0.400000 stress 0.068653 penalty 0.028190
itel 1 lambda 0.500000 stress 0.080258 penalty 0.011356
itel 1 lambda 0.600000 stress 0.086572 penalty 0.003578
itel 1 lambda 0.700000 stress 0.089140 penalty 0.000854
itel 1 lambda 0.800000 stress 0.089898 penalty 0.000116
itel 1 lambda 0.830000 stress 0.089958 penalty 0.000053
```

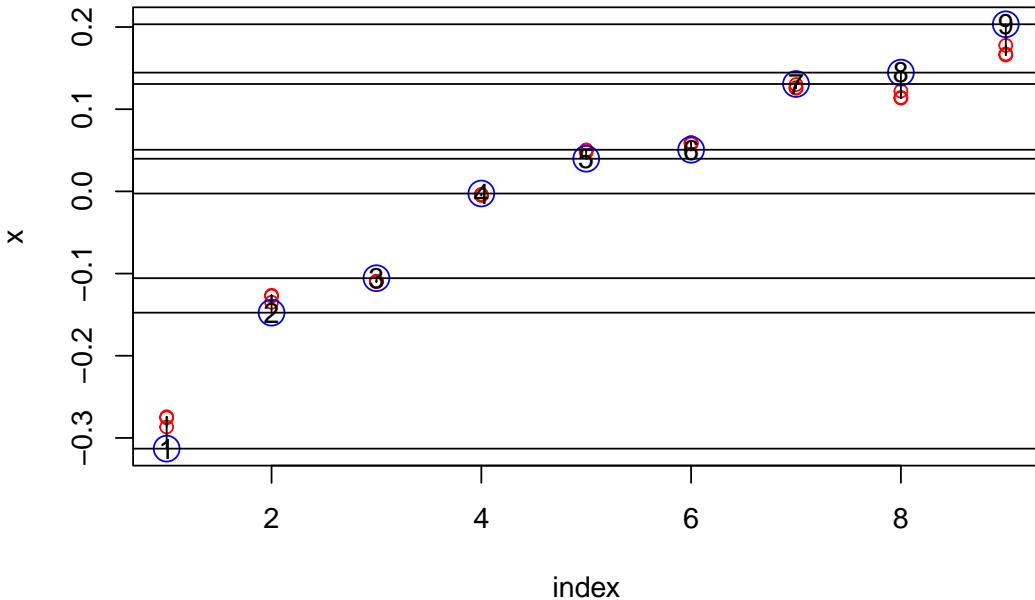
```
itel 1 lambda 0.840000 stress 0.089970 penalty 0.000040
itel 197 lambda 0.850000 stress 0.089949 penalty 0.000000
```



#### 24.5.3.8 Vegetables

Our first one-dimensional example uses paired comparisons of 9 vegetables, originating with Guilford (1954). The proportions are transformed to dissimilarities by using the absolute values of the normal quantile function, i.e.  $\delta_{ij} = |\Phi^{-1}(p_{ij})|$ . We use a short sequence for  $\lambda$ .

```
itel 1412 lambda 0.000000 stress 0.013675 penalty 0.269308
itel 5 lambda 0.010000 stress 0.013716 penalty 0.114786
itel 5 lambda 0.100000 stress 0.016719 penalty 0.069309
itel 23 lambda 1.000000 stress 0.035301 penalty 0.000000
```

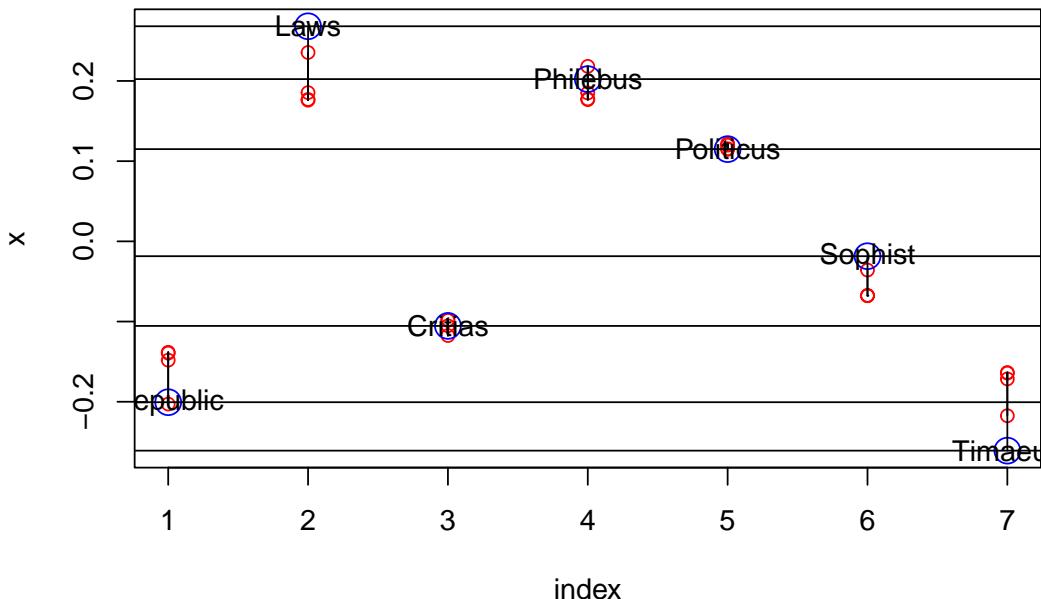


This example was previously analyzed by me in De Leeuw (2005c) using enumeration of all permutations. I found 14354 isolated local minima, and a global minimum equal to the one we computed here.

#### 24.5.3.9 Plato

Mair, Groenen, and De Leeuw (2019) use seriation of the works of Plato, from the data collected by D. R. Cox and Brandwood (1959), as an example of unidimensional scaling. We first run this example with our usual sequence of five  $\lambda$  values.

```
itel 169 lambda 0.000000 stress 0.000000 penalty 0.410927
itel 3 lambda 0.010000 stress 0.000062 penalty 0.255246
itel 3 lambda 0.100000 stress 0.005117 penalty 0.194993
itel 4 lambda 1.000000 stress 0.106058 penalty 0.019675
itel 9 lambda 10.000000 stress 0.139462 penalty 0.000000
```



This gives the order

```
[,1]
[1,] "Timaeus"
[2,] "Republic"
[3,] "Critias"
[4,] "Sophist"
[5,] "Politicus"
[6,] "Philebus"
[7,] "Laws"
```

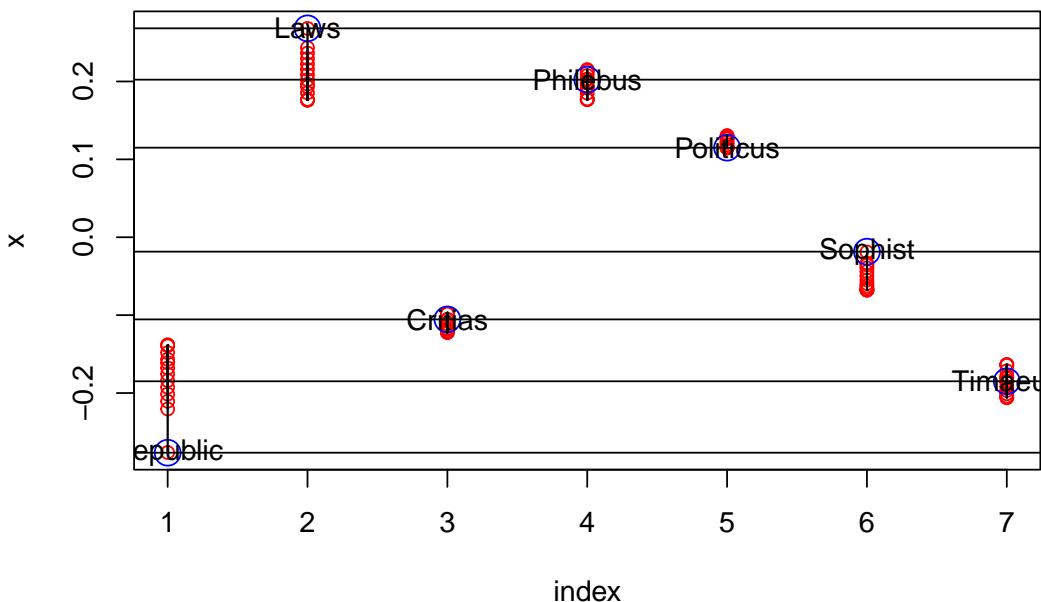
which is different from the order at the global minimum that has Republic before Timaeus. Thus we have recovered a local minimum, and it seems our sequence of  $\lambda$  values was not fine enough to do the job properly. So we try a longer and finer sequence.

```
itel 169 lambda 0.000000 stress 0.000000 penalty 0.410927
itel 3 lambda 0.000100 stress 0.000000 penalty 0.263015
itel 3 lambda 0.001000 stress 0.000001 penalty 0.262280
itel 3 lambda 0.010000 stress 0.000064 penalty 0.255078
```

```

itel 3 lambda 0.100000 stress 0.005123 penalty 0.194945
itel 2 lambda 0.200000 stress 0.016184 penalty 0.147493
itel 1 lambda 0.300000 stress 0.026997 penalty 0.119323
itel 1 lambda 0.400000 stress 0.040023 penalty 0.093615
itel 1 lambda 0.500000 stress 0.053688 penalty 0.072330
itel 1 lambda 0.600000 stress 0.066833 penalty 0.055452
itel 1 lambda 0.700000 stress 0.078832 penalty 0.042269
itel 1 lambda 0.800000 stress 0.089439 penalty 0.032019
itel 1 lambda 0.900000 stress 0.098557 penalty 0.024079
itel 1 lambda 1.000000 stress 0.106135 penalty 0.017940
itel 6 lambda 2.000000 stress 0.130789 penalty 0.000148
itel 13 lambda 3.000000 stress 0.131135 penalty 0.000000

```



Now the order is

```

[,1]
[1,] "Republic"
[2,] "Timaeus"
[3,] "Critias"
[4,] "Sophist"

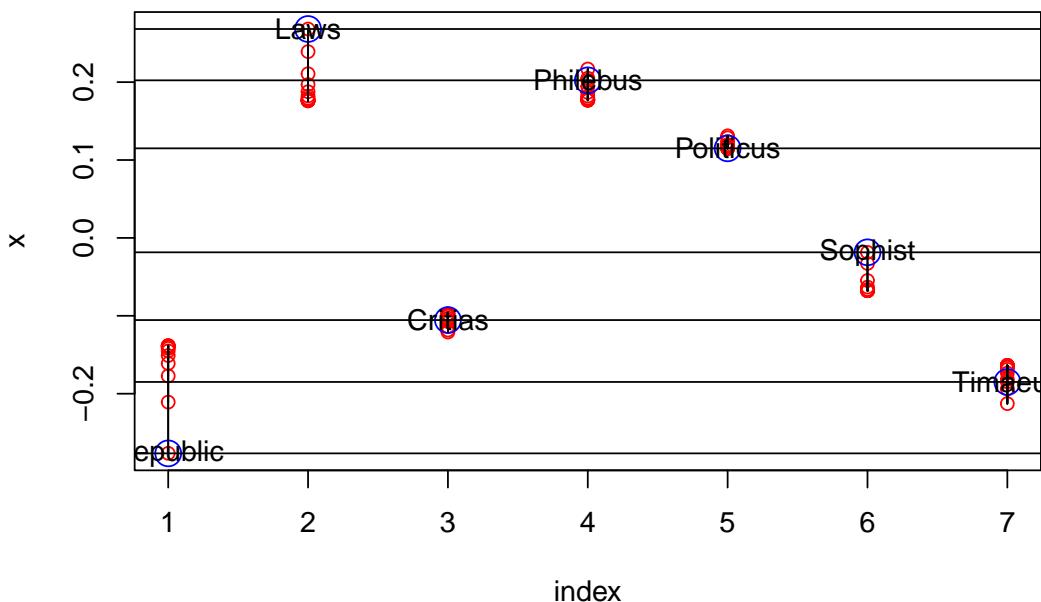
```

```
[5,] "Politicus"
[6,] "Philebus"
[7,] "Laws"
```

which does indeed correspond to the global minimum.

With a different  $\lambda$  sequence we find the same solution.

```
itel 169 lambda 0.000000 stress 0.000000 penalty 0.410927
itel 3 lambda 0.001000 stress 0.000001 penalty 0.262296
itel 2 lambda 0.002000 stress 0.000003 penalty 0.261483
itel 2 lambda 0.004000 stress 0.000010 penalty 0.259872
itel 2 lambda 0.008000 stress 0.000041 penalty 0.256690
itel 2 lambda 0.016000 stress 0.000159 penalty 0.250470
itel 2 lambda 0.032000 stress 0.000613 penalty 0.238574
itel 2 lambda 0.064000 stress 0.002266 penalty 0.216785
itel 2 lambda 0.128000 stress 0.007791 penalty 0.180067
itel 2 lambda 0.256000 stress 0.023483 penalty 0.127006
itel 2 lambda 0.512000 stress 0.056940 penalty 0.067948
itel 3 lambda 1.024000 stress 0.107743 penalty 0.017937
itel 8 lambda 2.048000 stress 0.131059 penalty 0.000032
itel 9 lambda 4.096000 stress 0.131135 penalty 0.000000
```



The order is

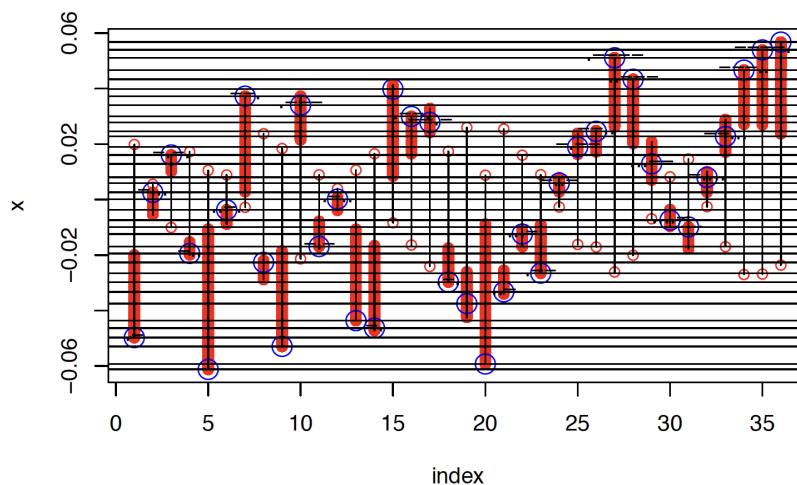
```
[,1]
[1,] "Republic"
[2,] "Timaeus"
[3,] "Critias"
[4,] "Sophist"
[5,] "Politicus"
[6,] "Philebus"
[7,] "Laws"
```

#### 24.5.3.10 Morse in One

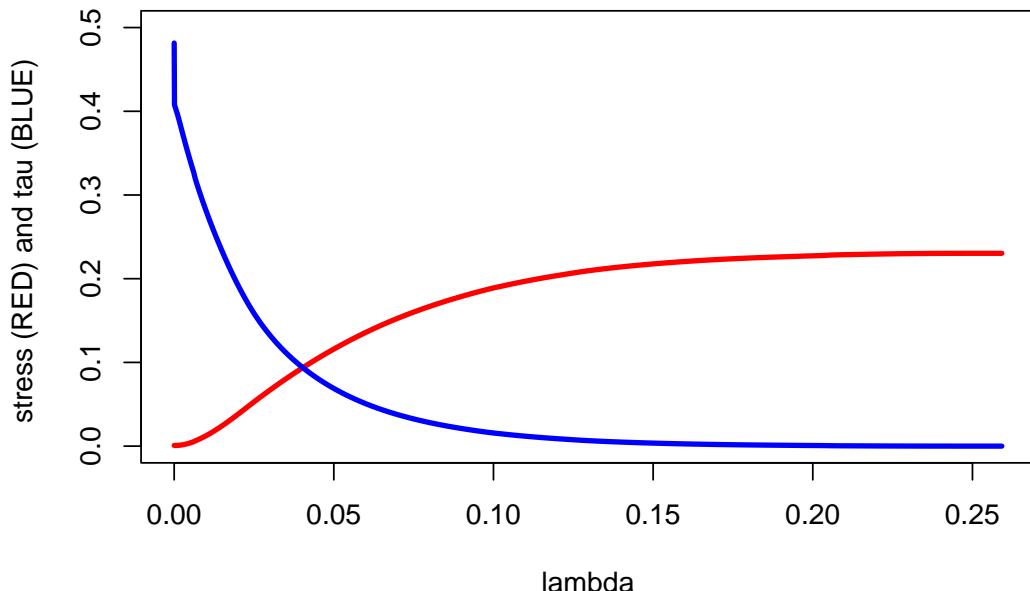
Now for a more challenging example. The Morse code data have been used to try out exact unidimensional MDS techniques, for example by Palubeckis (2013). We will enter the global minimum contest by using 10,000 values of  $\lambda$ , in an equally spaced sequence from 0 to 10. This is not as bad as it sounds. For the 10,000 FDS solutions `system.time()` tells us

```
user system elapsed
15.292 0.489 15.827
```

The one-dimensional plot show quite a bit of movement, but much of it seems to be contained in the very first change of  $\lambda$ .



We can also plot stress and the penalty term as functions of  $\lambda$ . Again, note the big change in the penalty term when  $\lambda$  goes from zero to 0.001.



After the first 2593 values of  $\lambda$  the penalty term is zero and we stop, i.e. we estimate  $\lambda_+$  is 2.593. At that point we have run a total of 5013 FDS iterations, and thus on average about two iterations per  $\lambda$  value. Stress has increased from 0.0007634501 to 0.2303106976 and the penalty value has decreased from 0.4815136419 to 0.0000000001. We find the following order of the points on the dimension.

```
[,1]
[1,] "."
[2,] "-"
[3,] ".."
[4,] ".-"
[5,] "-."
[6,] "--"
[7,] "... "
[8,] "...-"
[9,] "-.-"
```

