Published in final edited form as:

Math J. 2015 ; 17: . doi:10.3888/tmj.17-7.

Graphical Representation of Proximity Measures for Multidimensional Data:

Classical and Metric Multidimensional Scaling

Martin S. Zand, MD, PhD,

University of Rochester Medical Center, 601 Elmwood Avenue - Box 675, Rochester, NY 14618

Jiong Wang, PhD, and

University of Rochester Medical Center, 601 Elmwood Avenue - Box 675, Rochester, NY 14618

Shannon Hilchey, PhD

University of Rochester Medical Center, 601 Elmwood Avenue - Box 675, Rochester, NY 14618

Martin S. Zand: martin_zand@urmc.rochester.edu; Jiong Wang: jiong_wang@urmc.rochester.edu; Shannon Hilchey: shannon_hilchey@urmc.rochester.edu

Abstract

We describe the use of classical and metric multidimensional scaling methods for graphical representation of the proximity between collections of data consisting of cases characterized by multidimensional attributes. These methods can preserve metric differences between cases, while allowing for dimensional reduction and projection to two or three dimensions ideal for data exploration. We demonstrate these methods with three datasets for: (i) the immunological similarity of influenza proteins measured by a multidimensional assay; (ii) influenza protein sequence similarity; and (iii) reconstruction of airport-relative locations from paired proximity measurements. These examples highlight the use of proximity matrices, eigenvalues, eigenvectors, and linear and nonlinear mappings using numerical minimization methods. Some considerations and caveats for each method are also discussed, and compact *Mathematica* programs are provided.

1. Introduction

One “curse” of high- and multidimensional data is the difficulty in graphically displaying how the data points are related to each other. Each data point (in some fields referred to as a case) is characterized by a vector of attributes (e.g. shape, color, temperature, etc.) with numerical values that form a set of coordinates. These coordinates specify a location in an N -dimensional space, with N being the number of variables used to describe each case. Visualization of such N -dimensional data is problematic as, for example, we do not think in 17 dimensions! For human comprehension, we are limited to two- or three-dimensional graphics, sometimes referred to as visualizable dimensions, in which each case is represented by a data point. The distances between points and their relative locations reflect their proximity (similarity or dissimilarity), as measured by a metric function of their attributes, such as Euclidean distance. Larger distances between cases indicate that they are less related. Fortunately, multidimensional scaling (MDS) methods can be used to visualize the degree of proximity for high-dimensional datasets [1, 2, 3]. MDS methods can be used to project the N -dimensional coordinates of each case to two or three dimensions, while

preserving the relative distances between cases [4, 5]. These methods are increasingly employed in such diverse fields as immunology, molecular biology, marketing, perceptual mapping, and ecology to visualize relative “distances” between viruses, molecules, species, and customers’ behavior. In this article, we discuss some mathematical details of classical and metric MDS methods and build generalizable *Mathematica* programs that can be used for each. We incorporate use cases from geography, molecular virology, and immunology.

Before covering the analytical details, it is worth briefly discussing the variety of MDS methods. The simplest method is referred to as classical multidimensional scaling, also known as principal *coordinate* analysis (PCoA) [1]. PCoA is not to be confused with principal *component* analysis (PCA). The difference between PCA and classical MDS/PCoA is primarily based on the input data. PCA starts with a set of cases or data points z_i , each described by a set of attributes a , and uses a Euclidean distance to project the data to a lower-dimensional space oriented to maximize the variance observed between data points. In contrast, classical MDS begins with an input matrix that specifies proximities between pairs of items, often referred to as dissimilarities. It outputs a coordinate matrix in a lower dimension than the starting data (e.g. 120 dimensions projected to two) that preserves the relative distances between all data points. Classical MDS was originally developed by Gower [1] and others and is often demonstrated with a simple problem: given only the distances between entities (e.g. cities), is it possible to calculate the map showing their relative distances? We use this example in Section 4. In the specific case where the dissimilarity is measured by a Euclidean distance, PCA and classical MDS/PCoA are mathematically identical to another dimensional reduction method, singular value decomposition (SVD). In that case, there may be computational advantages to using SVD, which are discussed in this article.

While classical MDS relies on linear transformation of data using matrix operations, metric MDS methods depend on computational optimization. Metric MDS takes a proximity matrix and calculates a two- or three-dimensional coordinate matrix whose configuration minimizes the difference between the N -dimensional and the calculated distances in the reduced dimensions for each case or data point [2, 3]. Thus, similar cases are located next to each other in a reduced-dimensional plot, while disparate cases are farther away, and the final coordinates can be projected to a planar or three-dimensional space. Other varieties of MDS, not discussed here, include nonmetric MDS, which uses isotonic regression methods, and generalized MDS, which projects the coordinates to an arbitrary smooth target space [3–7]. Metric MDS is more flexible than classical MDS for certain problems, including when the proximity matrix contains missing data or reflects a distance function that is not Euclidean. In addition, it can accommodate nonlinear mappings from the N -dimensional data space to the visualization. However, because metric MDS uses numerical optimization, it is computationally expensive. These tradeoffs are explored in Section 7.

A byproduct of both MDS methods is the proximity matrix, which can also be used for other high-dimensional visualizations, such as heat mapping and multidimensional clustering methods. Combining MDS with these methods often reveals different facets of the data and aids investigators in gaining understanding of the underlying data structure and exploratory data analysis.