```

[10,] ".--"
[11,] "...."
[12,] "-.."
[13,] "-.-"
[14,] "...-"
[15,] "....."
[16,] "....-"
[17,] "...-"
[18,] ".-.."
[19,] "-... "
[20,] "-...-"
[21,] "-...."
[22,] "....-"
[23,] "-.-."
[24,] "-.--"
[25,] "--... "
[26,] "--.."
[27,] "--.-"
[28,] ".---."
[29,] ".---"
[30,] "--."
[31,] "----"
[32,] ".----"
[33,] "----."
[34,] ".----"
[35,] "----."
[36,] "----"

```

Our order, and consequently our solution, is the same as the exact global solution given by Palubeckis (2013). See his table 4, reproduced below. The difference is that computing our solution takes 10 seconds, while his takes 494 seconds. But of course we would not have known we actually found the global minimum if the exact exhaustive methods had not analyzed the same data as well.



# Chapter 25

## Mathematical Addenda

This huge chapter has several purposes. It reviews some well known mathematical results, defines notation and terminology, serves as a reminder, and provides a reference. We do not give proofs of these results, and not many references, because the proofs be found in most textbooks. But some sections (for instance the ones on majorization, quadratic programming, splines, and quadratic penalties) we offer more than that. These sections are taken mostly from my unpublished papers and they go into much more detail. The results I present are also well known, but somewhat harder to find, and I will try to emphasize the aspects relevant for MDS.

### 25.1 Notation

#### 25.1.1 Symbols

$\mathbb{R}^n$

$\mathbb{R}^{n \times p}$

$\mathbb{S}^n$ : unit sphere in

$\mathbb{B}^n$ : unit ball

$\sigma$ : generic MDS loss function

$\mathcal{D}f$ : derivative

$\mathcal{D}_i f$ : partial derivative

$\min_{x \in X} f(x)$

$\operatorname{argmin}_{x \in X} f(x)$

directional derivative gradient Hessian

$\partial f$ : subdifferential

$f : X \Rightarrow Y$ : function

$f : X_1 \times \dots \times X_n \Rightarrow Y$

$\otimes$ : Kronecker product of matrices

$\text{vec}$

$\oplus$ : direct sum of matrices

$1 : n$

$\Pi$ : permutation

$[a, b]$ : closed interval

### 25.1.2 Summation

$$\sum_{1 \leq i < j \leq n} x_{ij}$$

$$\sum_{i=1}^n \{x_i y_i \mid x_i > 0\}$$

## 25.2 Matrices

### 25.2.1 Matrix Spaces

The linear space of all  $n \times p$  matrices is  $\mathbb{R}^{n \times p}$ . The subspace of  $\mathbb{R}^{n \times n}$  of all symmetric matrices is  $\mathbb{S}^{n \times n}$ .

### 25.2.2 Constants

All vectors are column vectors. In this section we define some matrix and vector constants. We assume that the order of the matrices, or the length of the vectors, is clear from the context, as it will be a.e. in the body of the book.

- $I$  is the identity matrix,
- $e$  is a vector with all elements +1,
- $\iota$  is a vector with elements  $1, 2, \dots, n$ ,
- $E$  is a matrix with all elements +1,
- $\emptyset$  is a matrix with all elements 0,
- $J$  is the centering matrix  $J = I - \frac{1}{e'e}E$ .

### 25.2.3 Diff Matrices

- A *unit vector*  $e_i$  is a vector with element  $i$  equal to +1 and all other elements equal to 0. A *unit matrix*  $E_{ij}$  is a matrix of the form  $e_i e_j'$ ,
- A *diff matrix*  $A_{ij}$  is a matrix of the form  $(e_i - e_j)(e_i - e_j)'$ .

The element in row  $i$  and column  $j$  of a matrix  $X$  is normally referred to as  $x_{ij}$ . But in some cases, to prevent confusion, we use the notation  $\{X\}_{ij}$ . Thus, for example,  $\{e_i\}_j = \delta^{ij}$ , where  $\delta^{ij}$  is *Kronecker's delta* (zero when  $i = j$  and one otherwise).

The diff matrices  $A_{ij}$  with  $i \neq j$  have only four non-zero elements

$$\begin{aligned}\{A_{ij}\}_{ii} &= \{A_{ij}\}_{jj} = +1, \\ \{A_{ij}\}_{ij} &= \{A_{ij}\}_{ji} = -1,\end{aligned}\tag{25.1}$$

and all other elements of  $A_{ij}$  are zero. Thus  $A_{ij} = A_{ji}$  and  $A_{ii} = 0$ . Diff matrices are symmetric, and positive semidefinite. They are also *doubly-centered*, which means that their rows and columns add up to zero. If  $i \neq j$  they are of rank one and have one eigenvalue equal to two, which means  $A_{ij}^s = 2^{s-1}A_{ij}$ . Also

$$\sum_{1 \leq i < j \leq n} A_{ij} = nI - ee' = nJ, \quad (25.2)$$

with  $J$  the centering matrix\*.

#### 25.2.4 Sign Matrices

$S(x)$  is the *sign matrix* of  $x \in \mathbb{R}^n$  if  $s_{ij}(x) = \text{sign}(x_i - x_j)$  for all  $i$  and  $j$ , i.e.

$$s_{ij}(x) := \begin{cases} +1 & \text{if } x_i > x_j, \\ -1 & \text{if } x_i < x_j, \\ 0 & \text{if } x_i = x_j. \end{cases} \quad (25.3)$$

The set of all sign matrices is  $\mathcal{S}$ .

Sign matrices are hollow and anti-symmetric. A sign matrix  $S$  is *strict* if its only zeroes are on the diagonal, i.e.  $S = S(P\iota)$  for some permutation matrix  $P$ . The set of strict sign matrices is  $\mathcal{S}_+$ . Since there is a 1:1 correspondence between strict sign matrices and permutations, there are  $n!$  strict sign matrices. The row sums and column sums of a strict sign matrix are some permutation of the numbers  $n - 2\iota + 1$ .

#### 25.2.5 Sums and Products

If  $A$  and  $B$  are matrices with the same number of rows and columns then the *Hadamard product*  $C := A \times B$  is defined as  $c_{ij} = a_{ij}b_{ij}$ . The *Hadamard square*  $A^{(2)}$  is defined as  $A \times A$ , and similar definitions apply to higher Hadamard powers.

If  $A$  and  $B$  are matrices then their *direct sum*  $A \oplus B$  is the matrix

$$C := \begin{bmatrix} A & \emptyset \\ \emptyset & B \end{bmatrix}. \quad (25.4)$$

Direct sums of more than two matrices are defined recursively as  $D = C \oplus (A \oplus B)$ , and so on.

If  $A$  and  $B$  are matrices then their *direct product* or *Kronecker product*  $C = A \otimes B$  is the matrix with blocks  $a_{ij}B$ . Thus, for example, if  $A$  is  $2 \times 3$  and  $B$  is  $n \times m$  then  $C$  is  $(2n) \times (3m)$  and

$$C = \begin{bmatrix} a_{11}B & a_{12}B & a_{13}B \\ a_{21}B & a_{22}B & a_{23}B \end{bmatrix}. \quad (25.5)$$

If  $A$  and  $B$  are arrays ... then their *outer product*  $C = A \Delta B$  is the four-dimensional array with  $c_{ijkl} = a_{ij}b_{kl}$ . Thus for two vectors  $a$  and  $b$  we have  $ab' = a \Delta b$ . The *outer sum*  $C = A \square B$  is defined in the same way:  $c_{ijkl} = a_{ij} + b_{kl}$ . That's pretty atrocious notation, but I only use it once or twice in the whole book.

### 25.2.6 Inverse

Throughout the book we use  $X^{-1}$  for the Moore-Penrose Inverse (MPI) of  $X$  (see section 25.2.10). If  $X$  is square and non-singular the MPI is simply equal to the inverse of  $X$ , i.e  $XX^{-1} = X^{-1}X = I$ .

### 25.2.7 The Loewner order

If  $A$  and  $B$  are square symmetric matrices then

- $A \precsim B$  means  $B - A$  is positive semi-definite (PSD).
- $A \succsim B$  means  $B - A$  is negative semi-definite (NSD).
- $A \prec B$  means  $B - A$  is positive definite (PD).
- $A \succ B$  means  $B - A$  is negative definite (ND).

### 25.2.8 Eigen Decomposition

If  $A$  is square symmetric of order  $n$  then there exists

- a matrix  $K$  of order  $n$  with  $K'K = KK' = I$ ,
- a diagonal matrix  $\Lambda$  of order  $n$ ,

such that  $A = K\Lambda K'$ . If  $A$  is PSD, then so is  $\Lambda$ . The number of non-zero diagonal elements in  $\Lambda$  is the rank of  $A$ .

### 25.2.9 Singular Value Decomposition

If  $X$  is an  $n \times m$  matrix then there exist

- a matrix  $K$  of order  $n$  with  $K'K = KK' = I$ ,
- a matrix  $L$  of order  $m$  with  $L'L = LL' = I$ ,
- a non-negative diagonal  $n \times m$  matrix  $\Lambda$ ,

such that  $X = K\Lambda L'$ . The number of non-zero diagonal elements in  $\Lambda$  is the rank of  $X$ .

If  $X$  has rank  $r$  we can partition the matrices in the SVD accordingly as

$$\begin{aligned} K &= \begin{bmatrix} K_+ & K_0 \end{bmatrix}, \\ \Lambda &= \begin{bmatrix} \Lambda_{++} & \emptyset \\ \emptyset & \emptyset \end{bmatrix}, \\ L &= \begin{bmatrix} L_+ & L_0 \end{bmatrix}, \end{aligned} \tag{25.6}$$

where  $\Lambda_{++}$  is of order  $r$  and pd and both  $K_+$  and  $L_+$  in (25.6) have  $r$  columns. Thus also  $X = K_+\Lambda_{++}L'_+$ .

### 25.2.10 Moore-Penrose Inverse

The MPI can be defined in terms of the SVD as  $X^{-1} = L_+\Lambda_{++}^{-1}K'_+$ .

It follows that the MPI satisfies the four *Penrose Conditions*

- $XX^{-1}$  is symmetric,
- $X^{-1}X$  is symmetric,
- $X^{-1}XX^{-1} = X^{-1}$ ,
- $XX^{-1}X = X$ .

In fact, these four conditions uniquely define the MPI.

### 25.2.11 Full-rank Decomposition

If  $X = FG'$  is a full rank decomposition, then the Moore-Penrose inverse is  $X^+ = G(F'G)^{-1}F'$ .

### 25.2.12 SDC matrices

Diagonally dominant  $A_{ij}$  is basis

Taussky (1949)

To analyze the singularity of  $V$  in more detail we observe that  $z'Vz = \sum \sum_{1 \leq i < j \leq n} w_{ij}(z_i - z_j)^2$ . This is zero if and only if all  $w_{ij}(z_i - z_j)^2$  are zero. If we permute the elements of  $z$  such that  $z_1 \leq \dots \leq z_n$  then the matrix with elements  $(z_i - z_j)^2$  can be partitioned such that the diagonal blocks, corresponding with tie-blocks in  $z$ , are zero and the off-diagonal blocks are strictly positive. Thus  $z'Vz = 0$  if and only if the corresponding off-diagonal blocks of  $W$  are zero. In other words, we can find a  $z$  such that  $z'Vz = 0$  if and only if  $W$  is the direct sum of a number of smaller matrices. If this is not the case we call  $W$  *irreducible*, and  $z'Vz > 0$  for all  $z \neq e$ , so that the rank of  $V$  is  $n - 1$ . #### Sign Matrices

## 25.3 Inequalities

There are some elementary inequalities that play a large role throughout the book.

### 25.3.1 Cauchy-Schwartz

We start with the Cauchy-Schwarz inequality (CS, from now on).

**Theorem 25.1.** *If  $A$  is positive semi-definite then  $(x'Ay)^2 \leq (x'Ax)(y'Ay)$ . There is equality if and only if there is an  $(\alpha, \beta) \neq (0, 0)$  such that  $A(\alpha x + \beta y) = 0$ .*

### 25.3.2 Arithmetic/Geometric Mean

Next is the arithmetic mean/geometric mean inequality (from now on AM/GM). We only need the simplest case, proved by expanding  $\frac{1}{2}(\sqrt{x} - \sqrt{y})^2 \geq 0$ ,

**Theorem 25.2.** If  $x \geq 0$  and  $y \geq 0$  then  $\frac{1}{2}(x + y) \geq \sqrt{xy}$ . There is equality if and only if  $x = y$ .

Finally, from  $\frac{1}{2}(x - y)'A(x - y) \geq 0$ ,

**Corollary 25.1.** If  $A$  is positive semi-definite then  $x'Ay \leq \frac{1}{2}(x'Ax + y'Ay)$ .

## 25.4 Calculus/Analysis

### 25.4.1 Functions

A function  $f$  from  $X$  to  $Y$  is a subset of the Cartesian product  $X \otimes Y$ , for which  $(x, y) \in f$  and  $(x, z) \in f$  implies  $y = z$ . Thus there is a unique  $y$  associated with each  $x$ . We usually write  $f : X \Rightarrow Y$ , and we try distinguish the function  $f$  from its value at  $x \in X$ , which is  $f(x)$ .

$$f_j(x)$$

If  $f$  on  $X \otimes Y$  then  $f(\bullet, y)$  is on  $X$   $g(x) = f(x, y)$ . Projection

### 25.4.2 Differentiation

#### 25.4.2.1 Derivative

Suppose  $f$  maps an open set  $X \subseteq \mathbb{R}^n$  into  $\mathbb{R}^m$ . Then  $f$  is (Fréchet, totally) differentiable at  $x \in X$  if there is a linear map  $\mathcal{D}f$  of  $X$  into  $\mathbb{R}^m$  such that

$$\mathcal{D}f(x) = \lim_{\delta \rightarrow 0} \frac{f(x + \delta) - f(x) - \mathcal{D}f(x)\delta}{\|\delta\|} = 0, \quad (25.7)$$

which we can also write as

$$\mathcal{D}f(x + \delta) = f(x) + \mathcal{D}f(x)\delta + o(\|\delta\|). \quad (25.8)$$

So, just to be clear about the notation,  $\mathcal{D}f(x)(y)$  means the value in  $\mathbb{R}^m$  of the linear map  $\mathcal{D}f(x)$  at  $y \in X$ .

For real-valued functions of a single real variable we also use  $f'(x), f''(x), \dots$  for the successive derivatives.

Second derivatives symmetric quadratic form

$$\mathcal{D}^2 f(x) = \mathcal{D}(\mathcal{D}f(x))$$

$$\mathcal{D}^2 f(x)(y, z) = \mathcal{D}^2 f(x)(z, y)$$

#### 25.4.2.2 Partial Derivatives

$$\mathcal{D}_f(x, y) =$$

$$f_i(\epsilon) = f(x + \epsilon e_i)$$

$$\mathcal{D}_i f(x) = \mathcal{D}f_i(0)$$

$$\mathcal{D}f(x)(y) = \sum_{i=1}^n \mathcal{D}_i f(x)(y_i)$$

#### 25.4.2.3 Jacobian

$$\{\mathcal{J}F(x)\}_{ij} = \{\mathcal{D}_i f_j(x)\}$$

#### 25.4.2.4 Gradient

$$\{\nabla f(x)\}_i = \mathcal{D}f(x)(e_i)$$

#### 25.4.2.5 Hessian

$$\{\nabla^2 f(x)\}_{ij} = \mathcal{D}^2 f(x)(e_i, e_j)$$

### 25.4.2.6 Differentials

If  $f$  is differentiable at  $x$  then the function  $\mathcal{D}f(x)(u)$  is called the differential  
Flett (1980)

Magnus and Neudecker (1999)

### 25.4.3 Directional Derivatives

#### 25.4.3.1 First Order

For a function  $f : \mathbb{R}^{n \times p} \rightarrow \mathbb{R}$  we define the directional derivative at  $X$  in direction  $Y$  as

$$D_+ f(x; y) := \lim_{\epsilon \downarrow 0} \frac{f(x + \epsilon y) - f(x)}{\epsilon}, \quad (25.9)$$

where  $\epsilon \downarrow 0$  means a limit from the right, in which epsilon only takes positive values. The definition assumes, of course, that the limit exists.

#### 25.4.3.2 Second Order

Similarly, the (Dini) second directional derivative is

$$D_+^2 f(x; y) := \lim_{\epsilon \downarrow 0} \frac{f(x + \epsilon y) - f(x) - \epsilon D_+ f(x; y)}{\frac{1}{2}\epsilon^2}, \quad (25.10)$$

which assume that both  $D_+ f(x; y)$  and the limit exist. If  $f$  is twice differentiable

$$D_+^2 f(x; y) = \mathcal{D}^2 f(x)(y, y).$$

The second directional derivative in the sense of Ben-Tal and Zowe is

$$\mathcal{D}_{BZ}^2 f(x; y, z) := \lim_{\epsilon \downarrow 0} \frac{f(x + \epsilon y + \epsilon^2 z) - f(x) - \epsilon \mathcal{D}_+ f(x; y)}{\epsilon^2}, \quad (25.11)$$

provided again that both  $\mathcal{D}_+ f(x; y)$  and the limit exist.

If  $f$  is twice differentiable

$$\frac{1}{2}\{\mathcal{D}^2f(x)(y, y) + \mathcal{D}f(x)(z)\} \quad (25.12)$$

#### 25.4.4 Local Minima, Critical Points

A function  $f$  has a *global minimum* on  $X$  at  $\hat{x}$  if  $f(\hat{x}) \leq f(x)$  for all  $x \in X$ .

A function  $f$  has a *local minimum* on  $X$  at  $\hat{x}$  if there is a neighborhood  $\mathcal{N}$  of  $\hat{x}$  such that  $f(\hat{x}) \leq f(x)$  for all  $x \in \mathcal{N} \cap X$ .

Global and local maxima of  $f$  are global and local minima of  $-f$ .

A function  $f$  has a *singular point* at  $x$  if it is not differentiable at  $x$ .

A function  $f$  has a *stationary point* at  $x$  if it is differentiable at  $x$  and  $\mathcal{D}f(x) = 0$ .

A function  $f$  has a *saddle point* at a stationary point  $x$  if it is neither a local maximum nor a local minimum.

#### 25.4.5 Necessary Conditions

First order

Ben-Tal and Zowe

#### 25.4.6 l'Hôpital's Rule

We all know that  $0/0$  is not defined and should be avoided at all cost. But then again we have

$$\lim_{x \rightarrow 0} \frac{\sin(x)}{x} = \cos(0) = 1,$$

and in fact  $\sup_x \frac{\sin(x)}{x} = 1$ . Or, for that matter, if  $f$  is differentiable at  $x$  then

$$\lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon) - f(x)}{\epsilon} = f'(x)$$

If  $f : \mathbb{R} \Rightarrow \mathbb{R}$  and  $g : \mathbb{R} \Rightarrow \mathbb{R}$  are two functions

- differentiable in an interval  $\mathcal{I}$ , except possibly at  $c \in \mathcal{I}$ ,
- $g'(x) \neq 0$  for all  $x \in \mathcal{I}$ ,
- $\lim_{x \rightarrow c} f(x) = \lim_{x \rightarrow c} g(x) = 0$  or  $\lim_{x \rightarrow c} f(x) = \lim_{x \rightarrow c} g(x) = \pm\infty$ ,  
then

$$\lim_{x \rightarrow c} \frac{f(x)}{g(x)} = \lim_{x \rightarrow c} \frac{f'(x)}{g'(x)}$$

We use l'Hôpital's rule in chapter 14, section 14.3.1 on degeneracies in non-metric unfolding. We have not explored the multivariate versions of l'Hôpital's rule, discussed for example by Lawlor (2020).

### 25.4.7 Implicit Functions

## 25.5 Convexity

### 25.5.1 Existence of Minima

Projection on a convex set

### 25.5.2 Duality

### 25.5.3 Subdifferentials

Almost everywhere twice differentiable,

A vector  $g$  is a *subgradient* of a convex function  $f$  at  $x$  if  $f(x) \geq f(y) + g'(x-y)$  for all  $y$ . If  $f$  is differentiable at  $x$  then the subgradient is unique and equal to gradient, i.e. the vector of partial derivatives at  $x$ . The set of all subgradients of  $f$  at  $x$  is the *subdifferential*  $\partial f(x)$ . Subdifferentials of convex functions are non-empty, compact, and convex.

There are some useful calculus rules for the subdifferentials of finite convex function defined on  $\mathbb{R}^n$ . They can be found, with proofs, in any book on convex analysis, starting with Rockafellar (1970).

**Sum rule:** If  $f_1, \dots, f_m$  are convex,  $\lambda_1, \dots, \lambda_m$  are positive, and  $h = \lambda_1 f_1 + \dots + \lambda_m f_m$  then

$$\partial h(x) = \lambda_1 \partial f_1(x) + \dots + \lambda_m \partial f_m(x).$$

(Hiriart-Urruty and Lemaréchal (1993), theorem 4.1.1)

**Affine rule:** If  $f$  is convex and  $h(x) = f(Ax + b)$  then

$$\partial h(x) = A' \partial f(Ax + b).$$

(Hiriart-Urruty and Lemaréchal (1993), theorem 4.2.1)

**Sup rule:** Suppose  $f : \mathbb{R}^n \otimes \mathbb{Y}$ , with  $\mathbb{Y} \subseteq \mathbb{R}^m$ , and suppose

1.  $f(\bullet, y)$  is convex on  $\mathbb{R}^n$  for every  $y \in \mathbb{Y}$ ,
2.  $f(x, \bullet)$  is continuous on  $\mathbb{Y}$  for every  $x \in \mathbb{R}^n$ ,
3.  $\mathbb{Y}$  is compact.

If  $h(x) = \sup_{y \in \mathbb{Y}} f(x, y)$  then

$$\partial h(x) = \text{conv} \left( \bigcup_{y \in Y(x)} \partial f(x, y) \right),$$

where  $Y(x) = \{y \in \mathbb{Y} \mid f(x, y) = h(x)\}$  is the active set and  $\text{conv}()$  is the convex hull (Hiriart-Urruty and Lemaréchal (1993), theorem 4.4.2).

**Max rule:**  $f_1, \dots, f_m$  are convex and  $h(x) = \max_{j=1}^m f_j(x)$  then

$$\partial h(x) = \text{conv} \left\{ \bigcup_{k \in J(x)} \partial f_k(x) \right\},$$

where  $J(x) = \{j \mid f_j(x) = h(x)\}$  is the *active set* and  $\text{conv}(X)$  is the convex hull of  $X$ .

#### 25.5.4 DC Functions

Extremes, critical points

Toland Duality

### 25.5.5 Danskin-Berge Theorem

During my visit to Bell Labs in Murray Hill, somewhere around the time that Watergate was raging, there was some major excitement in our group because one of us had discovered that if  $g(x) = \min_y f(x, y)$  then  $\mathcal{D}g(x) = \mathcal{D}_1 f(x, y(x))$ , where  $y(x)$  is such that  $g(x) = f(x, y(x))$ . This result simplified calculation of partial derivatives in MDS and related techniques, and of course psychometrics at the time was all about partial derivatives.

Danskin (1966) Berge (1963)

## 25.6 Fixed-Point Iterations

In fixed point iteration we generate a sequence of points by the rule

$$x^{(k+1)} = F(x^{(k)})$$

In this book  $F$  is always a continuous self-map on  $\mathbb{R}^n$ , i.e.  $F : \mathbb{R}^n \Rightarrow \mathbb{R}^n$ .

If the sequence  $\{x_k\}$  converges to, say,  $x_\infty$  then, by the continuity of  $F$ , we have  $x_\infty = F(x_\infty)$ , and we say that  $x_\infty$  is a *fixed point* of  $F$ . Many iterative algorithms, and all iterative algorithms in this book, generate sequences by fixed point iterations that hopefully converge to fixed points.

It should be noted that iterations of the form

$$x^{(k+1)} = F(x^{(k)}, x^{(k-1)}, \dots, x^{(k-l)}),$$

which do look more general, can be written as special cases of .... Define

$$y^{(k)} := \begin{bmatrix} x^{(k)} \\ x^{(k-1)} \\ \vdots \\ x^{(k-l)} \end{bmatrix}$$

and, using  $F$  from ...

$$y^{(k+1)} = G(y^{(k)}) := \begin{bmatrix} x^{(k+1)} = F(y^k) \\ x^{(k)} \\ \vdots \\ x^{(k-l-1)} \end{bmatrix}$$

### 25.6.1 Point-to-set Maps

### 25.6.2 Zangwill's Theorem

### 25.6.3 Ostrowski's Theorem

## 25.7 Least Squares

### 25.7.1 Quadratic Programming

In this section we construct an algorithm for a general weighted linear least squares projection problem with equality and/or inequality constraints. It uses duality and unweighting majorization. The section takes the form of a small essay, with examples. This may seem somewhat excessive, but it provides an easy reference for both you and me and it serves as a manual for the corresponding R code.

We start with the *primal problem*, say problem  $\mathcal{P}$ , which is minimizing

$$f(x) = \frac{1}{2}(Hx - z)'V(Hx - z) \quad (25.13)$$

over all  $x$  satisfying equalities  $Ax \geq b$  and equations  $Cx = d$ . We suppose the *Slater condition* is satisfied, i.e. there is an  $x$  such that  $Ax > b$ . And, in addition, we suppose the system of inequalities and equations is *consistent*, i.e. has at least one solution.

We first reduce the primal problem to a simpler, and usually smaller, one by partitioning the loss function. Define

$$\begin{aligned} W &:= H'VH, \\ y &:= W^{-1}H'Vz, \\ Q &:= (I - H(H'VH)^{-1}H'V). \end{aligned} \quad (25.14)$$

Then

$$(Hx - y)'V(Hx - y) = (x - y)'W(x - y) + y'Q'VQy, \quad (25.15)$$

The simplified primal problem  $\mathcal{P}'$  is to minimize  $(x-y)'W(x-y)$  over  $Ax \geq b$  and  $Cx = d$ , where  $W$  is assumed to be positive definite. Obviously the solutions to  $\mathcal{P}$  and  $\mathcal{P}'$  are the same. The two loss function values only differ by the constant term  $y'Q'VQy$ .

We do not solve  $\mathcal{P}'$  directly, but we use Lagrangian duality and solve the dual quadratic programming problem. The Lagrangian for  $\mathcal{P}'$  is

$$\mathcal{L}(x, \lambda, \mu) = \frac{1}{2}(x-y)'W(x-y) - \lambda'(Ax-b) - \mu'(Cx-d), \quad (25.16)$$

where  $\lambda \geq 0$  and  $\mu$  are the Lagrange multipliers.

Now

$$\begin{aligned} \max_{\lambda \geq 0} \max_{\mu} \mathcal{L}(x, \lambda, \mu) &= \\ &= \begin{cases} \frac{1}{2}(x-y)'W(x-y) - \lambda'(Ax-b) - \mu'(Cx-d) & \text{if } Ax \geq b, \\ +\infty & \text{otherwise,} \end{cases} \end{aligned} \quad (25.17)$$

and thus

$$\min_x \max_{\lambda \geq 0} \max_{\mu} \mathcal{L}(x, \lambda, \mu) = \min_{Ax \geq b} \min_{Cx=d} \frac{1}{2}(x-y)'W(x-y), \quad (25.18)$$

which is our original simplified primal problem  $\mathcal{P}'$ .

We now look at the *dual problem*  $\mathcal{D}'$  (of  $\mathcal{P}'$ ), which means solving

$$\max_{\lambda \geq 0} \max_{\mu} \min_x \mathcal{L}(x, \lambda, \mu). \quad (25.19)$$

The inner minimum over  $x$  for given  $\lambda$  and  $\mu$  is attained at

$$x = y + W^{-1}(A' | C') \begin{bmatrix} \lambda \\ \mu \end{bmatrix}, \quad (25.20)$$

and is equal to  $-g(\lambda, \mu)$ , where

$$\frac{1}{2} \begin{bmatrix} \lambda & \mu \end{bmatrix} \begin{bmatrix} AW^{-1}A' & AW^{-1}C' \\ CW^{-1}A' & CW^{-1}C' \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} + \lambda'(Ay - b) + \mu'(Cy - d) \quad (25.21)$$

Our strategy is to solve  $\mathcal{D}'$  for  $\lambda \geq 0$  and/or  $\mu$ . Because of our biases we do not maximize  $-g$ , we minimize  $g$ . Then compute the solution of both  $\mathcal{P}'$  and  $\mathcal{P}$  from (25.20). The duality theorem for quadratic programming tells us the values of  $f$  at the optimum of  $\mathcal{P}'$  and  $-g$  at the optimum of  $\mathcal{D}'$  are equal, and of course the value at the optimum of  $\mathcal{P}$  is that of  $\mathcal{P}'$  plus the constant  $y'QVQy$ .

From here on we can proceed with unweighting in various ways. We could, for instance, minimize out  $\mu$  and then unweight the resulting quadratic form. Instead, we go the easy way. Majorize the partitioned matrix  $K$  in the quadratic part of (25.21) by a similarly partitioned diagonal positive matrix  $E$ .

$$E := \begin{bmatrix} F & \emptyset \\ \emptyset & G \end{bmatrix} \gtrsim K := \begin{bmatrix} AW^{-1}A' & AW^{-1}C' \\ CW^{-1}A' & CW^{-1}C' \end{bmatrix} \quad (25.22)$$

Suppose  $\tilde{\lambda} \geq 0$  and  $\tilde{\mu}$  are the current best solutions of the dual problem. Put them on top of each other to define  $\tilde{\gamma}$ , and do the same with  $\lambda$  and  $\mu$  to get  $\gamma$ . Then  $g(\lambda, \mu)$  becomes

$$\frac{1}{2}(\tilde{\gamma} + (\gamma - \tilde{\gamma}))'E(\tilde{\gamma} + (\gamma - \tilde{\gamma})) + \gamma'(Ry - e) = \frac{1}{2}(\gamma - \tilde{\gamma})'E(\gamma - \tilde{\gamma}) + (\gamma - \tilde{\gamma})'E(\tilde{\gamma} + (Ry - e)) + \frac{1}{2}\tilde{\gamma}'E\tilde{\gamma} + \tilde{\gamma}'(Ry - e) \quad (25.23)$$

The last two terms do not depend on  $\gamma$ , so for the majorization algorithm it suffices to minimize

$$\frac{1}{2}(\gamma - \tilde{\gamma})'F(\gamma - \tilde{\gamma}) + (\gamma - \tilde{\gamma})'E(\tilde{\gamma} + (Ry - e)) \quad (25.24)$$

Let

$$\xi := \tilde{\gamma} - F^{-1}E(\tilde{\gamma} + (Ry - e)) \quad (25.25)$$

then (25.24) becomes

$$\frac{1}{2}(\gamma - \xi)'F(\gamma - \xi) - \frac{1}{2}\xi'F\xi \quad (25.26)$$

Because  $F$  is diagonal  $\lambda_i = \max(0, \xi_i)$  for  $i = 1, \dots, m_1$  and  $\mu_i = \xi_{i+m_1}$  for  $i = 1, \dots, m_2$ .

Section A.1.18 has the R code for `qpmaj()`. The defaults are set to do a simple isotone regression, but of course the function has a much larger scope. It can handle equality constraints, linear convexity constraints, partial orders, and much more general linear inequalities. It can fit polynomials, monotone polynomials, splines, and monotone splines of various sorts. It is possible to have only inequality constraints, only equality constraints, or both. The matrix  $H$  of predictors in (25.13) can either be there or not be there.

The function `qpmaj()` returns both  $x$  and  $\lambda$ , and the values of  $\mathcal{P}$ ,  $\mathcal{P}'$ , and  $\mathcal{D}'$ . And also the *predicted values*  $Hx$ , and the *constraint values*  $Ax - b$  and  $Cx - d$ , if applicable. It's always nice to check *complimentary slackness*  $\lambda'(Ax - b) = 0$ , and another check is provided because the values of  $\mathcal{P}'$  and  $\mathcal{D}'$  must be equal. Finally `qpmaj()` returns the number of iterations for the dual problem.

The function `qpmaqj()` does not have the pretense to compete in efficiency with the sophisticated pivoting and active set strategies for quadratic programming discussed for example by Best (2017). But it seems to do a reliable job on our small examples, and it is an interesting example of majorization and unweighting.

### 25.7.1.1 Example 1: Simple Monotone Regression

Here are the two simple monotone regression examples from section 11.1.1, the first one without weights and the second one with a diagonal matrix of weights.