In the following sections, we discuss construction of proximity matrices, classical MDS, and metric MDS. Several examples illustrate the strengths and pitfalls of these methods; the full datasets are included in the article. The tradeoff between computational efficiency and accuracy is touched on as well. Each section briefly covers the mathematical underpinning of the method, a step-by-step example, a self-contained function using compact code, considerations for the method and alternate approaches, and a synopsis of the most salient points. A separate section discusses accuracy and computational efficiency and compares these methods.

2. Computational Environment and Data

This article was created using Wolfram *Mathematica* Version 10.0.2, running Macintosh OS X 10.9.5 on a MacBook Pro with a 2.8 GHz Intel Core i7 processor with 16 GB RAM. It contains the entire code and all the data needed for execution of all examples. Internet connectivity is required to retrieve the airport distances and perform the mapping of the airport locations within *Mathematica*.

For the sake of aesthetics and clarity, we have collapsed cells containing code related to the display of tables and figures, which appear in their final form.

Execution of the last section of the article may take 4–8 minutes, given the number of minimization problems that are calculated. Also, earlier versions of *Mathematica* may not execute the `GeoListPlot` commands properly (or at all for Version 9.0 or lower), given that this functionality was added in Version 10.0 and several options were added between Versions 10.0 and 10.0.2.

3. Datasets and Display Functions

For the sake of brevity, the datasets are placed here in collapsed cells, along with several house-keeping functions for displaying tables and figures. Some of the data for intra-airport distances is retrieved by using the `AirportData` function and requires internet access if executing the note-book again.

```
In[136]:= airLabsI = {"Atlanta", "Billings", "Birmingham", "Bismark",
                    "Boise", "Boston", "Buffalo", "Chicago", "Cleveland",
                    "Dallas", "Denver", "Des Moines", "Detroit", "El Paso",
                    "Houston", "Indianapolis", "Kansas City", "Little Rock",
                    "Los Angeles", "Louisville", "Memphis", "Miami",
                    "Minneapolis", "New Orleans", "New York", "Omaha",
                    "Philadelphia", "Phoenix", "Pittsburgh", "Portland",
                    "Raleigh-Durham", "St. Louis", "Salt Lake City",
                    "San Francisco", "Seattle", "Washington"};
airLabsShort = {"ATL", "BIL", "BHM", "BIS", "BOI", "BOS",
                "BUF", "ORD", "CLE", "DFW", "DEN", "DSM", "DTW", "ELP",
                "HOU", "IND", "MCI", "LIT", "LAX", "SDF", "MEM", "MIA",
                "MSP", "MSY", "JFK", "OMA", "PHL", "PHX", "PIT", "PDX",
```

```

"RDU", "STL", "SLC", "SFO", "SEA", "DCA");
airLabs =Table[airLabsI[[i]]<>"
(" <>airLabsShort[[i]]<>" ),
    {i, Length[airLabsShort]}}];
geoCoordsAir =
    airGeo =AirportData[#, "Coordinates"]& / @ airLabsShort;
airData =Table[Round[GeoDistance[airGeo[[i]], airGeo[[j]]]]
[[1]],
    {i, 1, Length[airGeo]}, {j, 1, Length[airGeo]}}];

In[141]:= fluData ={{5586, 274, 491, 246, 73, 0, 0, 0, 0, 16, 0, 0, 61, 47},
    {4876, 332, 441, 215, 131, 74, 27, 59, 51, 8, 54, 78,
94, 158},
    {5428, 819, 238, 165, 426, 55, 20, 33, 33, 26, 88,
78, 97, 109},
    {260, 720, 412, 707, 951, 47, 2, 12, 4, 22, 15, 33,
281, 223},
    {303, 805, 530, 7472, 1204, 135, 109, 92, 77, 75,
206, 213,
    451, 447}, {53, 236, 195, 527, 5791, 96, 32, 9, 28,
28,
    115, 103, 193, 129}, {137, 119, 157, 81, 135, 719,
1547,
    1551, 2045, 3914, 1701, 1775, 116, 220},
    {151, 100, 76, 118, 89, 3403, 5910, 4609, 1006, 6561,
1324,
    1821, 49, 142}, {86, 49, 97, 147, 52, 3391, 5159,
3621,
    717, 3972, 1352, 1578, 74, 124},
    {84, 13, 95, 145, 56, 3784, 5520, 3443, 703, 3561,
1119,
    1292, 54, 112}, {0, 0, 0, 0, 0, 4727, 7192, 5422,
491,
    4336, 832, 1284, 0, 0},
    {59, 157, 147, 335, 434, 48, 33, 7, 33, 0, 101, 121,
8806, 5432},
    {184, 304, 183, 350, 211, 290, 232, 147, 271, 210,
442,
    474, 1601, 1538}, {24, 59, 76, 179, 55, 12, 29, 1,
44,
    17, 108, 124, 3044, 4330},
    {94, 581, 133, 147, 317, 60, 17, 8, 46, 16, 80, 95,
5223, 5164}}];

```

```

In[142]:= seraLabLong ={"A/California/07/2009 (H1)",
                        "A/Mexico/4108/2009 (H1)", "A/Utah/20/2009 (H1)",
                        "A/Brisbane/59/2007 (H1)", "A/USSR/90/1977 (H1)",
                        "A/Taiwan/1/1986 (H1)", "A/Indiana/10/2011 (H3)",
                        "A/Victoria/361/2011 (H3)", "A/Wisconsin/15/2009 (H3)",
                        "A/Perth/16/2009 (H3)", "A/Victoria/210/2009 (H3)",
                        "A/Vietnam/1203/2004 (H5)", "A/India/NIV/2006 (H5)",
                        "A/Anhui/01/2005 (H5)", "A/Hubei/1/2010 (H5)"};

In[143]:= seraLabShort ={"CAL709", "MEX4109", "UT2009", "BR5907",
                        "USSR77", "TW186", "INDNA1011", "VIC361", "WISC09",
                        "PER1609", "VIC21009", "VIET1203", "IND2006",
                        "ANH0105",
                        "HUB0110"};

In[144]:= seraColor =
                        Flatten[{ConstantArray[Black, 6], ConstantArray[Blue,
5],
                        ConstantArray[Red, 4]}];

In[145]:= virusLabLong ={"A/California/04/2009 (H1)",
                        "A/South Carolina/01/1918 (H1)", "A/Puerto Rico/
08/1934 (H1)",
                        "A/USSR/01/1977 (H1)", "A/Texas/A/36/1991 (H1)",
                        "A/Perth/16/2009 (H3)", "A/Victoria/361/2011 (H3)",
                        "A/Texas/50/2012 (H3)", "A/Wisconsin/67/2005 (H3)",
                        "A/Hiroshima/52/2005 (H3)", "A/Port Chalmers/01/1973
(H3)",
                        "A/Hong Kong/01/1968 (H3)", "A/Viet/1204/2005 (H5)",
                        "A/Indonesia/05/05 (H5)"};

In[146]:= virusLabShort ={"CAL709-H1", "SC0118-H1", "PR34-H1",
                        "USSR77-H1", "TX91-H1", "PER1609-H3", "VIC361-H3",
                        "TX12-H3", "WISC05-H3", "HIRO05-H3", "PC73-H3", "HK68-
H3",
                        "VIET05-H5", "INDO05-H5"};

In[147]:= seqFluHA =

{"MKAILVLLYTFATANADTLCIGYHANNSTDVDTVLEKNVTVTHSVNLLEDKHNGKLCCL".

RGVAPLHLGKCNIGWILGNPECESLSTASSWSYIVETPSSDNGTCYPGDFIDYE".

ELREQLLSSVSSFERFEIFPKTSSWPNHDSNKGVTAACPHAGAKSFYKNLIWLVKK".

```

GNSYPKLSKSYINDKGKEVLVLWGIHHPPTSADQQSLYQNADAYVFGSSRYSKK'.
 FKPEIAIRPKVRDQEGRMNYYWTLVEPGDKITFEATGNLVVPRYAFAMERNAGSG'.
 IIISDTPVHDCNTTCQTPKGAINSTLPPQNIHPITIGKCPKYVKSTKLRLATGLR'.
 NIPSIQSRGLFGAIAAGFIEGGWTGMVDGWYGYHHQNEQSGSYAADLKSTQNAIDE'.
 ITNKVNSVIEKMTQFTAVGKEFNHLEKRIENLNKKVDDGFLDIWTYNAELLVLL'.
 ENERTLDYHDSNVKNLYEKVRSQKNNAKEIGNGCFEFYHKCDNTCMESVKNGTY'.
 DYPKYSEEAKLNREEIDGVKLESTRIYQILAIYSTVASSLVLVSLGAISFWMCS'.
 NGSQCRICI",
 "MKAILVLLYTFATANADTLCIGYHANNSTDVDTVLEKNVTVTHSVNLLEDKHNGKLCKL'.
 RGVAPHLGKCNIAWILGNPECESLSTASSWSYIVETSSSDNGTCYPGDFIDYE'.
 ELREQSSVSSFERFEIFPKTSSWPNHDSNKGVTAAACPHAGAKSFYKNLIWL VKK'.
 GNSYPKLSKSYINDKGKEVLVLWGIHHPSTSADQQSLYQNADAYVFGSSRYSKK'.
 FKPEIAIRPKVRDQEGRMNYYWTLVEPGDKITFEATGNLVVPRYAFAMERNAGSG'.
 IIISDTPVHDCNTTCQTPKGAINSTLPPQNIHPITIGKCPKYVKSTKLRLATGLR'.
 NVPSIQSRGLFGAIAAGFIEGGWTGMVDGWYGYHHQNEQSGSYAADLKSTQNAIDE'.
 ITNKVNSVIEKMTQFTAVGKEFNHLEKRIENLNKKVDDGFLDIWTYNAELLILL'.
 ENERTLDYHDSNVKNLYEKVRSQKNNAKEIGNGCFEFYHKCDNTCMESVKNGTY'.
 DYPKYSEEAKLNREEIDGVKLESTRIYQILAIYSTVASSLVLVSLGAISFWMCS'.
 NGSQCRICI",
 "MEAILVLLYTFAIANADTLCIGYHANNSTDVDTVLEKNVTVTHSVNLLEDKHNGKLCKL'.
 RGVAPHLGKCNIAWILGNPECESLSTASSWSYIVETSSSDNGTCYPGDFIDYE'.
 ELREQSSVSSFERFEIFPKTSSWPNHDSNKGVTAAACPHAGAKSFYKNLIWL VKK'.
 GDSYPKLSKSYINDKGKEVLVLWGIHHPSTSADQQSLYQNADAYVFGSSRYSKK'.