```
y<-c(1,2,1,3,2,-1,3)
qpmaj(y)
```

```
$x
```

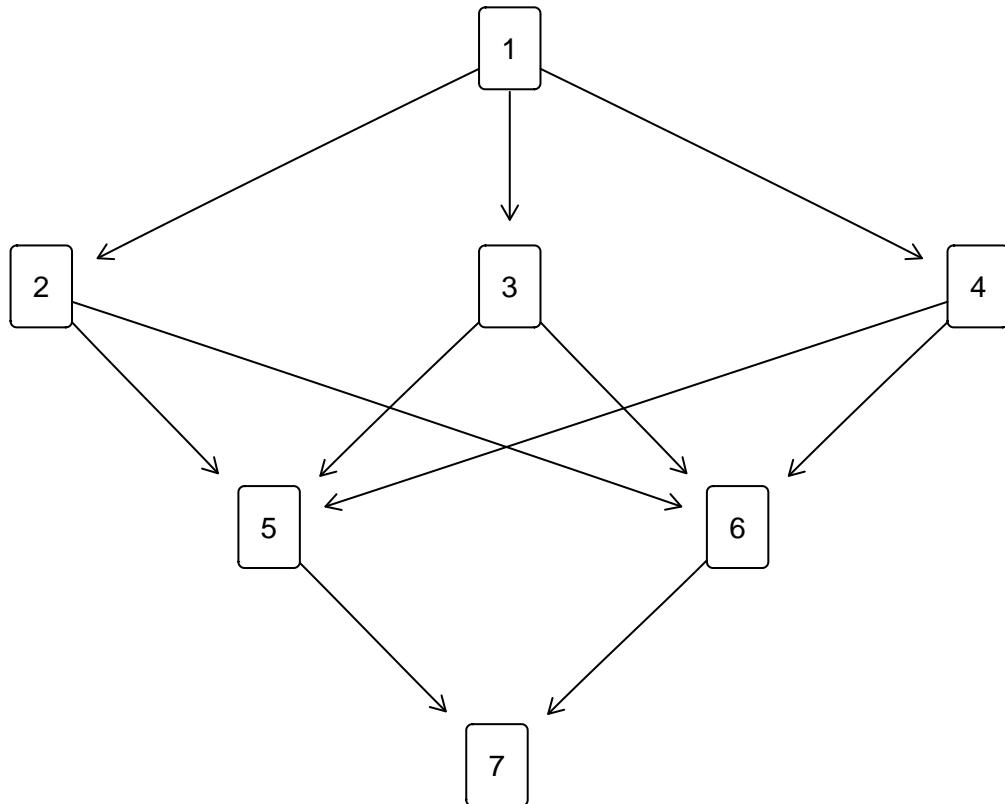
```
[1] 1.0 1.4 1.4 1.4 1.4 1.4 1.4 3.0
##
$fprimal
[1] 4.6
##
$fdual
[1] 4.6
##
$lambda
[1] 0.0000000 0.5999999 0.1999999 1.7999999 2.3999999 0.0000000
##
$inequalities
[1] 4.000001e-01 -2.760349e-08 -4.466339e-08 -4.466339e-08 -2.760349e-08
[6] 1.600000e+00
##
$itel
[1] 146

qpmaj(y, v = diag(c(1,2,3,4,3,2,1)))

$x
[1] 1.000000 1.400000 1.400000 1.777778 1.777778 1.777778 3.000000
##
$fprimal
[1] 11.37778
##
$fdual
[1] 11.37778
##
$lambda
[1] 0.000000 1.200000 0.000000 4.888889 5.555555 0.000000
##
$inequalities
[1] 4.000000e-01 0.000000e+00 3.777778e-01 -3.451610e-08 -2.391967e-08
[6] 1.222222e+00
##
$itel
[1] 82
```

### 25.7.1.2 Example 2: Monotone Regression with Ties

Now suppose the data have tie-blocks, which we indicate with  $\{1\} \leq \{2, 3, 4\} \leq \{5, 6\} \leq \{7\}$ . The Hasse diagram of the partial order (courtesy of Ciomek (2017)) is



In the primary approach to ties the inequality constraints  $Ax \geq 0$  are coded with  $A$  equal to

```

[,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -1 +1 +0 +0 +0 +0 +0
[2,] -1 +0 +1 +0 +0 +0 +0
[3,] -1 +0 +0 +1 +0 +0 +0
[4,] +0 -1 +0 +0 +1 +0 +0
[5,] +0 -1 +0 +0 +0 +1 +0

```

```

[6,] +0 +0 -1 +0 +1 +0 +0
[7,] +0 +0 -1 +0 +0 +1 +0
[8,] +0 +0 +0 -1 +1 +0 +0
[9,] +0 +0 +0 -1 +0 +1 +0
[10,] +0 +0 +0 +0 -1 +0 +1
[11,] +0 +0 +0 +0 +0 -1 +1

```

Applying our algorithm gives

```
qpmaj(y, a = a)
```

```

$x
[1] 1.000000 1.333333 1.000000 1.333333 2.000000 1.333333 3.000000
##
$fprimal
[1] 4.333333
##
$fdual
[1] 4.333333
##
$lambda
[1] 0.000000e+00 2.069906e-15 0.000000e+00 0.000000e+00 6.666667e-01
[6] 0.000000e+00 0.000000e+00 0.000000e+00 1.666667e+00 0.000000e+00
[11] 0.000000e+00
##
$inequalities
[1] 3.333333e-01 4.107825e-15 3.333334e-01 6.666667e-01 8.182895e-08
[6] 1.000000e+00 3.333333e-01 6.666666e-01 -8.182895e-08 1.000000e+00
[11] 1.666667e+00
##
$itel
[1] 96

```

In the secondary approach we require  $Cx = 0$ , with  $C$  equal to

```
[,1] [,2] [,3] [,4] [,5] [,6] [,7]
```

```
[1,] +0 +1 -1 +0 +0 +0 +0
[2,] +0 +1 +0 -1 +0 +0 +0
[3,] +0 +0 +0 +0 +1 -1 +0
```

In addition we construct  $A$  to require  $x_1 \leq x_2 \leq x_5 \leq x_7$ . This gives

```
qpmaj(y, a = a, c = c)
```

```
$x
[1] 1.0 1.4 1.4 1.4 1.4 1.4 3.0
##
$fprimal
[1] 4.6
##
$fdual
[1] 4.6
##
$lambda
[1] 0.0 1.8 0.0
##
$inequalities
[1] 4.000000e-01 -5.100425e-08 1.600000e+00
##
$mu
[1] -0.4 1.6 -2.4
##
$equations
[1] -2.055633e-08 -2.055633e-08 3.443454e-08
##
$itel
[1] 163
```

In the tertiary approach, without weights, we require  $x_1 \leq \frac{x_2+x_3+x_4}{3} \leq \frac{x_5+x_6}{2} \leq x_7$  which means

```
a <- matrix(c(-1, 1/3, 1/3, 1/3, 0, 0, 0,
 0, -1/3, -1/3, -1/3, 1/2, 1/2, 0,
 0, 0, 0, 0, -1/2, -1/2, 1),
 3, 7, byrow = TRUE)
matrixPrint(a, d = 2, w = 5)

[,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,] -1.00 +0.33 +0.33 +0.33 +0.00 +0.00 +0.00
[2,] +0.00 -0.33 -0.33 -0.33 +0.50 +0.50 +0.00
[3,] +0.00 +0.00 +0.00 +0.00 -0.50 -0.50 +1.00
```

This gives

```
qpmaj(y, a = a)

$x
[1] 1.0 1.4 0.4 2.4 2.9 -0.1 3.0
##
$fprimal
[1] 1.35
##
$fdual
[1] 1.35
##
$lambda
[1] 0.0 1.8 0.0
##
$inequalities
[1] 4.000000e-01 -1.716935e-08 1.600000e+00
##
$itel
[1] 30
```

### 25.7.1.3 Example 3: Weighted Rounding

This is a silly example in which a vector  $y = 0.5855288, 0.709466, -0.1093033, -0.4534972, 0.6058875, -1.817956, 0.6300986, -0.2761841, -0.1093033, -0.4534972, 0.6058875, -1.817956, 0.6300986, -0.2761841$

$0.2841597, -0.919322$  is “rounded” so that its elements are between  $-1$  and  $+1$ . The weights  $V = W$  are a banded positive definite matrix.

```
a<-rbind(-diag(10),diag(10))
b<-rep(-1, 20)
w<-ifelse(outer(1:10,1:10,function(x,y) abs(x-y) < 4), -1, 0)+7*diag(10)
qpmaj(y, v = w, a = a, b = b)

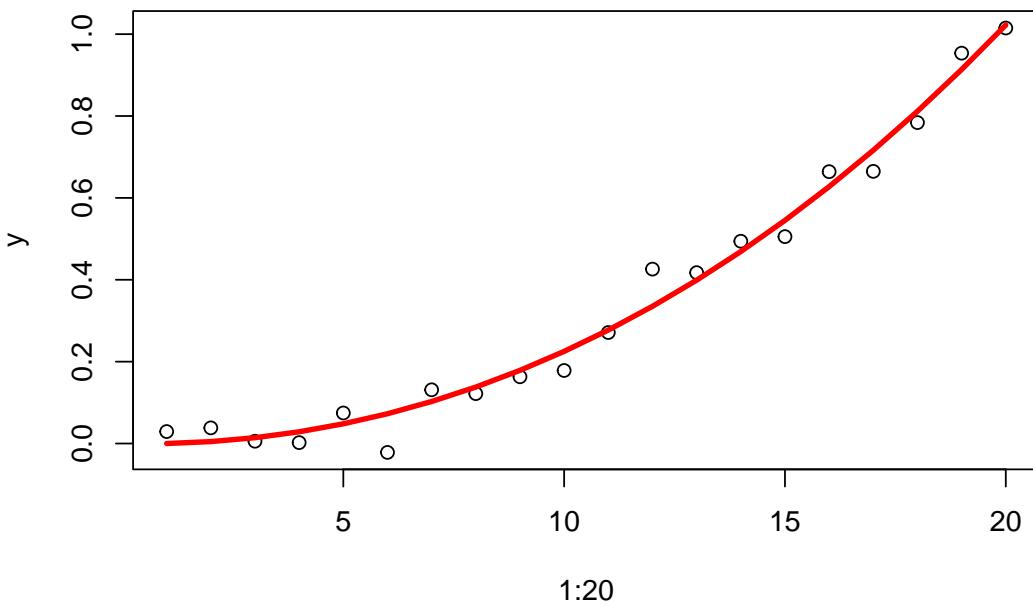
$x
[1] 0.737345533 0.917584527 0.215666042 -0.075684750 1.000000017
[6] -1.000000039 1.000000023 0.061944776 -0.002332657 -0.754345762
##
$fprimal
[1] 1.110748
##
$fdual
[1] 1.110749
##
$lambda
[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0722112 0.0000000 0.1554043
[8] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[15] 0.0000000 2.8209838 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##
$inequalities
[1] 2.626545e-01 8.241547e-02 7.843340e-01 1.075685e+00 -1.735205e-08
[6] 2.000000e+00 -2.303727e-08 9.380552e-01 1.002333e+00 1.754346e+00
[11] 1.737346e+00 1.917585e+00 1.215666e+00 9.243153e-01 2.000000e+00
[16] -3.922610e-08 2.000000e+00 1.061945e+00 9.976673e-01 2.456542e-01
##
$itel
[1] 224
```

#### 25.7.1.4 Example 4: Monotone Polynomials

This example has a matrix  $H$  with the monomials of degree 1, 2, 3 on the 20 points  $1, \dots, 20$ . We want to fit a third-degree polynomial which is monotone, non-negative, and anchored at zero (which is why we do not have a monomial

of degree zero, i.e. an intercept). Monotonicity is imposed by  $(h_{i+1} - h_i)'x \geq 0$  and non-negativity by  $h_1'x \geq 0$ . Thus there are  $19 + 1$  inequality restrictions. For  $y$  we choose points on the quadratic curve  $y = x^2$ , perturbed with random error.

```
set.seed(12345)
h <- cbind(1:20, (1:20)^2, (1:20)^3)
a <- rbind(h[1,], diff(diag(20)) %*% h)
y <- seq(0, 1, length=20)^2 + rnorm(20)/20
plot(1:20, y)
out <- qpmaj(y, a=a, h=h, verbose=FALSE, itmax=1000, eps = 1e-15)
lines(1:20, out$pred, type="l", lwd=3, col="RED")
```



The plot above and the output below shows what `qpmaj()` does in this case.

```
$x
[1] -2.311264e-03 2.292295e-03 1.895686e-05
##
$fprimal
[1] 0.0003265426
##
```

```

$fdual
[1] 0.0003265446
##
$ftotal
[1] 0.01655943
##
$predict
[1] -1.255287e-08 4.698305e-03 1.420869e-02 2.864490e-02 4.812065e-02
[6] 7.274970e-02 1.026458e-01 1.379226e-01 1.786940e-01 2.250737e-01
[11] 2.771753e-01 3.351127e-01 3.989996e-01 4.689497e-01 5.450767e-01
[16] 6.274945e-01 7.163167e-01 8.116571e-01 9.136294e-01 1.022347e+00
##
$lambda
[1] 0.1587839 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[8] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[15] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
##
$inequalities
[1] -1.255287e-08 4.698318e-03 9.510389e-03 1.443620e-02 1.947576e-02
[6] 2.462905e-02 2.989609e-02 3.527686e-02 4.077138e-02 4.637964e-02
[11] 5.210164e-02 5.793738e-02 6.388687e-02 6.995009e-02 7.612706e-02
[16] 8.241776e-02 8.882221e-02 9.534040e-02 1.019723e-01 1.087180e-01
##
$itel
[1] 97

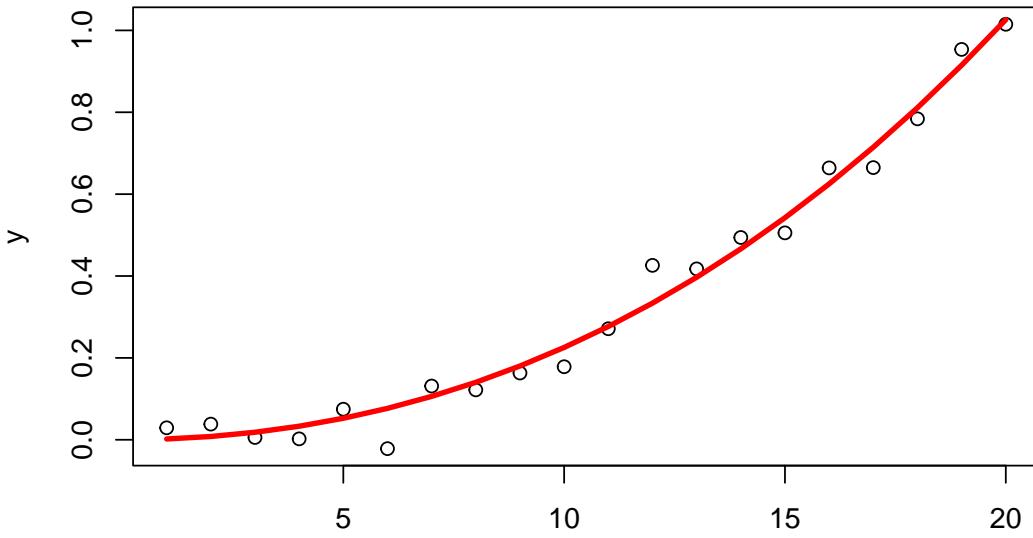
```

We now want to accomplish more or less the same thing, but using a cubic of the form  $f(x) = x(c + bx + ax^2)$ . Choosing  $a, b$  and  $c$  to be nonnegative guarantees monotonicity (and convexity) on the positive axis, with a root at zero. If  $b^2 \geq 4ac$  then the cubic has two additional real roots, and by AM/GM we can guarantee this by  $b \geq a + c$ . So  $a \geq 0$ ,  $c \geq 0$ , and  $b \geq a + c$  are our three inequalities.

```

h <- cbind(1:20,(1:20)^2,(1:20)^3)
a <- matrix(c(1,0,0,0,0,1,-1,1,-1), 3, 3, byrow = TRUE)
plot(1:20, y)
out<-qpmaj(y,a=a,h=h,verbose=FALSE,itmax=10000, eps = 1e-15)
lines(1:20,out$pred,type="l",lwd=3,col="RED")

```



1:20

The results of this alternative way of fitting the cubic are more or less indistinguishable from the earlier results, although this second approach is quite a bit faster (having only three inequalities instead of 21).

```
$x
[1] -5.673813e-09 1.945808e-03 3.098195e-05
##
$fprimal
[1] 0.0007170899
##
$fdual
[1] 0.000717091
##
$ftotal
[1] 0.01694997
##
$predict
[1] 0.001976784 0.008031075 0.018348765 0.033115745 0.052517907 0.076741142
[7] 0.105971343 0.140394402 0.180196208 0.225562656 0.276679635 0.333733038
[13] 0.396908757 0.466392682 0.542370707 0.625028722 0.714552619 0.811128290
[19] 0.914941626 1.026178519
##
```

```

$lambda
[1] 0.2021458 0.0000000 0.0000000
##
$inequalities
[1] -5.673813e-09 3.098195e-05 1.914831e-03
##
$itel
[1] 25

```

## 25.8 Quadratic penalties

Suppose  $\mathcal{X} \subseteq \mathbb{R}^n$  and  $f : \mathbb{R}^n \Rightarrow \mathbb{R}$  is continuous. Define

$$\mathcal{X}_* = \operatorname{argmin}_{x \in \mathcal{X}} f(x)$$

Suppose  $\mathcal{X}_*$  is non-empty and that  $x_*$  is any element of  $\mathcal{X}_*$ , and

$$f_* = f(x_*) = \min_{x \in \mathcal{X}} f(x).$$

The following convergence analysis of external linear penalty methods is standard and can be found in many texts (for example, Zangwill (1969), section 12.2).

The penalty term  $g : \mathbb{R}^n \Rightarrow \mathbb{R}^+$  is continuous and satisfies  $g(x) = 0$  if and only if  $x \in \mathcal{X}$ . For each  $\lambda > 0$  we define the (linear, external) penalty function

$$h(x, \lambda) = f(x) + \lambda g(x). \quad (25.27)$$

Suppose  $\{\lambda_k\}$  is a strictly increasing sequence of positive real numbers. Define

$$\mathcal{X}_k = \operatorname{argmin}_{x \in \mathcal{X}} h(x, \lambda_k). \quad (25.28)$$

Suppose all  $\mathcal{X}_k$  are nonempty and contained in a compact subset of  $\mathcal{X}$ . Choose  $x_k \in \mathcal{X}_k$  arbitrarily.

**Lemma 25.1.** 1:  $h(x_k, \lambda_k) \leq h(x_{k+1}, \lambda_{k+1})$ .

2:  $g(x_k) \geq g(x_{k+1})$ .

3:  $f(x_k) \leq f(x_{k+1})$ .

4:  $f_\star \geq h(x_k, \lambda_k) \geq f(x_k)$ .

*Proof.* 1: We have the chain

$$h(x_{k+1}, \lambda_{k+1}) = f(x_{k+1}) + \lambda_{k+1}g(x_{k+1}) \geq f(x_{k+1}) + \lambda_k g(x_{k+1}) \geq f(x_k) + \lambda_k g(x_k) = h(x_k, \lambda_k).$$

2: Both

$$f(x_k) + \lambda_k g(x_k) \leq f(x_{k+1}) + \lambda_k g(x_{k+1}), \quad (25.29)$$

$$f(x_{k+1}) + \lambda_{k+1}g(x_{k+1}) \leq f(x_k) + \lambda_{k+1}g(x_k). \quad (25.30)$$

Adding inequalities (25.29) and (25.30) gives

$$\lambda_k g(x_k) + \lambda_{k+1}g(x_{k+1}) \leq \lambda_k g(x_{k+1}) + \lambda_{k+1}g(x_k),$$

or

$$(\lambda_k - \lambda_{k+1})g(x_k) \leq (\lambda_k - \lambda_{k+1})g(x_{k+1}),$$

and thus  $g(x_k) \geq g(x_{k+1})$ .

3: First

$$f(x_{k+1}) + \lambda_k g(x_{k+1}) \geq f(x_k) + \lambda_k g(x_k). \quad (25.31)$$

We just proved that  $g(x_{k+1}) \geq g(x_k)$ , and thus

$$f(x_k) + \lambda_k g(x_k) \geq f(x_k) + \lambda_k g(x_{k+1}). \quad (25.32)$$

Combining inequalities (25.31) and (25.32) gives  $f(x_{k+1}) \geq f(x_k)$ .

4: We have the chain

$$f_\star = f(x_\star) + \lambda_k g(x_\star) \geq f(x_k) + \lambda_k g(x_k) \geq f(x_k).$$

□

“‘{theorem, label = “lbdconverge”}) Suppose the sequence  $\{\lambda_k\}_{k \in K}$  diverges to  $\infty$  and  $x_{**}$  is the limit of any convergent subsequence  $\{x_\ell\}_{\ell \in L}$ . Then  $x_{**} \in \mathcal{X}_*$ , and  $f(x_{**}) = f_\star$ , and  $g(x_{**}) = 0$ .

*Proof.* Using part 4 of lemma XXX

$$\lim_{\ell \in L} h(x_\ell, \lambda_\ell) = \lim_{\ell \in L} \{f(x_\ell) + \lambda_\ell g(x_\ell)\} = f(x_{**}) + \lim_{\ell \in L} \lambda_\ell g(x_\ell) \leq f(x_*).$$

Thus  $\{h(x_\ell, \lambda_\ell)\}_{\ell \in L}$  is a bounded increasing sequence, which consequently converges, and  $\lim_{\ell \in L} \lambda_\ell g(x_\ell)$  also converges. Since  $\{\lambda_\ell\}_{\ell \in L} \rightarrow \infty$  it follows that  $\lim_{\ell \in L} g(x_\ell) = g(x_{**}) = 0$ . Thus  $x_{**} \in \mathcal{X}$ . Since  $f(x_\ell) \leq f_*$  we see that  $f(x_{**}) \leq f_*$ , and thus  $x_{**} \in \mathcal{X}_*$  and  $f(x_{**}) = f_*$ .  $\square$

# Appendix A

## Code

### A.1 R Code

The MDS functions in R throughout work with square matrices of weights, dissimilarities, and distances. More efficient versions, that have their computations done in C, will be added along the way.

#### A.1.1 utilities.R

```
source ("common/indexing.R")
source ("common/io.R")
source ("common/linear.R")
source ("common/nextPC.R")
source ("common/smacof.R")
```

#### A.1.2 commom/indexing.r

```
kron <- function (i, j) {
 return (ifelse (i == j, 1, 0))
}
```

```
ein <- function (i, n) {
 return (ifelse (i == 1:n, 1, 0))
}

aijn <- function (i, j, n) {
 dif <- ein (i, n) - ein (j, n)
 return (outer (dif, dif))
}

jmat <- function (n) {
 return (diag(n) - 1 / n)
}

ccen <- function (x) {
 return (apply (x, 2, function (y)
 y - mean (y)))
}

repList <- function(x, n) {
 z <- list()
 for (i in 1:n)
 z <- c(z, list(x))
 return(z)
}

rcen <- function (x) {
 return (t (apply (x, 1, function (y)
 y - mean (y))))
}

dcen <- function (x) {
 return (ccen (rcen (x)))
}

wdef <- function (n) {
 return (1 - diag (n))
```

```

}

nonDiag <- function (n) {
 return (matrix (1, n, n) - diag (n))
}

lower_triangle <- function (x) {
 n <- nrow (x)
 return (x[outer(1:n, 1:n, ">")])
}

fill_symmetric <- function (x) {
 m <- length (x)
 n <- (1 + sqrt (1 + 8 * m)) / 2
 d <- matrix (0, n, n)
 d[outer(1:n, 1:n, ">")] <- x
 return (d + t(d))
}

```

### A.1.3 commom/io.r

```

matrixPrint <- function (x,
 digits = 6,
 width = 8,
 format = "f",
 flag = "+") {
 print (noquote (formatC (
 x,
 digits = digits,
 width = width,
 format = format,
 flag = flag
)))
}
```

```

}

iterationWrite <- function (labels, values, digits, width, format) {
 for (i in 1:length(labels)) {
 cat (labels[i],
 formatC(
 values[i],
 di = digits[i],
 wi = width[i],
 fo = format[i]
),
 " ")
 }
 cat("\n")
}

rotateEllipse <- function (x) {
 z <- (x[1,] + x[2,]) / 2
 x <- x - matrix (z, nrow(x), 2, byrow = TRUE)
 s <- sqrt (sum (x[1,] ^ 2))
 r <- matrix(c(x[1, 1], x[1, 2], -x[1, 2], x[1, 1]), 2, 2) / s
 x <- x %*% r
 e <- as.matrix (dist (x))
 d <- mean (rowSums(e[-(1:2), 1:2]))
 a <- d / 2
 c <- abs (x[1, 1])
 b <- sqrt (a ^ 2 - c ^ 2)
 return (list(
 x = x,
 a = a,
 b = b,
 c = c
))
}

plotEllipse <- function (x) {
 r <- rotateEllipse (x)

```

```

f <- seq (0, 2 * pi, length = 100)
z <- cbind(sin (f), cos (f))
z[, 1] <- z[, 1] * r$a
z[, 2] <- z[, 2] * r$b
plot(z,
 type = "l",
 col = "RED",
 lwd = 2)
text (r$x, as.character (1:nrow(r$x)))
abline(h = 0)
abline(v = 0)
}

draw_ellipse <- function (center,
 radius,
 a = diag (2),
 np = 100,
 ...) {
 par (pty = "s")
 e <- eigen(a)
 k <- e$vectors
 lbd <- e$values
 seq <- seq(0, 2 * pi, length = np)
 scos <- (radius * sin (seq)) / lbd[1]
 ccos <- (radius * cos (seq)) / lbd[2]
 sico <- k %*% rbind(scos, ccos) + center
 plot (sico[1,], sico[2,], type = "l", ...)
}

```

#### A.1.4 commom/linear.r

```

kron <- function (i, j) {
 return (ifelse (i == j, 1, 0))
}

```

```
ein <- function (i, n) {
 return (ifelse (i == 1:n, 1, 0))
}

aijn <- function (i, j, n) {
 dif <- ein (i, n) - ein (j, n)
 return (outer (dif, dif))
}

jmat <- function (n) {
 return (diag(n) - 1 / n)
}

ccen <- function (x) {
 return (apply (x, 2, function (y)
 y - mean (y)))
}

repList <- function(x, n) {
 z <- list()
 for (i in 1:n)
 z <- c(z, list(x))
 return(z)
}

rcen <- function (x) {
 return (t (apply (x, 1, function (y)
 y - mean (y))))
}

dcen <- function (x) {
 return (ccen (rcen (x)))
}

wdef <- function (n) {
 return (1 - diag (n))
```

```

}

nonDiag <- function (n) {
 return (matrix (1, n, n) - diag (n))
}

lower_triangle <- function (x) {
 n <- nrow (x)
 return (x[outer(1:n, 1:n, ">")])
}

fill_symmetric <- function (x) {
 m <- length (x)
 n <- (1 + sqrt (1 + 8 * m)) / 2
 d <- matrix (0, n, n)
 d[outer(1:n, 1:n, ">")] <- x
 return (d + t(d))
}

```

### A.1.5 commom/nextPC.r

```

nextPermutation <- function (x) {
 if (all (x == (length(x):1)))
 return (NULL)
 z <- .C("nextPermutation", as.integer(x), as.integer(length(x)))
 return (z[[1]])
}

nextCombination <- function (x, n) {
 m <- length (x)
 if (all (x == ((n - m) + 1:m)))
 return (NULL)
 z <-

```

```
.C("nextCombination",
 as.integer(n),
 as.integer(m),
 as.integer(x))
return (z[[3]])
}
```

### A.1.6 commom/smacof.r

```
library(MASS)

smacofLossR <- function (d, w, delta) {
 return (sum (w * (delta - d) ^ 2) / 2)
}

smacofBmatR <- function (d, w, delta) {
 dd <- ifelse (d == 0, 0, 1 / d)
 b <- -dd * w * delta
 diag (b) <- -rowSums (b)
 return(b)
}

smacofVmatR <- function (w) {
 v <- -w
 diag(v) <- -rowSums(v)
 return (v)
}

smacofGuttmanR <- function (x, b, vinv) {
 return (vinv %*% b %*% x)
}

smacofGradientR <- function (x, b, v) {
 return (2 * ((v - b) %*% x))
}
```

```

smacofHmatR <- function (x, b, v, d, w, delta) {
 n <- nrow (x)
 p <- ncol (x)
 r <- n * p
 h <- matrix (0, r, r)
 dd <- ifelse (d == 0, 0, 1 / d)
 cc <- w * delta * (dd ^ 3)
 for (s in 1:p) {
 ns <- (s - 1) * n + 1:n
 for (t in 1:s) {
 nt <- (t - 1) * n + 1:n
 cst <- matrix (0, n, n)
 for (i in 1:n) {
 for (j in 1:n) {
 cst[i, j] <- cc[i, j] * (x[i, s] - x[j, s]) * (x[i, t] - x[j, t])
 }
 }
 cst <- -cst
 diag(cst) <- -rowSums(cst)
 if (s == t) {
 h[ns, ns] <- b - cst
 } else {
 h[ns, nt] <- -cst
 h[nt, ns] <- -cst
 }
 }
 }
 return (h)
}

smacofHessianR <- function (x, b, v, d, w, delta) {
 n <- nrow (x)
 p <- ncol (x)
 h <- -smacofHmatR (x, b, v, d, w, delta)
 for (s in 1:p) {
 nn <- (s - 1) * n + 1:n
 h[nn, nn] <- h[nn, nn] + v
 }
}

```