FKPEIAIRPKVRDQXGRMNYWTLVEPGDKITFEATGNLVVPRYAFAMERNAGSG⁺.
 IIISDTPVHDCNTTCQTPKGAINSTLPHNIHPITIGKCPKYVKSTKLRLATGLR⁺.
 NVPSIQSRGLFGAIAAGFIEGGWTGMVDGWYGYHHQNEQSGYAADLKSTQNAIDE⁺.
 ITNKVNSVIEKMNTQFTAVGKEFNHLEKRIENLNKKVDDGFLDIWTYNAELLVLL⁺.
 ENERTLDYHDSNVKNLYEKVRSQKNNAKEIGNGCFEYHKCDNTCMESVKNGTY⁺.
 DYPKYSEEAKLNREEIDGVKLESTRIYQILAIYSTVASSLVLVSLGAISFWMCS⁺.
 NGS_{LQCRICI}" ,
 "MKVKLLVLLCTFTATYADTICIGYHANNSTDTVDTVLEKNVTVTHSVNLLENHNGKLCCL⁺.
 KGIAPLQLGNCSVAGWILGNPECELLISKESWSYIVEKPNPENGTCYPGHFADYE⁺.
 ELREQLSSVSSFERFEIFPKESSWPNHNTVTGVSASC SHNGESSFYRNLLWLTGKN⁺.
 GLYPNLSKSYANNKEKEVLVLWGVHHPNIGNQKALYHTENAYVSVVSSHYSRKF⁺.
 TPEIAKRPKVRDQEGRINYWTLLPEGDTIIFEANGNLIAPRYAFALSRGFGSGI⁺.
 INSAPMDKCDAKCQTPQGAINSSLPFQNVHPVTIGECPKYVRSAKLRMTGLRN⁺.
 IPSIQSRGLFGAIAAGFIEGGWTGMVDGWYGYHHQNEQSGYAADQKSTQNAINGI⁺.
 TNKVNSVIEKMNTQFTAVGKEFNKLERRMENLNKKVDDGFIDIWTYNAELLVLE⁺.
 NERTLDFHDSNVKNLYEKVKSQKNNAKEIGNGCFEYHKCNDECME SVKNGTYD⁺.
 YPKYSEESKLNREKIDGVKLESMGVYQILAIYSTVASSLVLLVSLGAISFWMCSN⁺.
 G_SL_{QCRICI}" ,
 "MKAKLLVLLCALSATDADTICIGYHANNSTDTVDTVLEKNVTVTHSVNLLED SHNGKLCRL⁺.
 KGIAPLQLGKCNIAGWILGNPECESLFSKKSWSYIAETPNSENGTCYPGYFADYE⁺.
 ELREQLSSVSSFERFEIFPKERSWPKNHVTRGV TASC SHKGKSSFYRNLLWLTEK⁺.
 NGSYPNLSKSYVNNKEKEVLVLWGVHHPNIEDQKTIYRKENAYVSVVSSNYNRR⁺.
 FTPEIAERP KVRGQAGRINYWTLLPEGDTIIFEANGNLIAPWHAFALNRGFGSG⁺.

IITSNASMDECDTKCQTPQGAINSSLPFQNIHPVTIGECPKYVRSTKLRMVTGLR'.
 NIPSIQSRGLFGAIAAGFIEGGWTGMIDGWYGYHHQNEQSGYAADQKSTQNAING'.
 ITNKVNSVIEKMNTQFTAVGKEFNKLEKRMENLNKKVDDGFLDIWTYNAELLVLL'.
 ENERTLDFHDSNVKNLYEKVKSQKNNAKEIGNGCFEFYHKCNNECMESVKNGTY'.
 DYPKYSEESKLNREKIDGVKLESMGVYQILAIYSTVASSLVLLVSLGAISFWMCS'.
 NGSLQCRICI",
 "MKAKLLVLLCAFTATDADTICIGYHANNSTDTVDTVLEKNVTVTHSVNLLEDSSHNGKLCRL'.
 KGIAPLQLGNCSIAGWILGNPECESLFSKKSWSYIAETPNSENGTCYPGYFADYE'.
 ELREQSSVSSFERFEIFPKESSWPNHVTVKGVTASCSHGKSSFYRNLLWLTEK'.
 NGSYPTLSKSYVNNKEKEVLVLWGVHHPNIGDQRAIYHTENAYVSVSSHYNRR'.
 FTPEIAKRPKVRGQEGRINYWTLLPEGDTIIFEANGNLIAPWYAFALSRGFGSG'.
 IITSNASMDECDKACQTPQGAINSSLPFQNVHPVTIGECPKYVRSTKLRMVTGLR'.
 NIPSIQSRGLFGAIAAGFIEGGWTGMIDGWYGYHHQNEQSGYAADQKSTQNAING'.
 ITNKVNSVIEKMNTQFTAVGKEFNKLEKRMENLNKKVDDGFLDIWTYNAELLVLL'.
 ENERTLDFHDSNVKNLYEKVKSQKNNAKEIGNGCFEFYHKCNNECMESVKNGTY'.
 DYPKYSEESKLNREKIDGVKLESMGVYQILAIYSTVASSLVLLVSLGAISFWMCS'.
 NGSLQCRICI",
 "MKTIIAFSCILCLIFAQKLPGSDNSMATLCLGHHAVPNGTLVKTIITDDQIEVTNATELVQS'.
 SSTGRICNSPHQILDGKNCTLIDALLGDPHCDDLQNKEDLDFVERSTAYSNCYPY'.
 YVPDYATLRSLVASSGNLEFTQESFNWTGVAQDGSSYACRRGSVNSFFSRLNWLY'.
 NLNYKYPEQNVTMPNNDKFDKLYIWGVHHPGTDKDQTNLYVQASGRVIVSTKRSQ'.
 QTVIPNIGSRPWVRGVSSIIISYWTIVKPGDILLINSTGNLIAPRGYFKIQSGKS'.
 SIMRSDAHIDECSNSECITPNGSIPNDKPFQNVNKITYGACPRYVKQNTLKLATGM'.