```

 }
 return(h)
}

smacofDerGuttmanR <- function(x, b, vinv, d, w, delta) {
 n <- nrow (x)
 p <- ncol (x)
 h <- smacofHmatR (x, b, v, d, w, delta)
 for (s in 1:p) {
 ns <- (s - 1) * n + 1:n
 for (t in 1:s) {
 nt <- (t - 1) * n + 1:n
 h[ns,nt] <- vinv %*% h[ns,nt]
 }
 }
 return(h)
}

smacofInitialR <- function (delta, p) {
 n <- nrow(delta)
 delta <- delta ^ 2
 rw <- rowSums (delta) / n
 sw <- sum (delta) / (n ^ 2)
 h <- -(delta - outer (rw, rw, "+") + sw) / 2
 e <- eigen (h)
 ea <- e$values
 ev <- e$vector
 ea <- ifelse (ea > 0, sqrt (abs(ea)), 0)[1:p]
 return (ev[, 1:p] %*% diag (ea))
}

smacofRandomStart <- function (w, delta, n, p) {
 x <- matrix(rnorm(n * p), n, p)
 x <- apply (x, 2, function(x) x - mean(x))
 d <- as.matrix(dist (x))
 a <- sum (w * delta * d) / sum (w * d ^ 2)
 return (a * x)
}

```

```

}

smacofR <-
 function (w,
 delta,
 p,
 xold = smacofInitialR(delta, p),
 xstop = FALSE,
 itmax = 1000,
 eps = 1e-10,
 verbose = TRUE) {
 labels = c("itel", "eiff", "sold", "snew")
 digits = c(4, 10, 10, 10)
 widths = c(6, 15, 15, 15)
 format = c("d", "f", "f", "f")
 itel <- 1
 dold <- as.matrix (dist (xold))
 sold <- smacofLossR (dold, w, delta)
 bold <- smacofBmatR (dold, w, delta)
 vmat <- smacofVmatR (w)
 vinv <- ginv (vmat)
 repeat {
 xnew <- smacofGuttmanR (xold, bold, vinv)
 dnew <- as.matrix (dist (xnew))
 bnew <- smacofBmatR (dnew, w, delta)
 snew <- smacofLossR (dnew, w, delta)
 if (xstop) {
 eiff <- max (abs (xold - xnew))
 } else {
 eiff <- sold - snew
 }
 if (verbose) {
 values = c(itel, eiff, sold, snew)
 iterationWrite (labels, values, digits, widths, format)
 }
 if ((eiff < eps) || (itel == itmax)) {
 break
 }
 }
}

```

```

 }
 itel <- itel + 1
 xold <- xnew
 bold <- bnew
 dold <- dnew
 sold <- snew
}
return (list (
 x = xnew,
 d = dnew,
 b = bnew,
 g = smacofGradientR(xnew, bnew, vmat),
 h = smacofHessianR(xnew, bnew, vmat, dnew, w, delta),
 s = snew,
 itel = itel
))
}

```

### A.1.7 properties.R

```

csupper <- function (lbd, mbd) {
 n <- length (lbd)
 m <- length (mbd)
 nad <- nad <- matrix (0, m, n)
 sad <- rep(0, n)
 plot(lbd,
 lbd,
 type = "n",
 ylab = "",
 ylim = c(0, 2))
 for (k in 1:m) {
 ly <- mbd[k]
 yy <- ly * z1 + (1 - ly) * z2
 dy <- dist (yy)
 by <- as.matrix(-delta / dy)
 }
}

```

```

diag (by) <- -rowSums(by)
for (l in 1:n) {
 xx <- lbd[l] * z1 + (1 - lbd[l]) * z2
 dx <- dist (xx)
 rx <- sum (xx * (by %*% yy))
 nx <- sum (dx ^ 2)
 mad[k, l] <- 1 - 2 * rx + nx
}
lines (lbd, mad [k,])
abline (v = ly)
}
lines(lbd,
 apply(mad, 2, min),
 type = "l",
 col = "BLUE",
 lwd = 3)
for (l in 1:n) {
 xx <- lbd[l] * z1 + (1 - lbd[l]) * z2
 dx <- dist (xx)
 sad[l] <- 1 - 2 * sum (dx * delta) + sum (dx ^ 2)
}
lines (lbd,
 sad,
 type = "l",
 col = "RED",
 lwd = 3)
}

aglower <- function (lbd, mbd) {
 n <- length (lbd)
 m <- length (mbd)
 mad <- matrix (0, m, n)
 sad <- rep(0, n)
 plot(lbd,
 lbd,
 type = "n",
 ylab = "",

```

```

 ylim = c(0, 2))
for (k in 1:m) {
 ly <- mbd[k]
 yy <- ly * z1 + (1 - ly) * z2
 dy <- dist (yy)
 ry <- sum (delta * dy)
 by <- as.matrix(-delta / dy)
 diag (by) <- -rowSums(by)
 for (l in 1:n) {
 xx <- lbd[l] * z1 + (1 - lbd[l]) * z2
 dx <- dist (xx)
 sx <- sum (xx * (by %*% xx))
 nx <- sum (dx ^ 2)
 mad[k, l] <- 1 - ry + nx - sx
 }
 lines (lbd, mad [k,])
 abline (v = ly)
}
lines(lbd,
 apply(mad, 2, max),
 type = "l",
 col = "BLUE",
 lwd = 3)
for (l in 1:n) {
 xx <- lbd[l] * z1 + (1 - lbd[l]) * z2
 dx <- dist (xx)
 sad[l] <- 1 - 2 * sum (dx * delta) + sum (dx ^ 2)
}
lines (lbd,
 sad,
 type = "l",
 col = "RED",
 lwd = 3)
}

```

### A.1.8 pictures.R

```

dommy <- function () {
 ya <- matrix(c(1, -1, 1, -1, 0, 1, 1, -1, -1, 0), 5, 2)
 yy <- seq (0, 2 * pi, length = 6)[1:5]
 yb <- cbind (sin (yy), cos (yy))
 ya <- apply(ya, 2, function (x)
 x - mean(x))
 yb <- apply(yb, 2, function (x)
 x - mean(x))
 y1 <- ya / sqrt (5 * sum (ya ^ 2))
 y2 <- yb / sqrt (5 * sum (yb ^ 2))
 deq <- as.dist (1 - diag(5))
 deq <- deq / sqrt (sum (deq ^ 2))
 y1 <- sum (dist (y1) * deq) * y1
 y2 <- sum (dist (y2) * deq) * y2
}

twostress <- function (deq, y1, y2, a, b) {
 d <- dist (a * y1 + b * y2)
 eta2 <- sum (d ^ 2)
 rho <- sum (d * deq)
 stress <- 1 - 2 * rho + eta2
 return(list(
 eta2 = eta2,
 rho = rho,
 stress = stress
))
}

zeroes <- function (y1, y2) {
 n <- nrow (y1)
 for (i in 2:n) {
 for (j in 1:(i - 1)) {
 yy <- rbind (y1[i,] - y1[j,], y2[i,] - y2[j,])
 ee <- eigen(tcrossprod(yy))$values
 print (c(i, j, ee))
 }
 }
}

```

```
 }
 }
}

pairme <- function (x, y) {
 n <- nrow(x)
 m <- ncol(x)
 z <- matrix (0, 2, m)
 for (i in 2:n) {
 for (j in 1:(i - 1)) {
 z[1,] <- x[i,] - x[j,]
 z[2,] <- y[i,] - y[j,]
 s <- svd (z)
 a <- s$d[2]
 b <- s$u[, 2]
 if (a < 1e-10) {
 cat (
 formatC(
 i,
 format = "d",
 digits = 2,
 width = 4
),
 formatC(
 j,
 format = "d",
 digits = 2,
 width = 4
),
 formatC(
 c(a, b),
 format = "f",
 digits = 6,
 width = 10
),
 "\n"
)
 }
 }
 }
}
```

```
 }
 }
}

dummy <- function () {
 set.seed(12345)
 x <- matrix(rnorm(10), 5, 2)
 x <- apply (x, 2, function(x)
 x - mean(x))
 delta <- dist(x)
 d <- dist (x)
 eps <- (-500:500) / 100
 sy <- rep (0, 1001)
 plot (
 0,
 0,
 xlim = c(-5, 5),
 ylim = c(0, 20),
 xlab = "epsilon",
 ylab = "stress",
 type = "n"
)
 for (i in 1:5) {
 for (j in 1:2) {
 for (k in 1:1001) {
 y <- x
 y[i, j] <- x[i, j] + eps[k]
 dy <- dist (y)
 sy[k] <- sum ((delta - dy) ^ 2)
 }
 lines (eps, sy, lwd = 2, col = "RED")
 }
 }
}
```

```

bmat2 <- function (a, b, x, y, delta) {
 bm <- matrix (0, 2, 2)
 hm <- matrix (0, 2, 2)
 z <- c(a, b)
 for (i in 1:4) {
 for (j in 1:4) {
 if (i == j)
 next
 uij <- uu (i, j, x, y)
 uz <- drop (uij %*% z)
 dij <- sqrt (sum (uij * outer (z, z)))
 bm <- bm + (delta[i, j] / dij) * uij
 hm <-
 hm + (delta[i, j] / dij) * (uij - outer (uz, uz) / sum (z * uz))
 }
 }
 return (list (b = bm, h = hm))
}

stress2 <- function (a, b, x, y, delta) {
 z <- c (a, b)
 bm <- bmat2 (a, b, x, y, delta)$b
 return (1 + sum(z ^ 2) / 2 - sum (z * bm %*% z))
}

rho2 <- function (a, b, x, y, delta) {
 z <- c (a, b)
 bm <- bmat2 (a, b, x, y, delta)$b
 return (sum (z * bm %*% z))
}

vv <- function (i, j, x, y) {
 a <- matrix (0, 2, 2)
 a[1, 1] <- sum ((x[i,] - x[j,]) ^ 2)
 a[2, 2] <- sum ((y[i,] - y[j,]) ^ 2)
 a[1, 2] <- a[2, 1] <- sum ((x[i,] - x[j,]) * (y[i,] - y[j,]))
 return (a)
}

```

```

}

uu <- function (i, j, x, y) {
 n <- nrow (x)
 asum <-
 2 * n * matrix (c (sum(x ^ 2), sum (x * y), sum (x * y), sum (y ^ 2)), 2, 2)
 csum <- solve (chol (asum))
 return (t(csum) %*% vv (i, j, x, y) %*% csum)
}

smacof2 <-
 function (a,
 b,
 x,
 y,
 delta,
 eps = 1e-10,
 itmax = 1000,
 verbose = TRUE) {
 zold <- c(a, b)
 bold <- bmat2 (a, b, x, y, delta)$b
 fold <- 1 + sum(zold ^ 2) / 2 - sum (zold * bold %*% zold)
 itel <- 1
 repeat {
 znew <- drop (bold %*% zold)
 bhmt <- bmat2 (znew[1], znew[2], x, y, delta)
 bnew <- bhmt$b
 fnew <- 1 + sum(znew ^ 2) / 2 - sum (znew * bnew %*% znew)
 if (verbose) {
 cat (
 formatC (itel, width = 4, format = "d"),
 formatC (
 fold,
 digits = 10,
 width = 13,
 format = "f"
),
 ...
)
 }
 }
}

```

```

 formatC (
 fnew,
 digits = 10,
 width = 13,
 format = "f"
),
 "\n"
)
 }
 if ((itel == itmax) || (fold - fnew) < eps)
 break ()
 itel <- itel + 1
 fold <- fnew
 zold <- znew
 bold <- bnew
 }
 return (
 list (
 stress = fnew,
 theta = znew,
 itel = itel,
 b = bnew,
 g = znew - bnew %*% znew,
 h = diag(2) - bhmt$h
)
)
}

newton2 <-
 function (a,
 b,
 x,
 y,
 delta,
 eps = 1e-10,
 itmax = 1000,

```

```

 verbose = TRUE) {
zold <- c(a, b)
bhmt <- bmat2 (a, b, x, y, delta)
bold <- bhmt$b
hold <- diag(2) - bhmt$h
fold <- 1 + sum(zold ^ 2) / 2 - sum (zold * bold %*% zold)
itel <- 1
repeat {
 znew <- drop (solve (hold, bold %*% zold))
 bhmt <- bmat2 (znew[1], znew[2], x, y, delta)
 bnew <- bhmt$b
 hnew <- diag(nrow(bnew)) - bhmt$h
 fnew <- 1 + sum(znew ^ 2) / 2 - sum (znew * bnew %*% znew)
 if (verbose) {
 cat (
 formatC (itel, width = 4, format = "d"),
 formatC (
 fold,
 digits = 10,
 width = 13,
 format = "f"
),
 formatC (
 fnew,
 digits = 10,
 width = 13,
 format = "f"
),
 "\n"
)
 }
 if ((itel == itmax) || abs (fold - fnew) < eps)
 break ()
 itel <- itel + 1
 fold <- fnew
 zold <- znew
 bold <- bnew
}

```

```

 hold <- hnew
}
return (list (
 stress = fnew,
 theta = znew,
 itel = itel,
 b = bnew,
 g = znew - bnew %*% znew,
 h = hnew
))
}
```

### A.1.9 classical.R

```

tau <- function (x) {
 return (- 0.5 * dcen (x))
}

kappa <- function (x) {
 return (outer (diag (x), diag (x), "+") - 2 * x)
}

fcmds <-
 function (delta,
 xold,
 ninner = 1,
 itmax = 100,
 eps = 1e-6,
 verbose = TRUE) {
 itel <- 0
 p <- ncol (xold)
 xold <- apply (xold, 2, function (x)
 x - mean(x))
 xold <- qr.Q (qr (xold))
 repeat {
```

```

xinn <- xold
for (i in 1:ninner) {
 xnew <- delta %*% xinn
 xnew <- -apply (xnew, 2, function (x)
 x - mean (x)) / 2
 xinn <- xnew
 itel <- itel + 1
}
qnew <- qr (xnew)
xnew <- qr.Q (qnew)
rnew <- qr.R (qnew)
epsi <- 2 * p - 2 * sum (svd (crossprod (xold, xnew))$d)
if (verbose) {
 cat(
 "itel ",
 formatC (
 itel,
 digits = 4,
 width = 6,
 format = "d"
),
 "epsi ",
 formatC (
 epsi,
 digits = 10,
 width = 15,
 format = "f"
),
 "\n"
)
}
if ((epsi < eps) || (itel == itmax))
 break
xold <- xnew
}
return (list (x = xnew, r = rnew, itel = itel))
}

```

```

treq <- function (x) {
 n <- nrow (d)
 m <- -Inf
 for (i in 2:n) {
 for (j in 1:(i - 1)) {
 for (k in 1:n) {
 if ((k == i) || (k == j))
 next
 m <- max (m, x[i, j] - (x[i, k] + x[k, j]))
 }
 }
 }
 return (m)
}

acbound <- function (d) {
 n <- nrow (d)
 s <- qr.Q (qr (cbind (1, matrix (rnorm (
 n * (n - 1)
), n, n - 1))))
 k <- tau (d * d)
 l <- 2 * tau (d)
 m <- jmat (n) / 2
 ma <- -Inf
 for (i in 2:n)
 for (j in 1:(i - 1)) {
 v <- solve(polynomial(c(k[i, j], l[i, j], m[i, j])))
 ma <- max(ma, max(v))
 }
 return (list(ma = ma, mw = k + ma * l + m * ma ^ 2))
}

aceval <- function (d, bnd = c(-10, 10)) {
 n <- nrow (d)
 k <- tau (d * d)
 l <- 2 * tau (d)
 m <- jmat (n) / 2
}

```

```

s <- qr.Q (qr (cbind (1, matrix (rnorm (
 n * (n - 1)
), n, n - 1))))
kc <- (crossprod (s, k) %*% s)[-1,-1]
lc <- (crossprod (s, l) %*% s)[-1,-1]
mc <- (crossprod (s, m) %*% s)[-1,-1]
a <- seq(bnd[1], bnd[2], length = 1000)
b <- rep(0, 1000)
for (i in 1:1000) {
 ww <- kc + lc * a[i] + mc * (a[i] ^ 2)
 b[i] <- min (eigen(ww)$values)
}
return (list(a = a, b = b))
}

acqep <- function(d) {
 n <- nrow (d)
 k <- tau (d * d)
 l <- 2 * tau (d)
 m <- jmat (n) / 2
 s <- qr.Q (qr (cbind (1, matrix (rnorm (
 n * (n - 1)
), n, n - 1))))
 nn <- n - 1
 ns <- 1:nn
 kc <- (crossprod (s, k) %*% s)[-1,-1]
 lc <- (crossprod (s, l) %*% s)[-1,-1]
 mc <- (crossprod (s, m) %*% s)[-1,-1]
 ma <- matrix(0, 2 * nn, 2 * nn)
 ma[ns, nn + ns] <- diag (n - 1)
 ma[nn + ns, ns] <- -2 * kc
 ma[nn + ns, nn + ns] <- -2 * lc
 return (list (ma = ma, me = eigen(ma)$values))
}

```

### A.1.10 minimization.R

### A.1.11 full.R

```

library(MASS)

torgerson <- function(delta, p = 2) {
 doubleCenter <- function(x) {
 n <- dim(x)[1]
 m <- dim(x)[2]
 s <- sum(x) / (n * m)
 xr <- rowSums(x) / m
 xc <- colSums(x) / n
 return((x - outer(xr, xc, "+")) + s)
 }
 z <-
 eigen(-doubleCenter((as.matrix(delta) ^ 2) / 2), symmetric = TRUE)
 v <- pmax(z$values, 0)
 return(z$vectors[, 1:p] %*% diag(sqrt(v[1:p])))
}

makeA <- function (n) {
 m <- n * (n - 1) / 2
 a <- list()
 for (j in 1:(n - 1))
 for (i in (j + 1):n) {
 d <- ein (i, n) -ein (j, n)
 e <- outer (d, d)
 a <- c(a, list (e))
 }
 return (a)
}

makeD <- function (a, x) {
 return (sapply (a, function (z)
 sqrt (sum (x * (

```

```

 z %*% x
)))))
}

makeB <- function (w, delta, d, a) {
 n <- length (a)
 m <- nrow (a[[1]])
 b <- matrix (0, m , m)
 for (i in 1:n)
 b <- b + w[i] * (delta[i] / d[i]) * a[[i]]
 return (b)
}

makeV <- function (w, a) {
 n <- length (a)
 m <- nrow (a[[1]])
 v <- matrix (0, m, m)
 for (i in 1:n)
 v <- v + w[i] * a[[i]]
 return (v)
}

inBetween <- function (alpha, beta, x, y, w, delta, a) {
 z <- alpha * x + beta * y
 d <- makeD (a, z)
 return (sum (w * (delta - d) ^ 2))
}

biBase <- function (x, y, a) {
 biBi <- function (x, y, v) {
 a11 <- sum (x * (v %*% x))
 a12 <- sum (x * (v %*% y))
 a22 <- sum (y * (v %*% y))
 return (matrix (c(a11, a12, a12, a22), 2, 2))
 }
 return (lapply (a, function (u)
 biBi (x, y, u)))
}

```

```

}

fullMDS <-
 function (delta,
 w = rep (1, length (delta)),
 xini,
 a,
 itmax = 100,
 eps = 1e-6,
 verbose = TRUE) {
 m <- length (a)
 v <- makeV (w, a)
 vv <- ginv (v)
 xold <- xini
 dold <- makeD (a, xini)
 sold <- sum ((delta - dold) ^ 2)
 bold <- makeB (w, delta, dold, a)
 itel <- 1
 repeat {
 xnew <- vv %*% bold %*% xold
 dnew <- makeD (a, xnew)
 bnew <- makeB (w, delta, dnew, a)
 snew <- sum ((delta - dnew) ^ 2)
 if (verbose) {
 cat (
 formatC (itel, width = 4, format = "d"),
 formatC (
 sold,
 digits = 10,
 width = 13,
 format = "f"
),
 formatC (
 snew,
 digits = 10,
 width = 13,
 format = "f"
)
)
 }
 }
}

```

```

),
"\n"
)
}
if ((itel == itmax) || (abs(sold - snew) < eps))
 break
itel <- itel + 1
xold <- xnew
dold <- dnew
sold <- snew
bold <- bnew
}
return (list (
 x = xnew,
 d = dnew,
 delta = delta,
 s = snew,
 b = bnew,
 v = v,
 itel = itel
))
}
```

### A.1.12 unfolding.R

```

dummy <- function () {
 set.seed(12345)

 x <- matrix (rnorm(16), 8, 2)
 x <- apply (x, 2, function (x)
 x - mean (x))
 y <- matrix (rnorm(10), 5, 2)
 a <- rowSums (x ^ 2)
 b <- rowSums (y ^ 2)
 d <- sqrt (outer(a, b, "+") - 2 * tcrossprod (x, y))
```

```

set.seed (12345)
x <- matrix(rnorm(10), 5, 2)
x <- apply (x, 2, function (x)
 x - mean (x))
x <- qr.Q(qr(x))
y <- matrix(rnorm(12), 6, 2)
v <- apply (y, 2, mean)
print (v)
dx <- diag(tcrossprod(x))
dy <- diag(tcrossprod(y))
xy <- tcrossprod(x, y)
dd <- outer(dx, dy, "+") - 2 * xy
j5 <- diag(5) - 1 / 5
j6 <- diag(6) - 1 / 6
dc <- -(j5 %*% dd %*% j6) / 2
sv <- svd (dc, nu = 2, nv = 2)
xs <- sv$u
ys <- sv$v %*% diag (sv$d[1:2])
tt <- crossprod (x, xs)
dk <- diag (tcrossprod(xs))
dl <- diag (tcrossprod(ys))
dr <- dd - outer (dk, dl, "+") + 2 * tcrossprod (xs, ys)
print (dr)
}

schoenemann <- function (delta, p) {
 n <- nrow (delta)
 m <- ncol (delta)
 l <- p * (p + 1) / 2
 d <- delta ^ 2
 e <- torgerson (d)
 q <- svd (e, nu = p, nv = p)
 g <- q$u
 h <- q$v %*% diag (q$d[1:p])
 f <- d + 2 * tcrossprod(g, h)
 a <- apply (ccen (f), 1, mean)
 r <- matrix (0, n, 1)
}

```

```

k <- 1
for (i in 1:p) {
 for (j in 1:i) {
 if (i == j) {
 r[, k] <- g[, i] ^ 2
 } else {
 r[, k] <- 2 * g[, i] * g[, j]
 }
 k <- k + 1
 }
}
lhs <- cbind (ccen(r), ccen (-2 * g))
b <- lm.fit (lhs, a)$coefficients
k <- 1
s <- matrix (0, p, p)
for (i in 1:p) {
 for (j in 1:i) {
 if (i == j) {
 s[i, i] = b[k]
 } else {
 s[i, j] <- s[j, i] <- b[k]
 }
 k <- k + 1
 }
}
e <- eigen (s)
f <- e$values
if (min(f) < 0) {
 stop ("Negative eigenvalue, cannot proceed")
}
t <- e$vectors %*% diag (sqrt (f))
v <- solve (t, b[-(1:1)])
x <- g %*% t
y <-
 h %*% (e$vectors %*% diag (1 / sqrt (f))) + matrix (v, m, p, byrow = TRUE)
return (list (x = x, y = y))
}

```

```

unfoldals <- function (offdiag) {
 n <- nrow (offdiag)
 m <- ncol (offdiag)
 dd <- offdiag ^ 2
 delta <- matrix (0, n + m, n + m)
 delta[1:n, n + (1:m)] <- dd
 delta <- pmax(delta, t(delta))
 cc <-
 dd - outer (rowSums(dd) / m, colSums (dd) / n, "+") + sum(dd) / (n * m)
 sc <- svd (-cc / 2)
 lb <- diag (sqrt(sc$d))
 xold <- sc$u %*% lb
 yold <- sc$v %*% lb
 zold <- rbind (xold, yold)
 lold <- rowSums (zold ^ 2)
 dold <- outer (lold, lold, "+") - 2 * tcrossprod (zold)

}

teqbounds <- function (offdiag) {
 n <- nrow (offdiag)
 m <- ncol (offdiag)
 a <- matrix (0, n, n)
 b <- matrix (0, m, m)
 for (i in 2:n) {
 for (j in 1:(i - 1)) {
 smin = Inf
 smax = -Inf
 for (k in 1:m) {
 smin = min (smin, offdiag[i, k] + offdiag[j, k])
 smax = max (smax, abs (offdiag[i, k] - offdiag[j, k]))
 }
 a[i, j] <- a[j, i] <- (smin + smax) / 2
 }
 }
 for (i in 2:m) {
 for (j in 1:(i - 1)) {

```

```

smin = Inf
smax = -Inf
for (k in 1:n) {
 smin = min (smin, offdiag[k, i] + offdiag[k, j])
 smax = max (smax, abs (offdiag[k, i] - offdiag[k, j]))
}
b[i, j] <- b[j, i] <- (smin + smax) / 2
}
}
return (list (a = a, b = b))
}

```

### A.1.13 constrained.R

```

pcircsmacof <-
function (delta,
 w = wdef (nrow (delta)),
 p = 2,
 x = smacofInitialR (delta, p),
 itmax = 1000,
 eps = 1e-6,
 verbose = TRUE) {
labels = c("itel", "sold", "snew")
digits = c(4, 10, 10)
widths = c(6, 15, 15)
format = c("d", "f", "f")
n <- nrow (x)
p <- ncol (x)
xold <- x / sqrt (rowSums (x ^ 2))
dold <- as.matrix (dist (xold))
v <- smacofVmatR (w)
e <- max (eigen (v, only.values = TRUE)$values)
vinv <- ginv(v)
itel <- 1
sold <- smacofLossR (dold, w, delta)

```

```

repeat {
 b <- smacofBmatR (dold, w, delta)
 xgut <- smacofGuttmanR(xold, b, vinv)
 xtar <- xold + v %*% (xgut - xold) / e
 xlen <- sqrt (rowSums (xtar ^ 2))
 xrad <- mean (xlen)
 xnew <- (xtar / xlen) * xrad
 dnew <- as.matrix (dist(xnew))
 snew <- smacofLossR (dnew, w, delta)
 if (verbose) {
 values = c(itel, sold, snew)
 iterationWrite (labels, values, digits, width, format)
 }
 if (((sold - snew) < eps) || (itel == itmax)) {
 break
 }
 itel <- itel + 1
 xold <- xnew
 sold <- snew
}
return (list (
 x = xnew,
 d = dnew,
 stress = snew,
 radius = xrad,
 itel = itel
))
}

pellipsmacof <-
 function (delta,
 w = wdef (nrow (delta)),
 p = 2,
 x = smacofInitialR (delta, p),
 itmax = 1000,
 eps = 1e-6,

```