RNVPEKQTRGIFGAIAGFIENGWEGMVDGWYGFRHQNSEGTGQAADLKSTQAAIN'.
 QITGKLN RVIKKTNEKFHQIEKEFSEVEGRIQDLEKYVEDTKIDLWSYNAELLVA'.
 LENQHTIDLTDSEMSKLFERTRRQLRENAEDMGNGCFKIYHKCDNACIGSIRNGT'.
 YDHDYRNEALNNRFQIKGVQLKSGYKD WILWISFAISCFLLCVLLGFIMWACQ'.
 KGNIRCNICI" ,
 "MKTIIALSHILCLVFAQKLP GNDNSTATLCLGHHAVPNGTIVKTIITNDQIEVTNATELVQN'.
 SSIGEICDSPHQILDGENCTLIDALLGDPQCDGFQNKKWDLFVERSKAYSNCYPY'.
 DVPDYASLRSLVASSGTLEFNNE SFNWTGVTQNGTSSACIRRSNNSFFSRLNWL'.
 QLNFKYPALNVTMPNNEQFDKLYIWGVHHPVTDKDQIFLYAQSSGRITVSTKRSQ'.
 QAVIPNIGYRPRIRNIPSRISIIYWTIVKPGDILLINSTGNLIAPRGYFKIRSGKS'.
 SIMRSDAPIGKCNSECITPNGSIPNDKPFQNVNRITYGACPRYVKQSTLKLATGM'.
 RNVPEKQTRGIFGAIAGFIENGWEGMVDGWYGFRHQNSEGRGQAADLKSTQAAID'.
 QINGKLNRLIGKTNEKFHQIEKEFSEVEGRIQDLEKYVEDTKIDLWSYNAELLVA'.
 LENQHTIDLTDSEMKNLFEKTKQLRENAEDMGNGCFKIYHKCDNACIGSIRNGT'.
 YDHDVYRDEALNNRFQIKGVELKSGYKD WILWISFAISCFLLCVALLGFIMWACQ'.
 KGNIRCNICI" ,
 "MKTIIALSYILCLVFAQKLP GNDNSTATLCLGHHAVPNGTIVKTIITNDQIEVTNATELVQS'.
 SSTGEICDSPHQILDGKNCTLIDALLGDPQCDGFQNKKWDLFVERSKAYSNCYPY'.
 DVPDYASLRSLVASSGTLEFNNE SFNWTGVTQNGTSSACIRRSKNSFFSRLNWL'.
 HLNFKYPALNVTMPNNEQFDKLYIWGVHHPGTDKDQIFPYAQASGRITVSTKRSQ'.
 QTAIPNIGSRPRVRNIPSRISIIYWTIVKPGDILLINSTGNLIAPRGYFKIRSGKS'.
 SIMRSDAPIGKCNSECITPNGSIPNDKPFQNVNRITYGACPRYVKQNTLKLATGM'.
 RNVPEKQTRGIFGAIAGFIENGWEGMVDGWYGFRHQNSEGRGQAADLKSTQAAID'.