```

 verbose = TRUE) {
labels = c("itel", "sold", "smid", "snew")
digits = c(4, 10, 10, 10)
widths = c(6, 15, 15, 15)
format = c("d", "f", "f", "f")
n <- nrow (x)
p <- ncol (x)
yold <- x / sqrt (rowSums (x ^ 2))
xlbd <- rep (1, p)
xold <- yold %*% diag (xlbd)
dold <- as.matrix (dist (xold))
v <- smacofVmatR (w)
e <- max (eigen (v, only.values = TRUE)$values)
vinv <- ginv(v)
itel <- 1
sold <- smacofLoss(dold, w, delta)
repeat {
 b <- smacofBmatR (dold, w, delta)
 xgut <- smacofGuttmanR(xold, b, inv)
 for (s in 1:p) {
 xlbd[s] <-
 sum (xgut[, s] * (v %*% yold[, s])) / sum (yold[, s] * (v %*% yold[, s]))
 }
 xmid <- yold %*% diag (xlbd)
 dmid <- as.matrix (dist (xmid))
 smid <- sum (w * (delta - dmid) ^ 2) / 2
 mlbd <- max (xlbd ^ 2)
 ytar <-
 yold + v %*% ((xgut %*% diag (1 / xlbd)) - yold) %*% diag (xlbd ^ 2) / (e * m)
 ylen <- sqrt (rowSums (ytar ^ 2))
 ynew <- ytar / ylen
 xnew <- ynew %*% diag (xlbd)
 dnew <- as.matrix (dist(xnew))
 snew <- sum (w * (delta - dnew) ^ 2) / 2
 if (verbose) {
 values = c(itel, sold, smid, snew)
 iterationWrite (labels, values, digits, width, format)
 }
}

```

```

 }
 if (((sold - snew) < eps) || (itel == itmax)) {
 break
 }
 itel <- itel + 1
 xold <- xnew
 yold <- ynew
 sold <- snew
 }
 return (list (
 x = xnew,
 d = dnew,
 stress = snew,
 axes = xlbd,
 itel = itel
))
}
}

dcircsmacof <-
function (delta,
 w = wdef (nrow (delta)),
 p = 2,
 x = smacofInitialR (delta, p),
 pen = 1,
 itmax = 1000,
 eps = 1e-6,
 verbose = TRUE) {
 labels = c("itel", "sold", "smid", "snew")
 digits = c(4, 10, 10, 10)
 widths = c(6, 15, 15, 15)
 format = c("d", "f", "f", "f")
 n <- nrow (x)
 xold <-
 rbind (0, x / sqrt (rowSums(x ^ 2)))
 dold <- as.matrix (dist (xold))
 w <- rbind (pen, cbind (pen, w))
 delta <- rbind (1, cbind (1, delta))
}

```

```
w[1, 1] <- delta [1, 1] <- 0
v <- smacofVmatR (w)
vinv <- ginv(v)
itel <- 1
sold <- smacofLossR(dold, w, delta)
repeat {
 b <- smacofBmatR(dold, w, delta)
 xnew <- smacofGuttmanR(xold, b, vinv)
 dnew <- as.matrix (dist (xnew))
 smid <- smacofLossR(dnew, w, delta)
 a <- sum (dnew[1,]) / n
 delta[1,] <- delta[, 1] <- a
 delta[1, 1] <- 0
 snew <- smacofLossR(dnew, w, delta)
 if (verbose) {
 values = c(itel, sold, smid, snew)
 iterationWrite (labels, values, digits, width, format)
 }
 if (((sold - snew) < eps) || (itel == itmax)) {
 break
 }
 itel <- itel + 1
 xold <- xnew
 sold <- snew
}
return (list (
 x = xnew,
 d = dnew,
 a = a,
 stress = snew,
 itel = itel
))
}

dellipsmacof <-
function (delta,
 w = wdef (nrow (delta)),
```

```

 p = 2,
 x = smacofInitialR (delta, p),
 pen = 1,
 itmax = 1000,
 eps = 1e-6,
 verbose = TRUE) {
 labels = c("itel", "sold", "smid", "snew")
 digits = c(4, 10, 10, 10)
 widths = c(6, 15, 15, 15)
 format = c("d", "f", "f", "f")
 n <- nrow (x)
 set.seed(12345)
 focal <- rnorm(2)
 xold <-
 rbind (focal, -focal, x / sqrt (rowSums(x ^ 2)))
 dold <- as.matrix (dist (xold))
 w <- rbind (pen, pen, cbind (pen, pen, w))
 delta <- rbind (1, 1, cbind (1, 1, delta))
 w[1:2, 1:2] <- delta [1:2, 1:2] <- 0
 v <- smacofVmatR (w)
 vinv <- ginv(v)
 itel <- 1
 sold <- smacofLossR(dold, w, delta)
 repeat {
 b <- smacofBmatR (dold, w, delta)
 xnew <- smacofGuttmanR (xold, b, vinv)
 dnew <- as.matrix (dist (xnew))
 smid <- smacofLossR(dnew, w, delta)
 dsub <- dnew[1:2, 2 + (1:n)]
 asub <- sum (dsub) / (2 * n)
 dsub <- ccen (dsub) + asub
 delta[1:2, 2 + (1:n)] <- dsub
 delta[2 + (1:n), 1:2] <- t(dsub)
 snew <- smacofLossR(dnew, w, delta)
 if (verbose) {
 values = c(itel, sold, smid, snew)
 iterationWrite (labels, values, digits, width, format)
 }
 }
}

```

```

 }
 if (((sold - snew) < eps) || (itel == itmax)) {
 break
 }
 itel <- itel + 1
 xold <- xnew
 sold <- snew
 }
 return (list (
 pen = pen,
 x = xnew,
 d = dnew,
 stress = snew,
 itel = itel
))
}

```

### A.1.14 nominal.R

```

baseplot <- function (x,
 y,
 z,
 wx = TRUE,
 wy = TRUE,
 wz = TRUE) {
 par(pty="s")
 plot(
 x,
 xlim = c(-3, 3),
 ylim = c(-3, 3),
 xlab = "",
 ylab = "",
 type = "n"
)
 if (wx)

```

```
 points(x, col = "RED", cex = 1)
 if (wy)
 points(y, col = "BLUE", cex = 1)
 if (wz)
 points(z, col = "GREEN", cex = 1)
 mx <- apply(x, 2, mean)
 my <- apply(y, 2, mean)
 mz <- apply(z, 2, mean)
 if (wx)
 points(
 matrix(mx, 1, 2),
 col = "RED",
 pch = 5,
 cex = 2,
 lwd = 2
)
 if (wy)
 points(
 matrix(my, 1, 2),
 col = "BLUE",
 pch = 5,
 cex = 2,
 lwd = 2
)
 if (wz)
 points(
 matrix(mz, 1, 2),
 col = "GREEN",
 pch = 5,
 cex = 2,
 lwd = 2
)
 if (wx)
 for (i in 1:10) {
 lines(rbind(x[i,], mx))
 }
 if (wy)
```

```

for (i in 1:5) {
 lines(rbind(y[i,], my))
}
if (wz)
 for (i in 1:5) {
 lines(rbind(z[i,], mz))
 }
}

```

### A.1.15 sstress.R

```

strainAdd <-
 function (delta,
 w = rep (1, length (delta)),
 p = 2,
 itmax = 100,
 eps = 1e-6,
 verbose = TRUE) {
 delta <- as.matrix (delta ^ 2)
 n <- nrow(delta)

}

strainWeight <-
 function (delta,
 w = rep (1, length (delta)),
 p = 2,
 itmax = 100,
 eps = 1e-6,
 verbose = TRUE) {

}

alscal <-
 function (delta,

```

```

p,
x = torgerson (delta, p),
w = wdef (nrow (x)),
itmax = 1000,
eps = 1e-6,
verbose = TRUE,
check = TRUE) {
n <- nrow (x)
delta <- delta ^ 2
d <- as.matrix (dist (x)) ^ 2
sold <- sum (w * (delta - d) ^ 2)
wsum <- rowSums (w)
itel <- 1
snew <- sold
repeat {
 for (k in 1:n) {
 t4 <- wsum[k]
 for (s in 1:p) {
 u <- x[, s] - x[k, s]
 t0 <- snew
 t1 <- t2 <- t3 <- 0
 for (i in 1:n) {
 t1 <- t1 + 4 * w[i, k] * (d[i, k] - delta[i, k]) * u[i]
 t2 <-
 t2 + 2 * w[i, k] * ((d[i, k] - delta[i, k]) + 2 * u[i] ^ 2)
 t3 <- t3 + 4 * w[i, k] * u[i]
 }
 pp <- polynomial(c(t0,-t1, t2,-t3, t4))
 qq <- deriv (pp)
 ss <- solve (qq)
 ss <- Re (ss[which (abs (Im (ss)) < 1e-10)])
 tt <- predict (pp, ss)
 snew <- min (tt)
 root <- ss[which.min (tt)]
 x[k, s] <- x[k, s] + root
 for (i in (1:n)[-k]) {
 d[i, k] <- d[i, k] - 2 * root * u[i] + root ^ 2
 }
 }
 }
}

```

```
 d[k, i] <- d[i, k]
 }
}
}
if (verbose) {
 cat(
 "itel ",
 formatC(itel, width = 6, format = "d"),
 "sold ",
 formatC(
 sold,
 digits = 6,
 width = 15,
 format = "f"
),
 "snew ",
 formatC(
 snew,
 digits = 6,
 width = 15,
 format = "f"
),
 "\n"
)
}
if (((sold - snew) < eps) || (itel == itmax)) {
 break
}
sold <- snew
itel <- itel + 1
}
return (list (
 x = x,
 sstress = snew,
 itel = itel
))
}
```

```
jeffrey <- function(a) {
 h <-
 .C("jeffreyC",
 a = as.double (a),
 minwhere = as.double (0),
 minvalue = as.double (0))
 return (list.remove (h, 1))
}
```

### A.1.16 inverse.R

```
imdsSolver <- function (tau) {
 radius <- sqrt (((tau - 3) ^ 2) / 3)
 center <- c(tau, tau) / 3
 a <- matrix (c(1, .5, .5, 1), 2, 2)
 draw_ellipse (
 center,
 radius,
 a,
 col = "RED",
 lwd = 2,
 xlim = c(0, tau),
 ylim = c(0, tau),
 xlab = "alpha",
 ylab = "beta"
)
 lines (matrix(c(0, tau, tau, 0), 2, 2), col = "BLUE", lwd = 2)
 abline(h = 0)
 abline(v = 0)
}

bs <- function () {
 z <- matrix(c(1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, 1, -1, -1, 1), 4 , 4)
```

```

for (i in 1:3) {
 for (j in (i + 1):4) {
 a[[k]] <- crossprod (z, aijn(i, j, 4) %*% z)
 k <- k + 1
 }
}
return(a)
}

imdsChecker <- function (a) {
 aa <- bs()
 bb <- matrix(0, 4, 4)
 for (i in 1:6) {
 bb <- bb + a[i] * aa[[i]]
 }
 return (bb)
}

inverseMDS <- function (x) {
 n <- nrow (x)
 m <- ncol (x)
 x <- apply (x, 2, function (y)
 y - mean (y))
 nm <- n - (m + 1)
 kk <- cbind (1, x, matrix (rnorm (n * nm), n , nm))
 kperp <- as.matrix (qr.Q (qr (kk))[,-(1:(m + 1))])
 dd <- as.matrix (dist (x))
 k <- 1
 base <- matrix (0, n * (n - 1) / 2, nm * (nm + 1) / 2)
 for (i in 1:nm) {
 for (j in 1:i) {
 oo <- outer (kperp[, i], kperp[, j])
 if (j != i) {
 oo <- oo + t(oo)
 }
 base[, k] <- lower_triangle (dd + (1 - oo))
 k <- k + 1
 }
 }
}

```

```

 print (c(i, j, k))
 }
}
return (base = cbind (lower_triangle (dd), base))
}

inversePlus <- function (base, affine = TRUE) {
 if (affine) {
 hrep <- makeH (
 a1 = d2q (-base),
 b1 = d2q (rep (0, nrow (base))),
 a2 = d2q (rep (1, ncol (base))),
 b2 = d2q (1)
)
 } else {
 hrep <- makeH (a1 = d2q (-base), b1 = d2q (rep (0, nrow (base))))
 }
 vrep <- scdd (hrep)
 hrep <- q2d (hrep)
 vrep <- q2d (vrep$output)
 pr <- tcrossprod (hrep[, -c(1, 2)], vrep[, -c(1, 2)])[-1,]
 return (list (
 base = base,
 hrep = hrep,
 vrep = vrep,
 pr = pr
))
}

twoPoints <- function (x, y, w = 1 - diag (nrow (x))) {
 dx <- lower_triangle (as.matrix (dist (x)))
 dy <- lower_triangle (as.matrix (dist (y)))
 w <- lower_triangle (w)
 gx <- makeG (x)
 gy <- makeG (y)
 hx <- (dx / w) * gx
 hy <- (dy / w) * gy
}
```

```

lxy <- lm.fit (cbind (hx,-hy), dx - dy)
lxx <- lxy$coefficients[1:ncol(hx)]
lyy <- lxy$coefficients[-(1:ncol(hx))]
return (list(
 delta1 = dx - hx %*% lxx,
 delta2 = dy - hy %*% lyy,
 res = sum (abs(lxy$residuals)),
 rank = lxy$rank
))
}

second_partials_stress <-
function (x, delta, w = nonDiag (nrow (x))) {
 n <- nrow (x)
 p <- ncol (x)
 d <- as.matrix (dist (x))
 fac <- (w * delta) / (d + diag (n))
 dd <- d * d
 v <- smacofVmatR (w)
 deri <- direct_sum (repList (v, p))
 xx <- as.vector (x)
 for (i in 1:(n - 1)) {
 for (j in (i + 1):n) {
 aa <- direct_sum (repList (aijn (i, j, n), p))
 ax <- drop (aa %*% xx)
 deri <- deri - fac[i, j] * (aa - outer (ax, ax) / dd[i, j])
 }
 }
 return (deri)
}

second_partials_numerical <-
function (x, delta, w = nonDiag (nrow (x))) {
 stress <- function (x, delta, w) {
 n <- nrow (delta)
 p <- length (x) / n
 d <- as.matrix(dist(matrix (x, n, p)))

```

```

 res <- delta - d
 return (sum (w * res * res) / 2)
 }
 return (hessian (stress, x, delta = delta, w = w))
}

cleanUp <- function (a, eps = 1e-3) {
 nv <- nrow (a)
 ind <- rep (TRUE, nv)
 for (i in 1:(nv - 1)) {
 xx <- a[i,]
 for (j in (i + 1):nv) {
 if (!ind[j])
 next
 yy <- a[j,]
 mm <- max (abs (xx - yy))
 if (mm < eps)
 ind[j] <- FALSE
 }
 }
 return (ind)
}

bruteForce <- function (a, b, eps = 1e-3) {
 n <- nrow (a)
 m <- ncol (a)
 cb <- combn (n, m)
 n1 <- ncol (cb)
 ind <- rep(TRUE, n1)
 ht <- numeric()
 for (i in 1:n1) {
 gg <- a[cb[, i],]
 bg <- b[cb[, i]]
 qg <- qr(gg)
 if (qg$rank < m) {
 ind[i] <- FALSE
 next
 }
 }
}
```

```

}
hh <- solve (qg, bg)
hg <- drop (a %*% hh)
if (min (b - hg) < -eps) {
 ind[i] <- FALSE
 next
}
ht <- c(ht, hh)
}
n2 <- sum (ind)
ht <- matrix (ht, m, n2)
ind <-
.C (
 "cleanup",
 as.double(ht),
 as.integer(n2),
 as.integer(m),
 as.integer(rep(1, n2)),
 as.double (eps)
)[[4]]
n3 <- sum (ind)
return (list (
 x = t(ht)[which(ind == 1),],
 n1 = n1,
 n2 = n2,
 n3 = n3
))
}

bruteForceOne <- function (a, b, p, q, v, eps = 1e-3) {
 n <- nrow (a)
 m <- ncol (a)
 ind <- which ((q - v %*% p) > -eps)
 v <- v[ind,]
 cb <- combn (n, m - 1)
 n1 <- ncol (cb)
 ind <- rep(TRUE, n1)
}

```

```

ht <- numeric()
for (i in 1:n1) {
 gg <- rbind (a[cb[, i],], p)
 bg <- c (b[cb[, i]], q)
 qg <- qr(gg)
 if (qg$rank < m) {
 ind[i] <- FALSE
 next
 }
 hh <- solve (qg, bg)
 hg <- drop (a %*% hh)
 if (min (b - hg) < -eps) {
 ind[i] <- FALSE
 next
 }
 ht <- c(ht, hh)
}
n2 <- sum (ind)
ht <- t (matrix (ht, m, n2))
ht <- rbind (v, ht)
ind <- cleanUp (ht, eps)
print (ind)
n3 <- sum (ind)
return (list (
 x = ht[ind,],
 n1 = n1,
 n2 = n2,
 n3 = n3
))
}

rankTest <- function (x, a, b, eps = 1e-3) {
 h <- drop (a %*% x)
 ind <- which (abs (h - b) < eps)
 r <- qr (a[ind,])$rank
 f <- min (b - h) > -eps
 return (list (rank = r, feasibility = f))
}

```

```

}

makeDC <- function (x) {
 y <- -x
 diag(y) <- -rowSums (y)
 return (y)
}

bmat <- function (delta, w, d) {
 n <- nrow (w)
 dd <- ifelse (d == 0, 0, 1 / d)
 return (makeDC (w * delta * dd))
}

smacof <-
 function (delta,
 w,
 xini,
 eps = 1e-6,
 itmax = 100,
 verbose = TRUE) {
 n <- nrow (xini)
 xold <- xini
 dold <- as.matrix (dist (xold))
 sold <- sum (w * (delta - dold) ^ 2) / 2
 itel <- 1
 v <- ginv (makeDC (w))
 repeat {
 b <- bmat (delta, w, dold)
 xnew <- v %*% b %*% xold
 dnew <- as.matrix (dist (xnew))
 snew <- sum (w * (delta - dnew) ^ 2) / 2
 if (verbose) {
 cat (
 formatC (itel, width = 4, format = "d"),
 formatC (
 sold,

```

```

 digits = 10,
 width = 13,
 format = "f"
),
 formatC (
 snew,
 digits = 10,
 width = 13,
 format = "f"
),
 "\n"
)
}
if ((itel == itmax) || (sold - snew) < eps)
 break ()
itel <- itel + 1
sold <- snew
dold <- dnew
xold <- xnew
}
return (list (
 x = xnew,
 d = dnew,
 s = snew,
 itel = itel
))
}

oneMore <- function (g, u) {
v <- bruteForce (g, u)$x
nv <- nrow (v)
s <- matrix (0, 2, 2)
ev <- rep (0, nv)
for (i in 1:nv) {
 s[1, 1] <- v[i, 1]
 s[2, 2] <- v[i, 2]
 s[1, 2] <- s[2, 1] <- v[i, 3]
}
}
```

```

ee <- eigen (s)
ev[i] <- min (ee$values)
if (ev[i] < 0) {
 yy <- ee$vectors[, 2]
 hh <- c (yy[1] ^ 2, yy[2] ^ 2, 2 * yy[1] * yy [2])
 g <- rbind (g,-hh)
 u <- c (u, 0)
}
}
return (list (
 v = v,
 g = g,
 u = u,
 e = ev
))
}

makeG <- function (x) {
 n <- nrow (x)
 p <- ncol (x)
 m <- n - p - 1
 k <- qr.Q(qr(cbind(1, x, diag (n))))[,-c(1:(p + 1))]
 g <- matrix (0, n * (n - 1) / 2, m * (m + 1) / 2)
 l <- 1
 if (m == 1) {
 g[, 1] <- lower_triangle (outer (k, k))
 }
 else {
 for (i in 1:m) {
 g[, 1] <- lower_triangle (outer(k[, i], k[, i]))
 l <- l + 1
 }
 for (i in 1:(m - 1))
 for (j in (i + 1):m) {
 g[, 1] <-
 lower_triangle (outer(k[, i], k[, j])) + outer(k[, j], k[, i]))
 l <- l + 1
 }
 }
}

```

```

 }
 }
 return (g)
}

iStress <-
 function (x,
 delta,
 w = rep (1, length (delta)),
 only = TRUE) {
 m <- length (delta)
 n <- (1 + sqrt (1 + 8 * m)) / 2
 x <- matrix (x, n, length (x) / n)
 d <- lower_triangle (as.matrix (dist (x)))
 g <- makeG (x)
 h <- (d / w) * makeG (x)
 u <- -colSums(w * (delta - d) * h)
 v <- crossprod (h, w * h)
 s <- solve.QP (
 Dmat = v,
 dvec = u,
 Amat = -t(h),
 bvec = -d
)
 ds <- d - h %*% s$solution
 is <- sum (w * (delta - ds) ^ 2)
 if (only)
 return (is)
 else
 return (list (istress = is, delta = fill_symmetric (ds)))
}

```

### A.1.17 global.R

```

checkUni <- function (w, delta, x) {
 x <- drop (x)
 n <- length (x)
 vinv <- solve (smacofVmat (w) + (1 / n)) - (1 / n)
 return (drop (vinv %*% rowSums (w * delta * sign (outer (
 x, x, "-"
)))))
}

matchMe <- function (x,
 itmax = 100,
 eps = 1e-10,
 verbose = FALSE) {
 m <- length (x)
 y <- sumList (x) / m
 itel <- 1
 fold <- sum (sapply (x, function (z)
 (z - y) ^ 2))
 repeat {
 for (j in 1:m) {
 u <- crossprod (x[[j]], y)
 s <- svd (u)
 r <- tcrossprod (su, sv)
 x[[j]] <- x[[j]] %*% r
 }
 y <- sumList (x) / m
 fnew <- sum (sapply (x, function (z)
 (z - y) ^ 2))
 if (verbose) {
 }
 if (((fold - fnew) < eps) || (itel == itmax))
 break
 itel <- itel + 1
 fold <- fnew
 }
 return (x)
}

```

```

}

sumList <- function (x) {
 m <- length (x)
 y <- x[[1]]
 for (j in 2:m) {
 y <- y + x[[j]]
 }
 return (y)
}

smacofLoss <- function (d, w, delta) {
 return (sum (w * (delta - d) ^ 2) / 4)
}

smacofBmat <- function (d, w, delta) {
 dd <- ifelse (d == 0, 0, 1 / d)
 b <- -dd * w * delta
 diag (b) <- -rowSums (b)
 return(b)
}

smacofVmat <- function (w) {
 v <- -w
 diag(v) <- -rowSums(v)
 return (v)
}

smacofGuttman <- function (x, b, vinv) {
 return (vinv %*% b %*% x)
}

columnCenter <- function (x) {
 return (apply (x, 2, function (z)
 z - mean (z)))
}

```

```

smacofComplement <- function (y, v) {
 return (sum (v * tcrossprod (y)) / 4)
}

smacofPenalty <-
 function (w,
 delta,
 p = 2,
 lbd = 0,
 zold = columnCenter (diag (nrow (delta))),
 itmax = 10000,
 eps = 1e-10,
 verbose = FALSE) {
 itel <- 1
 n <- nrow (zold)
 vmat <- smacofVmat (w)
 vinv <- solve (vmat + (1 / n)) - (1 / n)
 dold <- as.matrix (dist (zold))
 mold <- sum (w * delta * dold) / sum (w * dold * dold)
 zold <- zold * mold
 dold <- dold * mold
 yold <- zold [, (p + 1):n]
 sold <- smacofLoss (dold, w, delta)
 bold <- smacofBmat (dold, w, delta)
 told <- smacofComplement (yold, vmat)
 uold <- sold + lbd * told
 repeat {
 znew <- smacofGuttman (zold, bold, vinv)
 ynew <- znew [, (p + 1):n] / (1 + lbd)
 znew [, (p + 1):n] <- ynew
 xnew <- znew [, 1:p]
 dnew <- as.matrix (dist (znew))
 bnew <- smacofBmat (dnew, w, delta)
 tnew <- smacofComplement (ynew, vmat)
 snew <- smacofLoss (dnew, w, delta)
 unew <- snew + lbd * tnew
 if (verbose) {
 print (c (itel, itmax, eps, lbd, mold, mold - unew))
 if (itel >= itmax) break
 }
 itel <- itel + 1
 }
}

```

```
cat(
 "itel ",
 formatC(itel, width = 4, format = "d"),
 "sold ",
 formatC(
 sold,
 width = 10,
 digits = 6,
 format = "f"
) ,
 "snew ",
 formatC(
 snew,
 width = 10,
 digits = 6,
 format = "f"
) ,
 "told ",
 formatC(
 told,
 width = 10,
 digits = 6,
 format = "f"
) ,
 "tnew ",
 formatC(
 tnew,
 width = 10,
 digits = 6,
 format = "f"
) ,
 "uold ",
 formatC(
 uold,
 width = 10,
 digits = 6,
 format = "f"
```

```

),
"unew",
formatC(
 unew,
 width = 10,
 digits = 6,
 format = "f"
),
"\n"
)
}
if (((uold - unew) < eps) || (itel == itmax)) {
 break
}
itel <- itel + 1
zold <- znew
bold <- bnew
sold <- snew
told <- tnew
uold <- unew
}
zpri <- znew %*% svd(znew)$v
xpri <- zpri[, 1:p]
return (list (
 x = xpri,
 z = zpri,
 b = bnew,
 l = lbd,
 s = snew,
 t = tnew,
 itel = itel
))
}

plotMe2 <- function(hList, labels, s = 1, t = 2) {
 n <- nrow(hList[[1]]$x)
 m <- length (hList)

```

```

par(pty = "s")
hMatch <- matchMe (lapply (hList, function(r)
 r$x))
hMat <- matrix (0, 0, 2)
for (j in 1:m) {
 hMat <- rbind(hMat, hMatch[[j]][, c(s, t)])
}
plot(
 hMat,
 xlab = "dim 1",
 ylab = "dim 2",
 col = c(rep("RED", n * (m - 1)), rep("BLUE", n)),
 cex = c(rep(1, n * (m - 1)), rep(2, n)))
)
for (i in 1:n) {
 hLine <- matrix (0, 0, 2)
 for (j in 1:m) {
 hLine <- rbind (hLine, hMatch[[j]][i, c(s, t)])
 }
 lines(hLine)
}
text(hMatch[[m]], labels, cex = .75)
}

plotMe1 <- function(hList, labels) {
 n <- length (hList[[1]]$x)
 m <- length (hList)
 blow <- function (x) {
 n <- length (x)
 return (matrix (c(1:n, x), n, 2))
 }
 hMat <- matrix (0, 0, 2)
 for (j in 1:m) {
 hMat <- rbind(hMat, blow(hList[[j]]$x))
 }
 plot(
 hMat,

```

```

xlab = "index",
ylab = "x",
col = c(rep("RED", n * (m - 1)), rep("BLUE", n)),
cex = c(rep(1, n * (m - 1)), rep(2, n))
)
for (i in 1:n) {
 hLine <- matrix (0, 0, 2)
 for (j in 1:m) {
 hLine <- rbind (hLine, blow(hList[[j]]$x)[i,])
 lines(hLine)
 }
}
text(blow(hList[[m]]$x), labels, cex = 1.00)
for (i in 1:n) {
 abline(h = hList[[m]]$x[i])
}
}

runPenalty <-
 function (w,
 delta,
 lbd,
 p = 2,
 itmax = 10000,
 eps = 1e-10,
 cut = 1e-6,
 write = TRUE,
 verbose = FALSE) {
 m <- length (lbd)
 hList <- as.list (1:m)
 hList[[1]] <-
 smacofPenalty(
 w,
 delta,

```

```

p,
lbd = lbd[1],
itmax = itmax,
eps = eps,
verbose = verbose
)
for (j in 2:m) {
 hList[[j]] <-
 smacofPenalty(
 w,
 delta,
 p,
 zold = hList[[j - 1]]$z,
 lbd = lbd[j],
 itmax = itmax,
 eps = eps,
 verbose = verbose
)
}
mm <- m
for (i in 1:m) {
 if (write) {
 cat(
 "itel",
 formatC(hList[[i]]$itel, width = 4, format = "d"),
 "lambda",
 formatC(
 hList[[i]]$l,
 width = 10,
 digits = 6,
 format = "f"
),
 "stress",
 formatC(
 hList[[i]]$s,
 width = 8,
 digits = 6,

```

```

 format = "f"
),
 "penalty",
 formatC(
 hList[[i]]$t,
 width = 8,
 digits = 6,
 format = "f"
),
 "\n"
)
}
if (hList[[i]]$t < cut) {
 mm <- i
 break
}
return(hList[1:mm])
}

writeSelected <- function (hList, ind) {
 m <- length (hList)
 n <- length (ind)
 mn <- sort (union (union (1:3, ind), m - (2:0)))
 for (i in mn) {
 if (i > m) {
 next
 }
 cat(
 "itel",
 formatC(hList[[i]]$itel, width = 4, format = "d"),
 "lambda",
 formatC(
 hList[[i]]$l,
 width = 10,
 digits = 6,
 format = "f"

```

```

),
"stress",
formatC(
 hList[[i]]$s,
 width = 8,
 digits = 6,
 format = "f"
),
"penalty",
formatC(
 hList[[i]]$t,
 width = 8,
 digits = 6,
 format = "f"
),
"\n"
)
}
}

rhofun <- function (a) {
 rhomax <- 0
 rhoval <- c(0, 0)
 n <- nrow (a)
 for (i in 1:n) {
 z <- a[i, 1] * xbase + a[i, 2] * ybase
 rho <- sum (delta * dist (z))
 if (rho > rhomax) {
 rhomax <- rho
 rhoval <- a[i,]
 }
 }
 return(list(rhomax = rhomax, rhoval = rhoval))
}

rhomax2Plot <- function (inpoints) {
 par(pty = "s")
}

```

```
s <- (0:500) / 500 * 2 * pi
x <- sin(s)
y <- cos(s)
plot(
 x,
 y,
 type = "l",
 col = "RED",
 xlim = c(-1.25, 1.25),
 ylim = c(-1.25, 1.25),
 lwd = 3
)
points(0, 0, cex = 1.2)
n <- nrow(inpoints)
outpoints <- matrix(0, n, 2)
for (i in 1:n) {
 a <- inpoints[i,]
 if (i == n) {
 b <- inpoints[1,]
 } else {
 b <- inpoints[i + 1,]
 }
 d <- a[1] * b[2] - a[2] * b[1]
 outpoints[i, 1] <- (b[2] - a[2]) / d
 outpoints[i, 2] <- (a[1] - b[1]) / d
}
lines (
 x = inpoints[, 1],
 y = inpoints[, 2],
 col = "BLUE",
 lwd = 2
)
lines (
 x = inpoints[c(1, n), 1],
 y = inpoints[c(1, n), 2],
 col = "BLUE",
 lwd = 2
```

```

)
lines (
 x = outpoints[, 1],
 y = outpoints[, 2],
 col = "BLUE",
 lwd = 2
)
lines (
 x = outpoints[c(1, n), 1],
 y = outpoints[c(1, n), 2],
 col = "BLUE",
 lwd = 2
)
}

rhomax2Comp <-
 function (inpoints,
 itmax = 100,
 eps = 1e-8,
 verbose = TRUE) {
 itel = 1
 repeat {
 n <- nrow(inpoints)
 outpoints <- matrix(0, n, 2)
 for (i in 1:n) {
 a <- inpoints[i,]
 if (i == n) {
 b <- inpoints[1,]
 } else {
 b <- inpoints[i + 1,]
 }
 d <- a[1] * b[2] - a[2] * b[1]
 outpoints[i, 1] <- (b[2] - a[2]) / d
 outpoints[i, 2] <- (a[1] - b[1]) / d
 }
 infun <- rhofun (inpoints)

```

```
oufun <- rhofun (outpoints)
inmax <- infun$rhomax
oumax <- oufun$rhomax
inval <- infun$rhoval
ouval <- oufun$rhoval
for (i in 1:n) {
 outpoints[i,] <- outpoints[i,] / sqrt (sum (outpoints[i,] ^ 2))
}
impoints <- matrix (0, 2 * n, 2)
impoints[seq.int(1, (2 * n) - 1, by = 2),] <- inpoints
impoints[seq.int(2, 2 * n, by = 2),] <- outpoints
if (verbose) {
 cat(
 "itel ",
 formatC(itel, width = 6, format = "d"),
 "vertices ",
 formatC(n, width = 6, format = "d"),
 "innermax ",
 formatC(
 inmax,
 digits = 8,
 width = 15,
 format = "f"
),
 "outermax ",
 formatC(
 oumax,
 digits = 8,
 width = 15,
 format = "f"
),
 "\n"
)
}
if ((itel == itmax) || ((oumax - inmax) < eps)) {
 break
}
```

```

 itel <- itel + 1
 inpoints <- impoints
}
return (list (
 itel = itel,
 vertices = n,
 inmax = inmax,
 oumax = oumax,
 inval = inval,
 ouval = ouval
))
}
```

### A.1.18 mathadd.R

```

lsuw <- function (y,
 w,
 proj,
 xold = rep (1, length (y)),
 v = max (eigen (w)$values) * diag (length (y)),
 itmax = 100,
 eps = 1e-6,
 verbose = FALSE,
 add = 1e-6) {
f <- function (x, y, w) {
 return (sum ((x - y) * w %*% (x - y)))
}
n <- length (y)
labels <- c("itel", "fold", "fnew")
digits <- c(0, 6, 6)
widths <- c(3, 10, 10)
formats <- c("d", "f", "f")
fold <- f (xold, y, w)
itel <- 1
repeat {
```

```

u <- drop (solve (v, w %*% (xold - y)))
xnew <- proj (xold - u, v)
fnew <- f (xnew, y, w)
if (verbose) {
 values <- c(itel, fold, fnew)
 iterWrite (labels, values, digits, widths, formats)
}
if ((itel == itmax) || ((fold - fnew) < eps)) {
 break
}
fold <- fnew
xold <- xnew
itel <- itel + 1
}
return (list (x = xnew, f = fnew, itel = itel))
}

projeq <- function (x, v) {
 s <- sum (v)
 h <- sum (x * rowSums (v))
 return (rep (h / s, length (x)))
}

projplus <- function (x, v) {
 if (!all(v == diag(diag(v)))) {
 stop ("V must be diagonal")
 }
 if (min (diag (v)) < 0) {
 stop ("V must be positive semidefinite")
 }
 return (pmax(x, 0))
}

qpmaj <-
 function (z,
 v = diag (length (z)),
 a = diff (diag (length(z))),
```

```

 b = ifelse (!is.null(a), rep(0, nrow(a)), NULL),
 c = NULL,
 d = ifelse (!is.null(c), rep(0, nrow(c)), NULL),
 h = NULL,
 itmax = 1000,
 eps = 1e-15,
 verbose = FALSE) {
 labs <- c("itel", "fold", "fnew")
 digs <- c(0, 6, 6)
 wids <- c(3, 10, 10)
 fors <- c("d", "f", "f")
 if (is.null(h)) {
 w <- v
 y <- z
 rsum <- 0
 } else {
 w <- crossprod(h, v %*% h)
 y <- drop (solve (w, crossprod (h, v %*% z)))
 rsum <- sum (z * (v %*% (z - h %*% y))) / 2
 }
 winv <- solve (w)
 if (!is.null(a)) {
 nin <- nrow(a)
 dualaa <- a %*% winv %*% t(a)
 feasa <- drop ((a %*% y) - b)
 }
 if (!is.null(c)) {
 neq <- nrow (c)
 dualcc <- c %*% winv %*% t(c)
 feasc <- drop((c %*% y) - d)
 }
 if ((!is.null(a)) && (!is.null(c))) {
 dualac <- a %*% winv %*% t(c)
 feas <- c(feasa, feasc)
 dual <- rbind(cbind(dualaa, dualac), cbind(t(dualac), dualcc))
 }
 if ((!is.null(a)) && (is.null(c))) {

```

```

 feas <- feasa
 dual <- dualaa
}
if ((is.null(a)) && (!is.null(c))) {
 feas <- feasc
 dual <- dualcc
}
vmax <- max(eigen(dual)$values)
itel <- 1
lold <- rep (0, nrow (dual))
fold <-
 -(sum (lold * drop (dual %*% lold)) / 2 +
 sum (lold * feas))
repeat {
 lnew <- lold - (drop(dual %*% lold) + feas) / vmax
 if (!is.null(a)) {
 lnew[1:nin] <- pmax(lnew[1:nin], 0)
 }
 fnew <-
 -(sum (lnew * drop (dual %*% lnew)) / 2 + sum (lnew * feas))
 if (verbose) {
 vals <- c(itel, fold, fnew)
 iterWrite (labs, vals, digs, wids, fors)
 }
 if ((itel == itmax) || ((fnew - fold) < eps)) {
 break
 }
 fold <- fnew
 lold <- lnew
 itel <- itel + 1
}
fdua <- fnew
if ((!is.null(c) && (!is.null(a)))) {
 x <- y + drop (winv %*% drop(cbind(t(a), t(c)) %*% lnew))
 lb <- lnew[1:nin]
 mu <- lnew[-(1:nin)]
}

```

```

if ((is.null(c) && (!is.null(a)))) {
 x <- y + drop (winv %*% drop (t(a) %*% lnew))
 lb <- lnew
}
if ((!is.null(c) && (is.null(a)))) {
 x <- y + drop (winv %*% drop (t(c) %*% lnew))
 mu <- lnew
}
fprs <- sum ((x - y) * drop (w %*% (x - y))) / 2
out <- list(x = x,
 fprimal = fprs,
 fdual = fdua)
if (!is.null(h)) {
 out <-
 list.append(out, ftotal = fprs + rsum, predict = drop (h %*% x))
}
if (!is.null(a)) {
 out <-
 list.append(out, lambda = lb, inequalities = drop (a %*% x - b))
}
if (!is.null(c)) {
 out <- list.append(out, mu = mu, equations = drop (c %*% x - d))
}
return (list.append(out, itel = itel))
}

checkIncreasing <- function (innerknots, lowend, highend) {
 h <- .C(
 "checkIncreasing",
 as.double (innerknots),
 as.double (lowend),
 as.double (highend),
 as.integer (length (innerknots)),
 fail = as.integer (0)
)
 return (h$fail)
}

```

```

}

extendPartition <-
 function (innerknots,
 multiplicities,
 order,
 lowend,
 highend) {
 ninner <- length (innerknots)
 kk <- sum(multiplicities)
 nextended <- kk + 2 * order
 if (max (multiplicities) > order)
 stop ("multiplicities too high")
 if (min (multiplicities) < 1)
 stop ("multiplicities too low")
 if (checkIncreasing (innerknots, lowend, highend))
 stop ("knot sequence not increasing")
 h <-
 .C(
 "extendPartition",
 as.double (innerknots),
 as.integer (multiplicities),
 as.integer (order),
 as.integer (ninner),
 as.double (lowend),
 as.double (highend),
 knots = as.double (rep (0, nextended)))
)
 return (h)
}

bisect <-
 function (x,
 knots,
 lowindex = 1,
 highindex = length (knots)) {
 h <- .C(

```

```

 "bisect",
 as.double (x),
 as.double (knots),
 as.integer (lowindex),
 as.integer (highindex),
 index = as.integer (0)
)
return (h$index)
}

bsplines <- function (x, knots, order) {
 if ((x > knots[length(knots)]) || (x < knots[1]))
 stop ("argument out of range")
 h <- .C(
 "bsplines",
 as.double (x),
 as.double (knots),
 as.integer (order),
 as.integer (length (knots)),
 index = as.integer (0),
 q = as.double (rep(0, order))
)
 return (list (q = h$q, index = h$ind))
}

bsplineBasis <- function (x, knots, order) {
 n <- length (x)
 k <- length (knots)
 m <- k - order
 result <- rep (0, n * m)
 h <- .C(
 "bsplineBasis",
 as.double (x),
 as.double (knots),
 as.integer (order),
 as.integer (k),
)
}

```

```

 as.integer (n),
 result = as.double (result)
)
return (matrix (h$result, n, m))
}

isplineBasis <- function (x, knots, order) {
 n <- length (x)
 k <- length (knots)
 m <- k - order
 result <- rep (0, n * m)
 h <- .C(
 "isplineBasis",
 as.double (x),
 as.double (knots),
 as.integer (order),
 as.integer (k),
 as.integer (n),
 result = as.double (result)
)
return (matrix (h$result, n, m))
}

```

## A.2 C code

### A.2.1 deboor.c

```

#include <math.h>
#include <stdbool.h>
#include <stdlib.h>

inline int VINDEX(const int);
inline int MINDEX(const int, const int, const int);
inline int IMIN(const int, const int);

```

```

inline int IMIN(const int, const int);

void checkIncreasing(const double *, const double *, const double *,
 const int *, bool *);
void extendPartition(const double *, const int *, const int *, const int *,
 const double *, const double *, double *);
void bisect(const double *, const double *, const int *, const int *, int *);
void bsplines(const double *, const double *, const int *, const int *, int *,
 double *);
void bsplineBasis(const double *, const double *, const int *, const int *,
 const int *, double *);
void isplineBasis(const double *, const double *, const int *, const int *,
 const int *, double *);
void bsplineSparse(const double *, const double *, const int *, const int *,
 const int *, int *, double *);
void isplineSparse(const double *, const double *, const int *, const int *,
 const int *, int *, double *);

inline int VINDEX(const int i) { return i - 1; }

inline int MINDEX(const int i, const int j, const int n) {
 return (i - 1) + (j - 1) * n;
}

inline int IMIN(const int a, const int b) {
 if (a > b) return b;
 return a;
}

inline int IMAX(const int a, const int b) {
 if (a < b) return b;
 return a;
}

void checkIncreasing(const double *innerknots, const double *lowend,
 const double *highend, const int *ninner, bool *fail) {
 *fail = false;
}

```

```

if (*lowend >= innerknots[VINDEX(1)]) {
 *fail = true;
 return;
}
if (*highend <= innerknots[VINDEX(*ninner)]) {
 *fail = true;
 return;
}
for (int i = 1; i < *ninner; i++) {
 if (innerknots[i] <= innerknots[i - 1]) {
 *fail = true;
 return;
 }
}
}

void extendPartition(const double *innerknots, const int *multiplicities,
 const int *order, const int *ninner, const double *lowend,
 const double *highend, double *extended) {
 int k = 1;
 for (int i = 1; i <= *order; i++) {
 extended[VINDEX(k)] = *lowend;
 k++;
 }
 for (int j = 1; j <= *ninner; j++)
 for (int i = 1; i <= multiplicities[VINDEX(j)]; i++) {
 extended[VINDEX(k)] = innerknots[VINDEX(j)];
 k++;
 }
 for (int i = 1; i <= *order; i++) {
 extended[VINDEX(k)] = *highend;
 k++;
 }
}

void bisect(const double *x, const double *knots, const int *lowindex,
 const int *highindex, int *index) {

```

```

int l = *lowindex, u = *highindex, mid = 0;
while ((u - l) > 1) {
 mid = (int)floor((u + l) / 2);
 if (*x < knots[VINDEX(mid)])
 u = mid;
 else
 l = mid;
}
*index = l;
return;
}

void bsplines(const double *x, const double *knots, const int *order,
 const int *nknots, int *index, double *q) {
 int lowindex = 1, highindex = *nknots, m = *order, j, jp1;
 double drr, dll, saved, term;
 double *dr = (double *)calloc((size_t)m, sizeof(double));
 double *dl = (double *)calloc((size_t)m, sizeof(double));
 (void)bisect(x, knots, &lowindex, &highindex, index);
 int l = *index;
 for (j = 1; j <= m; j++) {
 q[VINDEX(j)] = 0.0;
 }
 if (*x == knots[VINDEX(*nknots)]) {
 q[VINDEX(m)] = 1.0;
 return;
 }
 q[VINDEX(1)] = 1.0;
 j = 1;
 if (j >= m) return;
 while (j < m) {
 dr[VINDEX(j)] = knots[VINDEX(l + j)] - *x;
 dl[VINDEX(j)] = *x - knots[VINDEX(l + 1 - j)];
 jp1 = j + 1;
 saved = 0.0;
 for (int r = 1; r <= j; r++) {
 drr = dr[VINDEX(r)];

```

```

 dll = dl[VINDEX(jp1 - r)];
 term = q[VINDEX(r)] / (drr + dll);
 q[VINDEX(r)] = saved + drr * term;
 saved = dll * term;
 }
 q[VINDEX(jp1)] = saved;
 j = jp1;
}
free(dr);
free(dl);
return;
}

void bsplineBasis(const double *x, const double *knots, const int *order,
 const int *nknots, const int *nvalues, double *result) {
int m = *order, l = 0;
double *q = (double *)calloc((size_t)m + 1, sizeof(double));
for (int i = 1; i <= *nvalues; i++) {
 (void)bsplines(x + VINDEX(i), knots, order, nknots, &l, q);
 for (int j = 1; j <= m; j++) {
 int r = IMIN(l - m + j, *nknots - m);
 result[MINDEX(i, r, *nvalues)] = q[VINDEX(j)];
 }
}
free(q);
return;
}

void isplineBasis(const double *x, const double *knots, const int *order,
 const int *nknots, const int *nvalues, double *result) {
int m = *nknots - *order, n = *nvalues;
(void)bsplineBasis(x, knots, order, nknots, nvalues, result);
for (int i = 1; i <= n; i++) {
 for (int j = m - 1; j >= 1; j--) {
 result[MINDEX(i, j, n)] += result[MINDEX(i, j + 1, n)];
 }
}
}

```

```

 return;
 }

void bsplineSparse(const double *x, const double *knots, const int *order,
 const int *nknots, const int *nvalues, int *columns,
 double *result) {
 int m = *order, l = 0;
 double *q = (double *)calloc((size_t)m + 1, sizeof(double));
 for (int i = 1; i <= *nvalues; i++) {
 (void)bsplines(x + VINDEX(i), knots, order, nknots, &l, q);
 columns[VINDEX(i)] = l;
 for (int j = 1; j <= m; j++) {
 result[MINDEX(i, j, *nvalues)] = q[VINDEX(j)];
 }
 }
 free(q);
 return;
}

void isplineSparse(const double *x, const double *knots, const int *order,
 const int *nknots, const int *nvalues, int *columns,
 double *result) {
 int m = *nknots - *order, n = *nvalues;
 for (int i = 1; i <= *nvalues; i++) {
 (void)bsplineSparse(x + VINDEX(i), knots, order, nknots, nvalues,
 columns, result);
 for (int j = m - 1; j >= 1; j--) {
 result[MINDEX(i, j, n)] += result[MINDEX(i, j + 1, n)];
 }
 }
 return;
}

```

### A.2.2 cleanup.c

```
#include <math.h>

#define MAX(a,b) (((a)>(b))?(a):(b))

void cleanup (double* a, int* n, int *m, int* ind, double *eps) {
 int i, j, l, ii, jj, nn = *n, mm = *m;
 double s;
 for (i = 0; i < (nn - 1); i++) {
 ii = i * mm;
 for (j = (i + 1); j < nn; j++) {
 s = 0.0;
 jj = j * mm;
 if (ind[j] == 0) continue;
 for (l = 0; l < mm; l++) {
 s = MAX (s, fabs (*(a+ii+l) - *(a + jj + l)));
 }
 if (s < *eps) ind[j] = 0;
 }
 }
}
```

### A.2.3 jacobi.c

```
#include "jacobi.h"

void jacobiC(const int *nn, double *a, double *evec, double *oldi, double *oldj,
 int *itmax, double *eps) {
 int n = *nn, itel = 1;
 double d = 0.0, s = 0.0, t = 0.0, u = 0.0, v = 0.0, p = 0.0, q = 0.0,
 r = 0.0;
 double fold = 0.0, fnew = 0.0;
 for (int i = 1; i <= n; i++) {
 for (int j = 1; j <= n; j++) {
```

```

 evec[MINDEX(i, j, n)] = (i == j) ? 1.0 : 0.0;
 }
}
for (int i = 1; i <= n; i++) {
 fold += SQUARE(a[TINDEX(i, i, n)]);
}
while (true) {
 for (int j = 1; j <= n - 1; j++) {
 for (int i = j + 1; i <= n; i++) {
 p = a[TINDEX(i, j, n)];
 q = a[TINDEX(i, i, n)];
 r = a[TINDEX(j, j, n)];
 if (fabs(p) < 1e-10) continue;
 d = (q - r) / 2.0;
 s = (p < 0) ? -1.0 : 1.0;
 t = -d / sqrt(SQUARE(d) + SQUARE(p));
 u = sqrt((1 + t) / 2);
 v = s * sqrt((1 - t) / 2);
 for (int k = 1; k <= n; k++) {
 int ik = IMIN(i, k);
 int ki = IMAX(i, k);
 int jk = IMIN(j, k);
 int kj = IMAX(j, k);
 oldi[VINDEX(k)] = a[TINDEX(ki, ik, n)];
 oldj[VINDEX(k)] = a[TINDEX(kj, jk, n)];
 }
 for (int k = 1; k <= n; k++) {
 int ik = IMIN(i, k);
 int ki = IMAX(i, k);
 int jk = IMIN(j, k);
 int kj = IMAX(j, k);
 a[TINDEX(ki, ik, n)] =
 u * oldi[VINDEX(k)] - v * oldj[VINDEX(k)];
 a[TINDEX(kj, jk, n)] =
 v * oldi[VINDEX(k)] + u * oldj[VINDEX(k)];
 }
 }
 }
}

```

```

 oldi[VINDEX(k)] = evec[MINDEX(k, i, n)];
 oldj[VINDEX(k)] = evec[MINDEX(k, j, n)];
 evec[MINDEX(k, i, n)] =
 u * oldi[VINDEX(k)] - v * oldj[VINDEX(k)];
 evec[MINDEX(k, j, n)] =
 v * oldi[VINDEX(k)] + u * oldj[VINDEX(k)];
 }
 a[TINDEX(i, i, n)] =
 SQUARE(u) * q + SQUARE(v) * r - 2 * u * v * p;
 a[TINDEX(j, j, n)] =
 SQUARE(v) * q + SQUARE(u) * r + 2 * u * v * p;
 a[TINDEX(i, j, n)] =
 u * v * (q - r) + (SQUARE(u) - SQUARE(v)) * p;
}
fnew = 0.0;
for (int i = 1; i <= n; i++) {
 fnew += SQUARE(a[TINDEX(i, i, n)]);
}
if (((fnew - fold) < *eps) || (itel == *itmax)) break;
fold = fnew;
itel++;
}
return;
}

void primat(const int *n, const int *m, const int *w, const int *p,
 const double *x) {
 for (int i = 1; i <= *n; i++) {
 for (int j = 1; j <= *m; j++) {
 printf(" %.*f ", *w, *p, x[MINDEX(i, j, *n)]);
 }
 printf("\n");
 }
 printf("\n\n");
 return;
}

```

```
void pritru(const int *n, const int *w, const int *p, const double *x) {
 for (int i = 1; i <= *n; i++) {
 for (int j = 1; j <= i; j++) {
 printf(" %.*f ", *w, *p, x[TINDEX(i, j, *n)]);
 }
 printf("\n");
 }
 printf("\n\n");
 return;
}

void trimat(const int *n, const double *x, double *y) {
 int nn = *n;
 for (int i = 1; i <= nn; i++) {
 for (int j = 1; j <= nn; j++) {
 y[MINDEX(i, j, nn)] =
 (i >= j) ? x[TINDEX(i, j, nn)] : x[TINDEX(j, i, nn)];
 }
 }
 return;
}

void mattri(const int *n, const double *x, double *y) {
 int nn = *n;
 for (int j = 1; j <= nn; j++) {
 for (int i = j; i <= nn; i++) {
 y[TINDEX(i, j, nn)] = x[MINDEX(i, j, nn)];
 }
 }
 return;
}
```

#### A.2.4 jbkTies.c

```
#include <math.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>

#define DEBUG false

struct block {
 double value;
 double weight;
 int size;
 int previous;
 int next;
};

struct quadruple {
 double value;
 double result;
 double weight;
 int index;
};

struct triple {
 double value;
 double weight;
 int index;
};

struct pair {
 int value;
 int index;
};

int myCompDouble (const void *, const void *);
```

```

int myCompInteger (const void *, const void *);
void mySortDouble (double *, double *, double *, int *, const int *);
void mySortInteger (int *, int *, const int *);
void mySortInBlock (double *, double *, int *, int *);
void tieBlock (double *, int *, const int *, int *);
void makeBlocks (double *, double *, double *, int *, const int *, const int *);
void sortBlocks (double *, double *, int *, const int *, const int *, const int *);
void jbkPava (double *, double *, const int *);
void primary (double *, double *, int *, int *, const int *, const int *);
void secondary (double *, double *, int *, const int *, const int *);
void tertiary (double *, double *, int *, const int *, const int *);
void monreg (double *, double *, double *, const int *, const int *);

int myCompDouble (const void *px, const void *py) {
 double x = ((struct quadruple *)px)->value;
 double y = ((struct quadruple *)py)->value;
 return (int)copysign(1.0, x - y);
}

int myCompInteger (const void *px, const void *py) {
 int x = ((struct pair *)px)->value;
 int y = ((struct pair *)py)->value;
 return (int)copysign(1.0, x - y);
}

void mySortInBlock (double *x, double *w, int *xind, int *n) {
 int nn = *n;
 struct triple *xi =
 (struct triple *) calloc((size_t) nn, (size_t) sizeof(struct triple));
 for (int i = 0; i < nn; i++) {
 xi[i].value = x[i];
 xi[i].weight = w[i];
 xi[i].index = xind[i];
 }
 (void) qsort(xi, (size_t)nn, (size_t)sizeof(struct triple), myCompDouble);
 for (int i = 0; i < nn; i++) {
 x[i] = xi[i].value;
 }
}

```