QINGKLNRLIGKTNKFKHQIEKEFSEVEGRIQDLEKYVEDTKIDLWSYNAELLVA`.
LENQHTIDLTDSEMKNLFKTKKQLRENAEDMGNGCFKIYHKCDNACIGSIRNGT`.
YAHDVYRDEALNNRFQIKGVELKSGYKDILWISFAISCFLLCVALLGFIMWACQ`.
KGNIRCNICI",
"MKTIIALS YILCLVFAQKLP GNDNSTATLCLGHHAVP NGTIVKTI TNDQIEVTNATELVQS`.
SSTGEICDSPHQILDGKNCTLIDALLGDPQCDGFQNKKWDLFVERS KAYSNCYPY`.
DVPDYASLRSLVASSGTLEFN NESFNWTGVTQNGTSSACIRRSKNSFFSRLNWL T`.
HLNFKYPALNVTMPNNEQFDKLYIWGVHHPGTDKDQIFLYAQASGRITVSTKRSQ`.
QTVSPNIGSRPRVRNIPSRISYWTIVKPGDILLINSTGNLIAPRGYFKIRSGKS`.
SIMRSDAPIGKCNSECITPNGSIPNDKPFQNVNRITYGACPRYVKQNTLKLATGM`.
RNVPEKQTRGIFGAIAGFIENGWEGMVDGWYGFRHQNSEGRGQAADLKSTQAAID`.
QINGKLNRLIGKTNKFKHQIEKEFSEVEGRIQDLEKYVEDTKIDLWSYNAELLVA`.
LENQHTIDLTDSEMKNLFKTKKQLRENAEDMGNGCFKIYHKCDNACIGSIRNGT`.
YDHDVYRDEALNNRFQIKGVELKSGYKDILWISFAISCFLLCVALLGFIMWACQ`.
KGNIRCNICI",
"MKTIIALS YILCLVFAQKLP GNDNSTATLCLGHHAVP NGTIVKTI TNDQIEVTNATELVQN`.
SSTGEICDSPHQILDGKNCTLIDALLGDPQCDGFQNKKWDLFVERS KAYSNCYPY`.
DVPDYASLRSLVASSGTLEFN NESFNWTGVTQNGTSSACIRRSKNSFFSRLNWL T`.
NLNFKYPALNVTMPNNEQFDKLYIWGVHHPVTDKDQIFLYAQASGRITVSTKRSQ`.
QTVIPNIGSRPRVRNIPSRISYWTIVKPGDILLINSTGNLIAPRGYFKMQSGKS`.
SIMRSDAPIGKCNSECITPNXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX`.
XNVPEKQTRGIFGAIAGFIENGWEGMVDGWYGFRHQNSEGRGQAADLKSTQAAID`.
QINGKLNRLIGKTNKFKHQIEKEFSEVEGRIQDLEKYVEDTKIDLWSYNAELLVA`.

LENQHTIDLTDSEMKNLFKTKKQLRENAEDMGNGCFKIYHKCDNACIGSIRNGT'.
 YDHNVYRDEALNNRFQIKGVELKSGYKDILWISFAISCFLLCVALLGFIMWACQ'.
 KGNIRCNICI",
 "MEKIVLLFAIVSLVKSDQICIGYHANNSTEQVDTIMEKNVTVTHAQDILEKKHNGKLCDL'D'.
 GVKPLILRDCSVAGWLLGNPMCEFINVPEWSYIVEKANPVNDLCYPGDFNDYEE'.
 LKHLLSRINHFEKIQIIPKSSWSSEASLGVSSACPYQGKSSFFRNVVWLIKNS'.
 TYPTIKRSYNNTNQEDLLVLWGIHHPNDAAEQTKLYQNPTTYISVGTSTLNQRLV'.
 PRIATRSKVNGQSGRMEFFWTILKPNDAINFESNGNFIAPYAYKIVKKG DSTIM'.
 KSELEYGNCNTKCQTPMGAINSSMPFHNHPLTIGECPKYVKS NRLVLATGLRNS'.
 PQRRRRKKRGLFGAIAGFIEGGWQGMVDGWYGYHHSNEQSGYAADKESTQKAI'.
 DGVTNKVNSIIDKMNTQFEAVGREFNLERRIENLNKKMEDGFLDVWTYNAELLV'.
 LMENERTLDFHDSNVKNLYDKVRLQLRDNAKELGNGCFEFYHKCDNECMESVRNG'.
 TYDYPQYSEEARLKREEISGVKLESIGIYQILSIYSTVASSLALAIMVAGLSLWM'.
 CSNGSLQCRICI",
 "MEKIVLLLAIIVSLVKSDQICIGYHANNSTEQVDTIMEKNVTVTHAQDILEKTHNGKLCDL'D'.
 GVKPLILRDCSVAGWLLGNPMCEFLNVPEWSYIVEKINPANDLCYPGNFNDYEE'.
 LKHLLSRINHFEKIQIIPKSSWDHEASSGVSSACPYQGRSSFFRNVVWLIKND'.
 AYPTIKRSYNNTNQEDLLVLWGIHHPNDAAEQTRLYQNPTTYISVGTSTLNQRLV'.
 PKIATRSKVNGQSGRMEFFWTILKPNDAINFESNGNFIAPENAYKIVKKG DSTIM'.
 KSELEYGNCNTKCQTPIGAINSSMPFHNHPLTIGECPKYVKS NRLVLATGLRNS'.
 PQGERRRKKRGLFGAIAGFIEGGWQGMVDGWYGYHHSNEQSGYAADKESTQKAI'.
 DGVTNKVNSIIDKMNTQFEAVGREFNLERRIENLNKKMEDGFLDVWTYNAELLV'.
 LMENERTLDFHDSNVKNLYDKVRLQLRDNAKELGNGCFEFYHRCNECMESVRNG'.