```
w[i] = xi[i].weight;
xind[i] = xi[i].index;
}
free(xi);
return;
}

void mySortDouble (double *x, double *y, double *w, int *xind, const int *n) {
 int nn = *n;
 struct quadruple *xi =
 (struct quadruple *) calloc((size_t) nn, (size_t) sizeof(struct quadruple));
 for (int i = 0; i < nn; i++) {
 xi[i].value = x[i];
 xi[i].result = y[i];
 xi[i].weight = w[i];
 xi[i].index = i + 1;
 }
 (void) qsort(xi, (size_t)nn, (size_t)sizeof(struct quadruple), myCompDouble);
 for (int i = 0; i < nn; i++) {
 x[i] = xi[i].value;
 y[i] = xi[i].result;
 w[i] = xi[i].weight;
 xind[i] = xi[i].index;
 }
 free(xi);
 return;
}

void mySortInteger (int *x, int *k, const int *n) {
 int nn = *n;
 struct pair *xi =
 (struct pair *) calloc((size_t) nn, (size_t) sizeof(struct pair));
 for (int i = 0; i < nn; i++) {
 xi[i].value = x[i];
 xi[i].index = i + 1;
 }
```

```
(void) qsort(xi, (size_t)nn, (size_t)sizeof(struct pair), myCompInteger);
for (int i = 0; i < nn; i++) {
 x[i] = xi[i].value;
 k[i] = xi[i].index;
}
free(xi);
return;
}

void tieBlock (double *x, int *iblks, const int *n, int *nblk) {
 iblks[0] = 1;
 for (int i = 1; i < *n; i++) {
 if (x[i - 1] == x[i]) {
 iblks[i] = iblks[i - 1];
 } else {
 iblks[i] = iblks[i - 1] + 1;
 }
 }
 *nblk = iblks[*n - 1];
 return;
}

void makeBlocks (double *x, double *w, double *xblks, double *wblks, int *iblk)
for (int i = 0; i < *nblk; i++) {
 xblks[i] = 0.0;
 wblks[i] = 0.0;
}
for (int i = 0; i < *n; i++) {
 xblks [iblks [i] - 1] += w[i] * x[i];
 wblks [iblks [i] - 1] += w[i];
}
for (int i = 0; i < *nblk; i++) {
 xblks[i] = xblks[i] / wblks[i];
}
return;
```

```

void sortBlocks (double *y, double *w, int *xind, const int *iblks, const int *n, const int *blk)
{
 int *nblk = (int *) calloc((size_t) * nblk, sizeof(int));
 for (int i = 0; i < *n; i++) {
 nblk[iblks[i] - 1]++;
 }
 int k = 0;
 for (int i = 0; i < *nblk; i++) {
 int nn = nblk[i];
 (void) mySortInBlock (y + k, w + k, xind + k, &nn);
 k += nn;
 }
 free (nblk);
 return;
}

void jbkPava (double *x, double *w, const int *n) {
 struct block *blocks = calloc ((size_t) * n, sizeof(struct block));
 for (int i = 0; i < *n; i++) {
 blocks[i].value = x[i];
 blocks[i].weight = w[i];
 blocks[i].size = 1;
 blocks[i].previous = i - 1; // index first element previous block
 blocks[i].next = i + 1; // index first element next block
 }
 int active = 0;
 do {
 bool upsatisfied = false;
 int next = blocks[active].next;
 if (next == *n) upsatisfied = true;
 else if (blocks[next].value > blocks[active].value) upsatisfied = true;
 if (!upsatisfied) {
 double ww = blocks[active].weight + blocks[next].weight;
 int nextnext = blocks[next].next;
 blocks[active].value = (blocks[active].weight * blocks[active].value + blocks[next].value) / ww;
 blocks[active].weight = ww;
 blocks[active].size += blocks[next].size;
 blocks[active].next = nextnext;
 }
 } while (!upsatisfied);
}

```

```
 if (nextnext < *n)
 blocks[nextnext].previous = active;
 blocks[next].size = 0;
 }

 bool downsatisfied = false;
 int previous = blocks[active].previous;
 if (previous == -1) downsatisfied = true;
 else if (blocks[previous].value < blocks[active].value) downsatisfied
 if (!downsatisfied) {
 double ww = blocks[active].weight + blocks[previous].weight;
 int previousprevious = blocks[previous].previous;
 blocks[active].value = (blocks[active].weight * blocks[active].val
 blocks[active].weight = ww;
 blocks[active].size += blocks[previous].size;
 blocks[active].previous = previousprevious;
 if (previousprevious > -1)
 blocks[previousprevious].next = active;
 blocks[previous].size = 0;
 }
 if ((blocks[active].next == *n) && downsatisfied) break;
 if (upsatisfied && downsatisfied) active = next;
} while (true);
int k = 0;
for (int i = 0; i < *n; i++) {
 int blksize = blocks[i].size;
 if (blksize > 0.0) {
 for (int j = 0; j < blksize; j++) {
 x[k] = blocks[i].value;
 k++;
 }
 }
}
free (blocks);
}
```

### A.2.5 matrix.c

```
#include "smacof.h"

void gsC(double *x, double *r, int *n, int *m, int *rank, int *pivot,
 double *eps) {
 int i, j, ip, nn = *n, mm = *m, rk = *m, jwork = 1;
 double s = 0.0, p;
 for (j = 1; j <= mm; j++) {
 pivot[j - 1] = j;
 }
 while (jwork <= rk) {
 for (j = 1; j < jwork; j++) {
 s = 0.0;
 for (i = 1; i <= nn; i++) {
 s += x[MINDEX(i, jwork, nn)] * x[MINDEX(i, j, nn)];
 }
 r[MINDEX(j, jwork, mm)] = s;
 for (i = 1; i <= nn; i++) {
 x[MINDEX(i, jwork, nn)] -= s * x[MINDEX(i, j, nn)];
 }
 }
 s = 0.0;
 for (i = 1; i <= nn; i++) {
 s += x[MINDEX(i, jwork, nn)] * x[MINDEX(i, jwork, nn)];
 }
 if (s > *eps) {
 s = sqrt(s);
 r[MINDEX(jwork, jwork, mm)] = s;
 for (i = 1; i <= nn; i++) {
 x[MINDEX(i, jwork, nn)] /= s;
 }
 jwork += 1;
 }
 if (s <= *eps) {
 ip = pivot[rk - 1];
 pivot[rk - 1] = pivot[jwork - 1];
 }
 }
}
```

```

pivot[jwork - 1] = ip;
for (i = 1; i <= nn; i++) {
 p = x[MINDEX(i, rk, nn)];
 x[MINDEX(i, rk, nn)] = x[MINDEX(i, jwork, nn)];
 x[MINDEX(i, jwork, nn)] = p;
}
for (j = 1; j <= mm; j++) {
 p = r[MINDEX(j, rk, mm)];
 r[MINDEX(j, rk, mm)] = r[MINDEX(j, jwork, mm)];
 r[MINDEX(j, jwork, mm)] = p;
}
rk -= 1;
}
*rank = rk;
}

```

### A.2.6 jeffrey.c

```

// This implements the formulas in
//
// D.J. Jeffrey
// Formulae, Algorithms, and Quartic Extrema
// Mathematics Magazine, 1997, 70(5), 341-348
//
// to find the minimum of a quartic polynomial.

#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define SQ(x) ((x) * (x))
#define CU(x) ((x) * (x) * (x))
#define QU(x) ((x) * (x) * (x) * (x))
#define MIN(a,b)

```

```

void jeffreyC (double *a, double *minwhere, double *minvalue) {
 if (a[4] <= 0.0) {
 printf("Quartic term must be positive. Exiting...\n");
 exit(-1);
 }
 double b0 = 0.0, b1 = 0.0, b2 = 0.0;
 b2 += a[2] / (3.0 * a[4]);
 b2 -= (SQ(a[3])) / (8.0 * SQ(a[4]));
 b1 += a[1] / (2.0 * a[4]);
 b1 += CU(a[3]) / (16.0 * CU(a[4]));
 b1 -= (a[2] * a[3]) / (4.0 * SQ(a[4]));
 b0 += (a[2] * SQ(a[3])) / (16.0 * SQ(a[4]));
 b0 -= (a[1] * a[3]) / (4.0 * a[4]);
 b0 -= (3.0 * QU(a[3])) / (256.0 * CU(a[4]));
 if (fabs (b1) > 1e-15) {
 double s, t, k, infp;
 s = SQ(b1) + CU(b2) + sqrt(QU(b1) + 2 * SQ(b1) * CU(b2));
 t = pow(s, (double)1 / 3);
 k = t + (SQ(b2) / t) + b2;
 infp = -0.75 * (k - b2) * (k - 3.0 * b2);
 *minwhere = -b1 / k - 0.25 * a[3] / a[4];
 *minvalue = infp * a[4] + a[0] + b0;
 } else {
 double infp;
 infp = -2.25 * SQ (fmin (0.0, b2));
 *minwhere = sqrt (- fmin(0.0, 1.5 * b2));
 *minvalue = infp * a[4] + a[0] + b0;
 }
 return;
}

```

### A.2.7 mySort.c

```
#include <math.h>
```

```
#include <stdlib.h>
#include <stdio.h>

int myComp(const void *, const void *);
void myBlockSort(const double *, const int, couple *, block *);

typedef struct couple {
 double value;
 int index;
} couple;

typedef struct block {
 double value;
 int size;
 int rank;
} block;

int myComp(const void *px, const void *py) {
 double x = ((struct couple *)px)->value;
 double y = ((struct couple *)py)->value;
 return (int)copysign(1.0, x - y);
}

void myBlockSort(const double *x, const int n, couple *xi, block *yi) {
 for (int i = 0; i < n; i++) {
 xi[i].value = x[i];
 xi[i].index = i;
 }
 (void)qsort(xi, (size_t)n, (size_t)sizeof(couple), myComp);
 int k = 0, r = 0, *m = 0;
 for (int i = 0; i < n - 1; i++) {
 block[r].value = xi[i].value;
 block[r].rank = r;
 if (xi[i].value == xi[i+1].value) {
 block[r].size += 1;
 } else {
 r += 1;
 }
 }
}
```

```

 }
 }

 for (int i = 0; i < n; i++) {
 printf("value %4.2f size %2d, rank %2d\n",
 block[i].value, block[i].size, block[i].rank);
 }
 return;
}

double x[10] = {3.0, 1.0, 1.0, 5.0, 1.0, 5.0, 1.0, 2.0, 5.0, 2.0};
int n = 10;

int main () {
 couple *xi =
 (struct couple *)calloc((size_t)n, (size_t)sizeof(couple));
 block *yi =
 (struct couple *)calloc((size_t)n, (size_t)sizeof(block));

 (void) myBlockSort (x, n, xi, yi);
 free(yi)
}

```

### A.2.8 nextPC.c

```

void
swap(int *x, int i, int j)
{
 int temp;
 temp = x[i];
 x[i] = x[j];
 x[j] = temp;
}

void

```

```
nextPermutation (int *x, int *nn)
{
 int i, j, n = *nn;
 i = n - 1;
 while (x[i - 1] >= x[i])
 i--;
 if (i == 0)
 return;
 j = n;
 while (x[j - 1] <= x[i - 1])
 j--;
 swap(x, i - 1, j - 1);
 j = n;
 i++;
 while (i < j) {
 swap(x, i - 1, j - 1);
 j--;
 i++;
 }
}

void nextCombination (int* n, int* m, int* next) {
 int i, j, mm = *m - 1, nn = *n;
 for (i = mm; i >= 0; i--) {
 if (next[i] != nn - mm + i) {
 next[i]++;
 if (i < mm) {
 for (j = i + 1; j <= mm; j++)
 next[j] = next[j - 1] + 1;
 }
 return;
 }
 }
}
```

# Appendix B

## Data

### B.1 Small

```
1 2 3
2 1
3 2 4
4 5 2 1
```

### B.2 De Gruijter

```
KVP PvdA VVD ARP CHU CPN
KVP 0.00000000 0.10473553 0.09803841 0.08557432 0.08929495 0.14026748
PvdA 0.10473553 0.00000000 0.12501293 0.10492156 0.11571137 0.09524794
VVD 0.09803841 0.12501293 0.00000000 0.10157300 0.09245748 0.15124332
ARP 0.08557432 0.10492156 0.10157300 0.00000000 0.05952996 0.14584841
CHU 0.08929495 0.11571137 0.09245748 0.05952996 0.00000000 0.14510429
CPN 0.14026748 0.09524794 0.15124332 0.14584841 0.14510429 0.00000000
PSP 0.12519896 0.08538829 0.14045351 0.12519896 0.13171005 0.07590070
BP 0.13357036 0.13431448 0.12836149 0.13543067 0.12947767 0.11794374
D66 0.11478121 0.10175903 0.08687654 0.11403709 0.11236281 0.13803510
PSP BP D66
KVP 0.12519896 0.1335704 0.11478121
PvdA 0.08538829 0.1343145 0.10175903
```

```
VVD 0.14045351 0.1283615 0.08687654
ARP 0.12519896 0.1354307 0.11403709
CHU 0.13171005 0.1294777 0.11236281
CPN 0.07590070 0.1179437 0.13803510
PSP 0.00000000 0.1279894 0.11831580
BP 0.12798942 0.0000000 0.13691892
D66 0.11831580 0.1369189 0.00000000
```

### B.3 Ekman

### B.4 Vegetables

```
veg <- abs (qnorm (matrix (c(.500,.818,.770,.811,.878,.892,.899,.892,.926,
.182,.500,.601,.723,.743,.736,.811,.845,.858,
.230,.399,.500,.561,.736,.676,.845,.797,.818,
.189,.277,.439,.500,.561,.588,.676,.601,.730,
.122,.257,.264,.439,.500,.493,.574,.709,.764,
.108,.264,.324,.412,.507,.500,.628,.682,.628,
.101,.189,.155,.324,.426,.372,.500,.527,.642,
.108,.155,.203,.399,.291,.318,.473,.500,.628,
.074,.142,.182,.270,.236,.372,.358,.372,.500), 9, 9)))
```

# Appendix C

## Unlicense

This book is free and unencumbered text and software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this book, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author of this book dedicates any and all copyright interest in the software to the public domain. I make this dedication for the benefit of the public at large and to the detriment of my heirs and successors. I intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this book under copyright law.

THE BOOK IS PROVIDED “AS IS,” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE BOOK OR THE USE OR OTHER DEALINGS IN THE BOOK.

For more information, please refer to <http://unlicense.org/>



# References

- Bacak, M., and J. M. Borwein. 2011. "On Difference Convexity of Locally Lipschitz Functions." *Optimization* 60 (8-9): 961–78.
- Bailey, R. A., and J. C. Gower. 1990. "Approximating a Symmetric Matrix." *Psychometrika* 55 (4): 665–75.
- Bauschke, H. H., M. N. Bui, and X. Wang. 2018. "Projecting onto the Intersection of a Cone and a Sphere." *SIAM Journal on Optimization* 28: 2158–88.
- Bavaud, F. 2011. "On the Schoenberg Transformations in Data Analysis: Theory and Illustrations." *Journal of Classification* 28: 297–314.
- Berge, C. 1963. *Topological Spaces*. Oliver & Boyd.
- Best, M. J. 2017. *Quadratic Programming with Computer Programs*. Advances in Applied Mathematics. CRC Press.
- Blumenthal, L. M. 1953. *Theory and Applications of Distance Geometry*. Oxford University Press.
- Borg, I., and P. J. F. Groenen. 2005. *Modern Multidimensional Scaling: Theory and Applications*. Second Edition. Springer.
- Borg, I., and J. C. Lingoes. 1980. "A Model and Algorithm for Multidimensional Scaling with External Constraints on the Distances." *Psychometrika* 45 (1): 25–38.
- Browne, M. W. 1987. "The Young-Householder Algorithm and the Least Squares Multidimensional Scaling of Squared Distances." *Journal of Classification* 4: 175–90.
- Busing, F. M. T. A. 2021. "Monotone Regression: A Simple and Fast O(n) PAVA Implementation." *Journal of Statistical Software* Submitted.
- Cailliez, F. 1983. "The Analytical Solution to the Additive Constant Problem." *Psychometrika* 48 (2): 305–8.
- Carroll, J. D., and J. J. Chang. 1970. "Analysis of Individual Differences in Multidimensional scaling via an N-way generalization of "Eckart-Young"

- Decomposition.” *Psychometrika* 35: 283–319.
- Ciomek, K. 2017. *hasseDiagram: Drawing Hasse Diagram*. %7Bhttps://CRAN.R-project.org/package=hasseDiagram%7D.
- Coombs, C. H. 1964. *A Theory of Data*. Wiley.
- Cooper, L. G. 1972. “A New Solution to the Additive Constant Problem in Metric Multidimensional Scaling.” *Psychometrika* 37 (3): 311–22.
- Cox, D. R., and L. Brandwood. 1959. “On a Discriminatory Problem Connected with the Works of Plato.” *Journal of the Royal Statistical Society, Series B* 21: 195–200.
- Cox, M. G. 1972. “The Numerical Evaluation of B-splines.” *Journal of the Institute of Mathematics and Its Applications* 10: 134–49.
- Cox, T. F., and M. A. A. Cox. 1991. “Multidimensional Scaling on a Sphere.” *Communications in Statistics* 20: 2943–53.
- . 2001. *Miltidimensional Scaling*. Second Edition. Monographs on Statistics and Applied Probability 88. Chapman & Hall.
- Critchley, F. 1988. “On Certain Linear Mappings Between Inner Product and Squared Distance Matrices.” *Linear Algebra and Its Applications* 105: 91–107.
- Danskin, J. M. 1966. “The Theory of Max-Min, with Applications.” *SIAM Journal on Applied Mathematics* 14: 641–64.
- . 1967. *The Theory of Max-Min and Its Application to Weapons Allocation Problems*. Springer.
- Datorro, J. 2015. *Convex Optimization and Distance Geometry*. Second Edition. Palo Alto, CA: Meebo Publishing. [https://ccrma.stanford.edu/~dattorro/0976401304\\_v2015.04.11.pdf](https://ccrma.stanford.edu/~dattorro/0976401304_v2015.04.11.pdf).
- De Boor, C. 1972. “On Calculating with B-splines. II. Integration.” *Journal of Approximation Theory* 6: 50–62.
- . 1976. “Splines as Linear Combination of B-splines. A Survey.” In *Approximation Theory II*, edited by G. G. Lorentz, C. K. Chui, and L. L. Schumaker, 1–47. Academic Press.
- . 2001. *A Practical Guide to Splines*. Revised Edition. New York: Springer-Verlag.
- De Boor, C., and K. Höllig. 1985. “B-splines without Divided Differences.” Technical Report 622. Department of Computer Science, University of Wisconsin-Madison.
- De Boor, C., T. Lyche, and L. L. Schumaker. 1976. “On Calculating with B-splines. II. Integration.” In *Numerische Methoden der Approximationstheorie*, edited by L. Collatz, G. Meinardus, and H. Werner, 123–46.

- Basel: Birkhauser.
- 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. 1968a. "Canonical Discriminant Analysis of Relational Data." Research Note 007-68. Department of Data Theory FSW/RUL. [http://deleeuwpdx.net/janspubs/1968/reports/deleeuw\\_R\\_68e.pdf](http://deleeuwpdx.net/janspubs/1968/reports/deleeuw_R_68e.pdf).
- . 1968b. "Nonmetric Discriminant Analysis." Research Note 06-68. Department of Data Theory, University of Leiden. [http://deleeuwpdx.net/janspubs/1968/reports/deleeuw\\_R\\_68d.pdf](http://deleeuwpdx.net/janspubs/1968/reports/deleeuw_R_68d.pdf).
- . 1968c. "Nonmetric Multidimensional Scaling." Research Note 010-68. Department of Data Theory FSW/RUL. [http://deleeuwpdx.net/janspubs/1968/reports/deleeuw\\_R\\_68g.pdf](http://deleeuwpdx.net/janspubs/1968/reports/deleeuw_R_68g.pdf).
- . 1969. "Some Contributions to the Analysis of Categorical Data." Research Note 004-69. Leiden, The Netherlands: Department of Data Theory FSW/RUL. [http://deleeuwpdx.net/janspubs/1969/reports/deleeuw\\_R\\_69d.pdf](http://deleeuwpdx.net/janspubs/1969/reports/deleeuw_R_69d.pdf).
- . 1970. "The Euclidean Distance Model." Research Note 002-70. Department of Data Theory FSW/RUL. [http://deleeuwpdx.net/janspubs/1970/reports/deleeuw\\_R\\_70b.pdf](http://deleeuwpdx.net/janspubs/1970/reports/deleeuw_R_70b.pdf).
- . 1973a. "Canonical Analysis of Categorical Data." PhD thesis, University of Leiden, The Netherlands.
- . 1973b. "Smoothness Properties of Nonmetric Loss Functions." Technical Memorandum. Murray Hill, N.J.: Bell Telephone Laboratories. [http://deleeuwpdx.net/janspubs/1973/reports/deleeuw\\_R\\_73g.pdf](http://deleeuwpdx.net/janspubs/1973/reports/deleeuw_R_73g.pdf).
- . 1974. "Approximation of a Real Symmetric Matrix by a Positive Semidefinite Matrix of Rank r." Technical Memorandum TM-74-1229-10. Murray Hill, N.J.: Bell Telephone Laboratories. [http://deleeuwpdx.net/janspubs/1974/reports/deleeuw\\_R\\_74c.pdf](http://deleeuwpdx.net/janspubs/1974/reports/deleeuw_R_74c.pdf).
- . 1975a. "A Normalized Cone Regression Approach to Alternating Least Squares Algorithms." Department of Data Theory FSW/RUL. [http://deleeuwpdx.net/janspubs/1975/notes/deleeuw\\_U\\_75a.pdf](http://deleeuwpdx.net/janspubs/1975/notes/deleeuw_U_75a.pdf).
- . 1975b. "An Alternating Least Squares Approach to Squared Distance Scaling." Department of Data Theory FSW/RUL.
- . 1977a. "Applications of Convex Analysis to Multidimensional Scaling." In *Recent Developments in Statistics*, edited by J. R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, 133–45. Amsterdam, The Netherlands: North Holland Publishing Company. [http://deleeuwpdx.net/janspubs/1977/reports/deleeuw\\_R\\_77a.pdf](http://deleeuwpdx.net/janspubs/1977/reports/deleeuw_R_77a.pdf).

- net/janspubs/1977/chapters/deleeuw\_C\_77.pdf.
- . 1977b. “Correctness of Kruskal’s Algorithms for Monotone Regression with Ties.” *Psychometrika* 42: 141–44. [http://deleeuw.pdx.net/janspubs/1977/articles/deleeuw\\_A\\_77.pdf](http://deleeuw.pdx.net/janspubs/1977/articles/deleeuw_A_77.pdf).
- . 1984a. *Canonical Analysis of Categorical Data*. Leiden, The Netherlands: DSWO Press. [http://deleeuw.pdx.net/janspubs/1984/books/deleeuw\\_B\\_84.pdf](http://deleeuw.pdx.net/janspubs/1984/books/deleeuw_B_84.pdf).
- . 1984b. “Convergence of Majorization Algorithms for Multidimensional Scaling.” Research Note RR-84-07. Leiden, The Netherlands: Department of Data Theory FSW/RUL. [http://deleeuw.pdx.net/janspubs/1984/reports/deleeuw\\_R\\_84c.pdf](http://deleeuw.pdx.net/janspubs/1984/reports/deleeuw_R_84c.pdf).
- . 1984c. “Differentiability of Kruskals Stress at a Local Minimum.” *Psychometrika* 49: 111–13. [http://deleeuw.pdx.net/janspubs/1984/articles/deleeuw\\_A\\_84f.pdf](http://deleeuw.pdx.net/janspubs/1984/articles/deleeuw_A_84f.pdf).
- . 1984d. “Fixed-Rank Approximation with Singular Weight Matrices.” *Computational Statistics Quarterly* 1: 3–12. [http://deleeuw.pdx.net/janspubs/1984/articles/deleeuw\\_A\\_84b.pdf](http://deleeuw.pdx.net/janspubs/1984/articles/deleeuw_A_84b.pdf).
- . 1988. “Convergence of the Majorization Method for Multidimensional Scaling.” *Journal of Classification* 5: 163–80. [http://deleeuw.pdx.net/janspubs/1988/articles/deleeuw\\_A\\_88b.pdf](http://deleeuw.pdx.net/janspubs/1988/articles/deleeuw_A_88b.pdf).
- . 1993. “Fitting Distances by Least Squares.” Preprint Series 130. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.pdx.net/janspubs/1993/reports/deleeuw\\_R\\_93c.pdf](http://deleeuw.pdx.net/janspubs/1993/reports/deleeuw_R_93c.pdf).
- . 1994. “Block Relaxation Algorithms in Statistics.” In *Information Systems and Data Analysis*, edited by H. H. Bock, W. Lenski, and M. M. Richter, 308–24. Berlin: Springer Verlag. [http://deleeuw.pdx.net/janspubs/1994/chapters/deleeuw\\_C\\_94c.pdf](http://deleeuw.pdx.net/janspubs/1994/chapters/deleeuw_C_94c.pdf).
- . 2005a. “Applications of Convex Analysis to Multidimensional Scaling.” Preprint Series 448. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.pdx.net/janspubs/2005/reports/deleeuw\\_R\\_05b.pdf](http://deleeuw.pdx.net/janspubs/2005/reports/deleeuw_R_05b.pdf).
- . 2005b. “Fitting Ellipsoids by Least Squares.” UCLA Department of Statistics. [http://deleeuw.pdx.net/janspubs/2005/notes/deleeuw\\_U\\_05j.pdf](http://deleeuw.pdx.net/janspubs/2005/notes/deleeuw_U_05j.pdf).
- . 2005c. “Unidimensional Scaling.” In *The Encyclopedia of Statistics in Behavioral Science*, edited by B. S. Everitt and D. C, 4:2095–97. New York, N.Y.: Wiley. [http://deleeuw.pdx.net/janspubs/2005/chapters/deleeuw\\_C\\_05h.pdf](http://deleeuw.pdx.net/janspubs/2005/chapters/deleeuw_C_05h.pdf).

- . 2006. “Accelerated Least Squares Multidimensional Scaling.” Preprint Series 493. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2006/reports/deleeuw\\_R\\_06b.pdf](http://deleeuw.psu.edu/janspubs/2006/reports/deleeuw_R_06b.pdf).
- . 2007a. “A Horseshoe for Multidimensional Scaling.” Preprint Series 530. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2007/reports/deleeuw\\_R\\_07a.pdf](http://deleeuw.psu.edu/janspubs/2007/reports/deleeuw_R_07a.pdf).
- . 2007b. “Derivatives of Generalized Eigen Systems with Applications.” Preprint Series 528. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2007/reports/deleeuw\\_R\\_07c.pdf](http://deleeuw.psu.edu/janspubs/2007/reports/deleeuw_R_07c.pdf).
- . 2007c. “Quadratic Surface Embedding.” UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2007/notes/deleeuw\\_U\\_07h.pdf](http://deleeuw.psu.edu/janspubs/2007/notes/deleeuw_U_07h.pdf).
- . 2008a. “Accelerating Majorization Algorithms.” Preprint Series 543. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2008/reports/deleeuw\\_R\\_08h.pdf](http://deleeuw.psu.edu/janspubs/2008/reports/deleeuw_R_08h.pdf).
- . 2008b. “Derivatives of Fixed-Rank Approximations.” Preprint Series 547. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2008/reports/deleeuw\\_R\\_08b.pdf](http://deleeuw.psu.edu/janspubs/2008/reports/deleeuw_R_08b.pdf).
- . 2008c. “Polynomial Extrapolation to Accelerate Fixed Point Algorithms.” Preprint Series 542. Los Angeles, CA: UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2008/reports/deleeuw\\_R\\_08i.pdf](http://deleeuw.psu.edu/janspubs/2008/reports/deleeuw_R_08i.pdf).
- . 2012. “Inverse Multidimensional Scaling.” UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2012/notes/deleeuw\\_U\\_12b.pdf](http://deleeuw.psu.edu/janspubs/2012/notes/deleeuw_U_12b.pdf).
- . 2014a. “Bounding, and Sometimes Finding, the Global Minimum in Multidimensional Scaling.” UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2014/notes/deleeuw\\_U\\_14b.pdf](http://deleeuw.psu.edu/janspubs/2014/notes/deleeuw_U_14b.pdf).
- . 2014b. “Minimizing rStress Using Nested Majorization.” UCLA Department of Statistics. [http://deleeuw.psu.edu/janspubs/2014/notes/deleeuw\\_U\\_14c.pdf](http://deleeuw.psu.edu/janspubs/2014/notes/deleeuw_U_14c.pdf).
- . 2015. “Regression with Linear Inequality Restrictions on Predicted Values.” <http://rpubs.com/deleeuw/78897>.
- . 2016a. *Block Relaxation Methods in Statistics*. Bookdown. [http://deleeuw.psu.edu/bookdown/bras/\\_book](http://deleeuw.psu.edu/bookdown/bras/_book).
- . 2016b. “Derivatives of Low Rank PSD Approximation.” 2016. <http://deleeuw.psu.edu/pubfolders/rank/rank.pdf>.

- \_\_\_\_\_. 2016c. “Gower Rank.” 2016. <http://deleeuw.pdx.net/pubfolders/gower/gower.pdf>.
- \_\_\_\_\_. 2016d. *Multivariate Analysis with Optimal Scaling*. Bookdown. [http://deleeuw.pdx.net/bookdown/gifi/\\_book](http://deleeuw.pdx.net/bookdown/gifi/_book).
- \_\_\_\_\_. 2016e. “Pictures of Stress.” 2016. <http://deleeuw.pdx.net/pubfolders/twoPoints/twoPoints.pdf>.
- \_\_\_\_\_. 2017a. “Computing and Fitting Monotone Splines.” 2017. <http://deleeuw.pdx.net/pubfolders/splines/splines.pdf>.
- \_\_\_\_\_. 2017b. “Multidimensional Scaling with Distance Bounds.” 2017. <http://deleeuw.pdx.net/pubfolders/updown/updown.pdf>.
- \_\_\_\_\_. 2017c. “Multidimensional Scaling with Lower Bounds.” 2017. <http://deleeuw.pdx.net/pubfolders/above/above.pdf>.
- \_\_\_\_\_. 2017d. “Multidimensional Scaling with Upper Bounds.” 2017. <http://deleeuw.pdx.net/pubfolders/below/below.pdf>.
- \_\_\_\_\_. 2017e. “Pseudo Confidence Regions for MDS.” 2017. <http://deleeuw.pdx.net/pubfolders/confidence/confidence.pdf>.
- \_\_\_\_\_. 2017f. “Shepard Non-metric Multidimensional Scaling.” 2017. <http://deleeuw.pdx.net/pubfolders/shepard/shepard.pdf>.
- \_\_\_\_\_. 2018a. “Differentiability of Stress at Local Minima.” 2018. <http://deleeuw.pdx.net/pubfolders/zero/zero.pdf>.
- \_\_\_\_\_. 2018b. “MM Algorithms for Smoothed Absolute Values.” 2018. <http://deleeuw.pdx.net/pubfolders/absapp/absapp.pdf>.
- \_\_\_\_\_. 2019. “Normalized Cone Regression.” 2019. <http://deleeuw.pdx.net/pubfolders/dcone/dcone.pdf>.
- \_\_\_\_\_. 2020. *Gifi Analysis of Multivariate Data*. [http://deleeuw.pdx.net/bookdown/gifi/\\_book/\\_main.pdf](http://deleeuw.pdx.net/bookdown/gifi/_book/_main.pdf).
- De Leeuw, J., and P. J. F. Groenen. 1997. “Inverse Multidimensional Scaling.” *Journal of Classification* 14 (3–21). [http://deleeuw.pdx.net/janspubs/1997/articles/deleeuw\\_groenen\\_A\\_97.pdf](http://deleeuw.pdx.net/janspubs/1997/articles/deleeuw_groenen_A_97.pdf).
- De Leeuw, J., P. Groenen, and P. Mair. 2016a. “Minimizing qStress for Small q.” 2016. <https://doi.org/10.13140/RG.2.1.4843.1764>.
- \_\_\_\_\_. 2016b. “Minimizing rStress Using Majorization.” 2016. <https://doi.org/10.13140/RG.2.1.3871.3366>.
- \_\_\_\_\_. 2016c. “More on Inverse Multidimensional Scaling.” 2016. <https://doi.org/10.13140/RG.2.1.1454.5684>.
- \_\_\_\_\_. 2016d. “Second Derivatives of rStress, with Applications.” 2016. <https://doi.org/10.13140/RG.2.1.1058.4081>.
- \_\_\_\_\_. 2016e. “Singularities and Zero Distances in Multidimensional Scal-

- ing.” 2016. <https://doi.org/10.13140/RG.2.1.3339.2403>.
- De Leeuw, J., P. Groenen, and R. Pietersz. 2016. “An Alternating Least Squares Approach to Squared Distance Scaling.” 2016. <https://doi.org/10.13140/RG.2.2.15357.97766>.
- De Leeuw, J., and W. J. Heiser. 1977. “Convergence of Correction Matrix Algorithms for Multidimensional Scaling.” In *Geometric Representations of Relational Data*, edited by J. C. Lingoes, 735–53. Ann Arbor, Michigan: Mathesis Press. [http://deleeuw.psu.edu/janspubs/1977/chapters/deleeuw\\_heiser\\_C\\_77.pdf](http://deleeuw.psu.edu/janspubs/1977/chapters/deleeuw_heiser_C_77.pdf).
- . 1980. “Multidimensional Scaling with Restrictions on the Configuration.” In *Multivariate Analysis, Volume v*, edited by P. R. Krishnaiah, 501–22. Amsterdam, The Netherlands: North Holland Publishing Company. [http://deleeuw.psu.edu/janspubs/1980/chapters/deleeuw\\_heiser\\_C\\_80.pdf](http://deleeuw.psu.edu/janspubs/1980/chapters/deleeuw_heiser_C_80.pdf).
- . 1982. “Theory of Multidimensional Scaling.” In *Handbook of Statistics, Volume II*, edited by P. R. Krishnaiah and L. Kanal. Amsterdam, The Netherlands: North Holland Publishing Company. [http://deleeuw.psu.edu/janspubs/1982/chapters/deleeuw\\_heiser\\_C\\_82.pdf](http://deleeuw.psu.edu/janspubs/1982/chapters/deleeuw_heiser_C_82.pdf).
- De Leeuw, J., K. Hornik, and P. Mair. 2009. “Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods.” *Journal of Statistical Software* 32 (5): 1–24. [http://deleeuw.psu.edu/janspubs/2009/articles/deleeuw\\_hornik\\_mair\\_A\\_09.pdf](http://deleeuw.psu.edu/janspubs/2009/articles/deleeuw_hornik_mair_A_09.pdf).
- De Leeuw, J., and P. Mair. 2009. “Multidimensional Scaling Using Majorization: SMACOF in R.” *Journal of Statistical Software* 31 (3): 1–30. [http://deleeuw.psu.edu/janspubs/2009/articles/deleeuw\\_mair\\_A\\_09c.pdf](http://deleeuw.psu.edu/janspubs/2009/articles/deleeuw_mair_A_09c.pdf).
- De Leeuw, J., and J. J. Meulman. 1986. “Principal Component Analysis and Restricted Multidimensional Scaling.” In *Classification as a Tool of Research*, edited by W. Gaul and M. Schader, 83–96. Amsterdam, London, New York, Tokyo: North-Holland. [http://deleeuw.psu.edu/janspubs/1986/chapters/deleeuw\\_meulman\\_C\\_86.pdf](http://deleeuw.psu.edu/janspubs/1986/chapters/deleeuw_meulman_C_86.pdf).
- De Leeuw, J., and K. Sorenson. 2012. “Derivatives of the Procrustes Transformation with Applications.” [http://deleeuw.psu.edu/janspubs/2012/notes/deleeuw\\_sorenson\\_U\\_12b.pdf](http://deleeuw.psu.edu/janspubs/2012/notes/deleeuw_sorenson_U_12b.pdf).
- De Leeuw, J., and I. Stoop. 1984. “Upper Bounds for Kruskal’s Stress.” *Psychometrika* 49: 391–402. [http://deleeuw.psu.edu/janspubs/1984/articles/deleeuw\\_stoop\\_A\\_84.pdf](http://deleeuw.psu.edu/janspubs/1984/articles/deleeuw_stoop_A_84.pdf).
- Defays, D. 1978. “A Short Note on a Method of Seriation.” *British Journal of Mathematical and Statistical Psychology* 31: 49–53.

- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum Likelihood for Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*39: 1–38.
- Demyanov, V. F., and V. N. Malozemov. 1990. *Introduction to Minimax*. Dover.
- Dinkelbach, W. 1967. "On Nonlinear Fractional Programming." *Management Science* 13: 492–98.
- Dür, M., R. Horst, and M. Locatelli. 1998. "Necessary and Sufficient Global Optimality Conditions for Convex Maximization Revisited." *Journal of Mathematical Analysis and Applications* 217: 637–49.
- Eckart, C., and G. Young. 1936. "The Approximation of One Matrix by Another of Lower Rank." *Psychometrika* 1 (3): 211–18.
- Ekman, G. 1954. "Dimensions of Color Vision." *Journal of Psychology* 38: 467–74.
- Flett, T. M. 1980. *Differential Analysis*. Cambridge University Press.
- Gaffney, P. W. 1976. "The Calculation of Indefinite Integrals of B-splines." *Journal of the Institute of Mathematics and Its Applications* 17: 37–41.
- Gifi, A. 1990. *Nonlinear Multivariate Analysis*. New York, N.Y.: Wiley.
- Gohberg, I., P. Lancaster, and L. Rodman. 2009. *Matrix Polynomials*. Classic in Applied Mathematics. SIAM.
- Gold, E. Mark. 1973. "Metric Unfolding: Data Requirement for Unique Solution & Clarification of Schönemann's Algorithm." *Psychometrika* 38 (4): 555–69.
- Gower, J. C. 1966. "Some Distance Properties of Latent Root and Vector Methods Used in Multivariate Analysis." *Biometrika* 53: 325–38.
- Greenacre, M. J., and M. W. Browne. 1986. "An Efficient Alternating Least Squares Algorithm to Perform Multidimensional Unfolding." *Psychometrika* 51 (2): 241–50.
- Groenen, P. J. F., and J. De Leeuw. 2010. "Power-Stress for Multidimensional Scaling." [http://deleeuw.psu.edu/janspubs/2010/notes/groenen\\_deleeuw\\_U\\_10.pdf](http://deleeuw.psu.edu/janspubs/2010/notes/groenen_deleeuw_U_10.pdf).
- Groenen, P. J. F., P. Giaquinto, and H. A. L. Kiers. 2003. "Weighted Majorization Algorithms for Weighted Least Squares Decomposition Models." EI 2003-09. Rotterdam, Netherlands: Econometric Institute, Erasmus University.
- Groenen, P. J. F., W. J. Heiser, and J. J. Meulman. 1998. "City-Block Scaling: Smoothing Strategies for Avoiding Local Minima." In *Classification, Data Analysis, and Data Highways*, edited by I. Balderjahn, R. Mathar,

- and M. Schader. Springer.
- . 1999. “Global Optimization in Least-Squares Multidimensional Scaling by Distance Smoothing.” *Journal of Classification* 16: 225–54.
- Groenen, P. J. F., and M. Van de Velden. 2016. “Multidimensional Scaling by Majorization: A Review.” *Journal of Statistical Software* 73 (8): 1–26. <https://doi.org/10.18637/jss.v073.i08>.
- Guilford, J. P. 1954. *Psychometric Methods*. McGraw-Hill.
- Guttman, L. 1941. “The Quantification of a Class of Attributes: A Theory and Method of Scale Construction.” In *The Prediction of Personal Adjustment*, edited by P. Horst, 321–48. New York: Social Science Research Council.
- . 1967. “The Development of Nonmetric Space Analysis: A Letter to Professor John Ross.” *Multivariate Behavioral Research* 2 (1): 71–82.
- . 1968. “A General Nonmetric Technique for Fitting the Smallest Coordinate Space for a Configuration of Points.” *Psychometrika* 33: 469–506.
- Harman, R., and V. Lacko. 2010. “On Decompositional Algorithms for Uniform Sampling from n-Spheres and n-Balls.” *Journal of Multivariate Analysis* 101: 2297–2304.
- Harshman, R. A. 1970. “Foundations of the PARAFAC Procedure.” Working Papers in Phonetics 16. UCLA. <https://www.psychology.uwo.ca/faculty/harshman/wppffac0.pdf>.
- Heiser, W. J. 1988. “Multidimensional Scaling with Least Absolute Residuals.” In *Classification and Related Methods of Data Analysis*, edited by H. H. Bock, 455–62. North-Holland Publishing Co.
- . 1991. “A Generalized Majorization Method for Least Squares Multidimensional Scaling of Pseudodistances That May Be Negative.” *Psychometrika* 56 (1): 7–27.
- . 1995. “Convergent Computing by Iterative Majorization: Theory and Applications in Multidimensional Data Analysis.” In *Recent Advances in Descriptive Multivariate Analysis*, edited by W. J. Krzakowski. Oxford, England: Clarendon Press.
- Heiser, W. J., and J. De Leeuw. 1977. “How to Use SMACOF-i.” Department of Data Theory FSW/RUL. [http://deleeuw.pdx.net/janspubs/1977/reports/heiser\\_deleeuw\\_R\\_77.pdf](http://deleeuw.pdx.net/janspubs/1977/reports/heiser_deleeuw_R_77.pdf).
- . 1979. “Metric Multidimensional Unfolding.” *Methoden En Data Nieuwsbrief SWS/VVS* 4: 26–50. [http://deleeuw.pdx.net/janspubs/1979/articles/heiser\\_deleeuw\\_A\\_79.pdf](http://deleeuw.pdx.net/janspubs/1979/articles/heiser_deleeuw_A_79.pdf).

- Hiriart-Urruty, J.-B. 1988. “Generalized Differentiability / Duality and Optimization for Problems Dealing with Differences of Convex Functions.” In *Convexity and Duality in Optimization*, edited by Ponstein. J., 37–70. Lecture Notes in Economics and Mathematical Systems 256. Springer.
- Hiriart-Urruty, J.-B., and C. Lemaréchal. 1993. *Convex Analysis and Minimization Algorithms i: Fundamentals*. Springer.
- Keller, J. B. 1962. “Factorization of Matrices by Least Squares.” *Biometrika* 49: 239–42.
- Kiers, H. A. L. 1997. “Weighted Least Squares Fitting Using Iterative Ordinary Least Squares Algorithms.” *Psychometrika* 62: 251–66.
- Kruskal, J. B. 1964a. “Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis.” *Psychometrika* 29: 1–27.
- . 1964b. “Nonmetric Multidimensional Scaling: a Numerical Method.” *Psychometrika* 29: 115–29.
- . 1971. “Monotone Regression: Continuity and Differentiability Properties.” *Psychometrika* 36 (1): 57–62.
- Kruskal, J. B., and J. D. Carroll. 1969. “Geometrical Models and Badness of Fit Functions.” In *Multivariate Analysis, Volume II*, edited by P. R. Krishnaiah, 639–71. North Holland Publishing Company.
- Lange, K. 2016. *MM Optimization Algorithms*. SIAM.
- Lawlor, G. R. 2020. “l’Hôpital’s Rule for Multivariable Functions.” *American Mathematical Monthly* 127 (8): 717–25.
- Levelt, W. J. M., J. P. Van De Geer, and R. Plomp. 1966. “Triadic Comparisons of Musical Intervals.” *British Journal of Mathematical and Statistical Psychology* 19: 163–79.
- Lingoes, J. C. 1968a. “The Multivariate Analysis Of Qualitative Data.” *Multivariate Behavioral Research* 3 (1): 61–94.
- . 1968b. “The Rationale of the Guttman-Lingoes Nonmetric Series: A Letter to Doctor Philip Runkel.” *Multivariate Behavioral Research* 3 (4): 495–507.
- . 1971. “Some Boundary Conditions for a Monotone Analysis of Symmetric Matrices.” *Psychometrika* 36 (2): 195–203.
- Lingoes, J. C., and E. E. Roskam. 1973. “A Mathematical and Empirical Analysis of Two Multidimensional Scaling Algorithms.” *Psychometrika* 38: Monograph Supplement.
- Lyusternik, L., and L. Schnirelmann. 1934. *Méthodes Topologiques Dans Les Problèmes Variationnelle*. Hermann.
- Maehara, H. 1986. “Metric Transforms of Finite Spaces and Connected

- Graphs.” *Discrete Mathematics* 61: 235–46.
- Magnus, J. R., and H. Neudecker. 1999. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Revised Edition. Wiley.
- Mair, P., P. J. F. Groenen, and J. De Leeuw. 2019. “More on Multidimensional Scaling in R: smacof Version 2.” *Journal of Statistical Software*. [http://deleeuwpdx.net/janspubs/2019/articles/mair\\_groenen\\_deleeuw\\_A\\_19.pdf](http://deleeuwpdx.net/janspubs/2019/articles/mair_groenen_deleeuw_A_19.pdf).
- Messick, S. J., and R. P. Abelson. 1956. “The Additive Constant Problem in Multidimensional Scaling.” *Psychometrika* 21 (1–17).
- Meulman, J. J. 1986. “A Distance Approach to Nonlinear Multivariate Analysis.” PhD thesis, Leiden University.
- . 1992. “The Integration of Multidimensional Scaling and Multivariate Analysis with Optimal Transformations.” *Psychometrika* 57 (4): 539–65.
- Meulman, J. M., and W. J. Heiser. 2012. *IBM SPSS Categories 21*. IBM Corporation.
- Ortega, J. M., and W. C. Rheinboldt. 1970. *Iterative Solution of Nonlinear Equations in Several Variables*. New York, N.Y.: Academic Press.
- Palubeckis, G. 2013. “An Improved Exact Algorithm for Least-Squares Unidimensional Scaling.” *Journal of Applied Mathematics*, 1–15.
- Papazoglou, S., and K. Mylonas. 2017. “An Examination of Alternative Multidimensional Scaling Techniques.” *Educational and Psychological Measurement* 77 (3): 429–48.
- Pliner, V. 1984. “A Class of Metric Scaling Models.” *Automation and Remote Control* 45: 789–94.
- . 1986. “The Problem of Multidimensional Metric Scaling.” *Automation and Remote Control* 47: 560–67.
- . 1996. “Metric Unidimensional Scaling and Global Optimization.” *Journal of Classification* 13: 3–18.
- Ramirez, C., R. Sanchez, V. Kreinovich, and M. Argaez. 2014. “ $\sqrt{x^2 + \mu}$  is the Most Computationally Efficient Smooth Approximation to  $|x|$ .” *Journal of Uncertain Systems* 8: 205–10.
- Ramsay, J. O. 1977. “Maximum Likelihood Estimation in Multidimensional Scaling.” *Psychometrika* 42: 241–66.
- . 1988. “Monotone Regression Splines in Action.” *Statistical Science* 3: 425–61.
- Robert, F. 1967. “Calcul du Rapport Maximal de Deux Normes sur  $\mathbb{R}^n$ .” *Revue Francaise d'Automatique, d'Informatique Et De Recherche Operative*

- tionnelle* 1: 97–118.
- Rockafellar, R. T. 1970. *Convex Analysis*. Princeton University Press.
- Roskam, E. E. 1968. “Metric Analysis of Ordinal Data in Psychology.” PhD thesis, University of Leiden.
- Ross, J., and N. Cliff. 1964. “A Generalization of the Interpoint Distance Model.” *Psychometrika* 29 (167-176).
- Rothkopf, E. Z. 1957. “A Measure of Stimulus Similarity and Errors in some Paired-associate Learning.” *Journal of Experimental Psychology* 53: 94–101.
- Schoenberg, I. J. 1935. “Remarks to Maurice Frechet’s article: Sur la Definition Axiomatique d’une Classe d’Espaces Vectoriels Distancies Appliables Vectoriellement sur l’Espace de Hllbert.” *Annals of Mathematics* 36: 724–32.
- \_\_\_\_\_. 1937. “On Certain Metric Spaces Arising From Euclidean Spaces by a Change of Metric and Their Imbedding in Hilbert Space.” *Annals of Mathematics* 38 (4): 787–93.
- Schönemann, P. H. 1970. “On Metric Multidimensional Unfolding.” *Psychometrika* 35 (3): 349–66.
- Schumaker, L. 2007. *Spline Functions: Basic Theory*. Third Edition. Cambridge University Press.
- Shepard, R. N. 1962a. “The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. I.” *Psychometrika* 27: 125–40.
- \_\_\_\_\_. 1962b. “The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II.” *Psychometrika* 27: 219–46.
- Spang, H. A. 1962. “A Review of Minimization Techniques for Nonlinear Functions.” *SIAM Review* 4 (4): 343–65.
- Stephenson, W. 1953. *The Study of Behavior: Q-technique and its Methodology*. University of Chicago Press.
- Takane, Y., F. W. Young, and J. De Leeuw. 1977. “Nonmetric Individual Differences in Multidimensional Scaling: An Alternating Least Squares Method with Optimal Scaling Features.” *Psychometrika* 42: 7–67. [http://deleeuw.psu.edu/janspubs/1977/articles/takane\\_young\\_deleeuw\\_A\\_77.pdf](http://deleeuw.psu.edu/janspubs/1977/articles/takane_young_deleeuw_A_77.pdf).
- Taussky, O. 1949. “A Recurring Theorem on Determinants.” *American Mathematical Monthly* 56: 672–76.
- Tisseur, F., and K. Meerbergen. 2001. “The Quadratic Eigenvalue Problem.” *SIAM Review* 43 (2): 235–86.
- Torgerson, W. S. 1952. “Multidimensional Scaling: I. Theory and Method.”

- Psychometrika* 17 (4): 401–19.
- . 1958. *Theory and Methods of Scaling*. New York: Wiley.
- . 1965. “Multidimensional Scaling of Similarity.” *Psychometrika* 30 (4): 379–93.
- Trosset, M. W., and R. Mathar. 1997. “On the Existence on Nonglobal Minimizers of the STRESS Criterion for Metric Multidimensional Scaling.” In *Proceedings of the Statistical Computing Section*, 158–62. Alexandria, VA: American Statistical Association.
- Turlach, B. A., and A. Weingessel. 2013. *quadprog: Functions to solve Quadratic Programming Problems*. <https://CRAN.R-project.org/package=quadprog>.
- Tuy, H. 1998. *Convex Analysis and Global Optimization*. Kluwer Academic Publishers.
- Van de Geer, J. P. 1967. *Inleiding in de Multivariate Analyse*. Van Loghum Slaterus.
- . 1971. *Introduction to Multivariate Analysis for the Social Sciences*. San Francisco, CA: Freeman.
- Van de Geer, J. P., W. J. M. Levelt, and R. Plomp. 1962. “The Connotation of Musical Consonance.” *Acta Psychologica* 20: 308–19.
- Vesely, L., and L. Zajicek. 1989. “Delta-convex Mappings between Banach Spaces and Applications.” *Dissertationes Mathematicae* 289: 1–48.
- Wang, Y., C. L. Lawson, and R. J. Hanson. 2015. *lsei: Solving Least Squares Problems under Equality/Inequality Constraints*. <http://CRAN.R-project.org/package=lsei>.
- Young, F. W. 1981. “Quantitative Analysis of Qualitative Data.” *Psychometrika* 46: 357–88.
- Young, F. W., Y. Takane, and R. Lewyckyj. 1978a. “ALSCAL: A Nonmetric Multidimensional Scaling Program with Several Individual-Differences Options.” *Behavior Research Methods & Instrumentation* 10 (3): 451–53.
- . 1978b. “Three Notes on ALSCAL.” *Psychometrika* 43 (3): 433–35.
- Young, G., and A. S. Householder. 1938. “Discussion of a Set of Points in Terms of Their Mutual Distances.” *Psychometrika* 3 (19–22).
- Zangwill, W. I. 1969. *Nonlinear Programming: a Unified Approach*. Englewood-Cliffs, N.J.: Prentice-Hall.
- Zilinskas, A., and A. Poslipskyte. 2003. “On Multimodality of the SSTRESS Criterion for Metric Multidimensional Scaling.” *Informatica* 14 (1): 121–30.