TYDYPQYSEEARLKREEISGVKLESIGTYQILSIYSTVASSLALAIMVAGLSLWM'.
 CSNGSLQCRICI",
 "MEKIVLLLAIVSLVKSQICIGYHANNSTEQVDTIMEKNVTVTHAQDILEKTHNGKLCDLN'.
 GVKPLILRDCSVAGWLLGNPMCDEFINVPEWSYIVEKANPANDLCYPGNFNDYEE'.
 LKHLLSRINHFEKIQIIPKSSWSDHEASSGVSSACPYQGTPSFERNVWLIIKKN'.
 TYPTIKRSYNNNTNQEDLLILWGIHHSNDAAEQTKLYQNPTTYISVGTSTLNQRLV'.
 PKIATRISKVNGQSGRMDFFWTILKPNDAINFESNGNFIAPYAYKIVKKGDSAIV'.
 KSEVEYGNCNTKCQTPIGAINSSMPFHNIHPLTIGECPKYVKS NKLVLATGLRNS'.
 PLRERRRRKGLFGAIAGFIEGGWQGMVDGWYGYHHSNEQSGGYAADKESTQKAID'.
 GVTNKVNSIIDKMNTQFEAVGREFNLERRIENLNKKMEDGFLDVWTYNAELLV'.
 MENERTLDFHDSNVKNLYDKVRLQLRDNAKELGNGCFEYHKCDNECMESVRNGT'.
 YDYPQYSEEARLKREEISGVKLESIGTYQILSIYSTVASSLALAIMVAGLSLWMC'.
 SNGSLQCRICI",
 "MEKIVLLFATISLVKSDHICIGYHANNSTEQVDTIMEKNVTVTHAQDILEKTHNGKLCDLN'.
 GVKPLILKDCSVAGWLLGNPMCDEFINVPEWSYIVEKANPANDLCYPGNFNDYEE'.
 LKHLLSRINHFEKIQIIPKNSWSDHEASLGVSAAACPYQGKSSFFERNVWLIIKKN'.
 AYPTIKKGYNNNTNQEDLLVLWGIHHPNDEAEQTRLYQNPTTYISIGTSTLNQRLV'.
 PKIATRISKINGQSGRIDFFWTILKPNDAIHFEESNGNFIAPYAYKIVKKGDSITIM'.
 KSEVEYGNCNTRCQTPIGAINSSMPFHNIHPLTIGECPKYVKS NKLVLATGLRNS'.
 PQRERRRRKGLFGAIAGFIEGGWQGMVDGWYGYHHSNEQSGGYAADKESTQKAID'.
 GVTNKVNSIIDKMNTQFEAVGREFNLERRIENLNKKMEDGFLDVWTYNAELLV'.
 MENERTLDFHDSNVNRNLYDKVRLQLKDNAKELGNGCFEYHKCDNECMESVRNGT'.
 YDYPQYSEEARLKREEISGVKLESIGIYQILSIYSTVASSLVLAIMMAGLSLWMC'.

```

SNGSLQCRICI"};

In[148]:= ScrollTable [data_, xlabs_, ylabs_, size_: {500, 300},
  sbars_: {True, True}] :=
  Column[
    {Pane[Style[TableForm[data, TableAlignments → {Right,
Bottom},
      TableHeadings → {ylabs, Rotate[#, -Pi/2] & /
@xlabs}],
      8, FontFamily → "Arial", Bold], size, Scrollbars
→ sbars,
      ImageSizeAction → "Scrollable"]}, Alignment →
{Left, Baseline}]

TableFigure [data_, xlabs_, ylabs_, size_: {500, 300}] :=
  Column[
    {Style[TableForm[data, TableAlignments → {Right,
Bottom},
      TableHeadings → {ylabs, xlabs}], size, Bold,
      FontFamily → "Arial"]}, Alignment → Left]

In[150]:= ScrollMatrix[data_, size_: {500, 300},
  mtxStyle_: {FontFamily → "Arial", 9}] :=
  Pane[Style[MatrixForm[data], mtxStyle], ImageSize →
size,
  Scrollbars → True, ImageSizeAction → "Scrollable"]

In[151]:= colorbarL[{min_, max_}, colorFunction_: Automatic, divs_: 150,
  title_: "", [size_: {200, 80}] :=
  DensityPlot [x, {x, min, max}, {y, 0, 0.1}, AspectRatio
→ .1,
  PlotRangePadding → 0, PlotPoints → {divs, 2},
  MaxRecursion → 0,
  Frame → True, FrameLabel → {{None, None}, {title,
None}},
  ImageSize → size,
  LabelStyle → Directive[FontFamily → "Arial", 10,
  Bold],
  FrameTicks → {{None, None}, {All, None}},
  FrameTicksStyle → Directive[FontFamily → "Arial", 9,
  Plain],
  ColorFunction → colorFunction]

```

4. Proximity Matrices

Classical and metric MDS methods rely on a symmetric input matrix M (referred to as a proximity matrix) that specifies the N -dimensional distance between pairs of objects a and b , denoted δ_{ab}^N . Proximity may be defined as a distance or dissimilarity.

In some applications, M is composed of distances δ_{ab}^N that are specified directly, for example, the flying distances between airport pairs in the United States, as shown in Table 1. The proximity matrix is a symmetric distance matrix with the airport names in the first column and the corresponding airport abbreviations for the column labels.

4.1. Constructing a Proximity Matrix from a Cases by Attributes Table

Often, M is not the primary data but must be calculated from a set of cases, each of which is described by a vector of attributes, with the primary data specified in an $m \times n$ cases by attributes data matrix. The attributes can be binary membership in a category or group, physical properties, or string sequences. The methods used to calculate δ_{ab}^N vary by the data and the application. The most commonly used metric is the Euclidean distance for continuous variables, but other distance functions may also be used, such as the Hamming and Manhattan distances, Boolean distances for Boolean data, Smith–Waterman string sequence similarity, or color distance functions for images.

Table 2 is an example of such a cases by attributes matrix. The cases (rows) represent the influenza virus surface hemagglutinin proteins from various influenza strains (H1, H3, H5) and sub-strains (e.g. A/California/04/2009). The data is taken from an antibody binding assay, with the values reflecting the amount of antibody binding to each influenza hemagglutinin protein. The antibodies were isolated from the serum of ferrets infected with an influenza virus strain, shown in the columns. Each case (row) is characterized by a 15-dimensional attribute vector, with each attribute corresponding to the serum antibody binding values against that particular influenza strain hemagglutinin from a biological assay. Higher values indicate more antibody binding. Such assays are used to select vaccines each year against strains predicted to circulate in the fall. Note that there are 15 different cases characterized by 14 sera.

Before performing the MDS calculations, it is important to rescale each attribute to span the same range across all the cases, especially if the attributes have different units or variable ranges. This rescaling does not assume any specific statistical distribution. It is designed so that each attribute has equal weight in the proximity calculation, although weighted proximity matrices can also be used if the application requires this. Here we rescale the results for the above attributes so that the range of each attribute column (antibody binding test result for a single hemagglutinin) has a minimum of 0 and a maximum of 1000 (Table 3).

For multidimensional scaling, this data is transformed into a proximity matrix using a EuclideanDistance calculation. The larger the proximity measurement, the greater the difference between the hemagglutinin proteins from the two viral infection strains, as specified by the reactivity of immune sera against the probe hemagglutinin proteins. Table 4

uses the `DistanceMatrix` function to create a proximity matrix for the influenza. There are many ways to calculate distance matrices, and custom functions are often used. One convenient option is to use the `DistanceMatrix` function, which allows access to a number of standardized distance metrics (e.g. `EuclideanDistance`, `ManhattanDistance`, `CorrelationDistance`, etc.), which can be directly set by the option `DistanceFunction`. The `DistanceMatrix` function is contained in the `HierarchicalClustering` package.

4.2. Visualization

It is often helpful to visualize a proximity matrix as a heat map using the `ArrayPlot` function. In Figure 1 the most similar hemagglutinin protein pairs are blue and the most different are red. As can be seen, this representation of the data suggests a similarity of influenza virus hemagglutinin proteins from similar strains (H1, H3, H5) and the occurrence of year-to-year differences, especially for the H1 strains.

4.3. Proximity Matrices from Molecular Sequence Comparisons

Another common application of multidimensional scaling is to compare protein or DNA sequences. Such sequences are essentially linear lists of strings that specify the protein or nucleic acid composition. Sequence differences are used to estimate molecular evolution, how the immune system may respond to a new virus after immunization, and cross-species comparison of molecules.

To construct a protein sequence proximity matrix, we begin with the amino acid sequences of the hemagglutinin proteins from influenza strains specified in the proximity matrix of the previous section. These were retrieved from the UniProt database (www.uniprot.org). Each letter in the sequence represents an amino acid, and each sequence is approximately 566 amino acids long.

To calculate the proximity matrix, we use a rescaled inverse of the Smith–Waterman similarity calculation, with the PAM70 similarity rules for amino acid comparison. Sequence similarity calculations have larger values with greater similarity, and the minimum score is generally the length of the sequence. Because MDS methods use proximity measures that increase monotonically with dissimilarity, the inverse of the sequence similarity function is used. Thus, for clarity in this example, we have transformed the result by subtracting a constant (528), which is one less than the minimum sequence length, and multiplying the inverse by a scale factor of 10^6 . Note that nonzero values do not occur (Table 5).

4.4. Synopsis: Proximity Matrices

The first step in multidimensional scaling is to create a proximity matrix. The value of each matrix element is a distance measure; the greater the difference between the two elements being compared, the greater the value of the metric. We have demonstrated three different cases of proximity matrix construction, including: (i) the airport distance case, where the proximities are known *a priori* and do not need to be calculated; (ii) the influenza protein antibody reactivity case, with proximity calculation from a cases by attributes matrix; and (iii) the influenza protein sequence example, with calculation from a sequence similarity

measure. Heat maps are useful for displaying proximity matrices, giving a visual view of similarities. All the resulting proximity matrices can be used for MDS calculations, as will be discussed next.

5. Classical Multidimensional Scaling

Here we briefly describe the mathematics of classical multidimensional scaling, which uses a linear coordinate transformation calculation to reduce case coordinates from N dimensions down to two or three dimensions. The approach begins with calculating a proximity matrix. The orientation of the new coordinate system is such that it accounts for the greatest amount of variance between cases, as defined by attributes, in the reduced number of dimensions. The new coordinates can be used to plot the relative locations of the data points and help visualize differences between data points and clusters of similar cases. The proximity matrix can also be used for data clustering and heat mapping, which provide alternate visualizations of case relationships.

5.1. Mathematics

Let M be a $c \times v$ matrix, where c is the number of individual cases (e.g. data points) and v is the number of measured variables or attributes (e.g. financial variables, consumer product attributes, antibody binding measurements, etc.), which are used to discriminate between each case z_i , $i = 1, 2, \dots, c$.

If the data attributes are of different scales and units, they are generally rescaled with a range of $[0, 1]$. This assures that each attribute contributes equally to the proximity measure.

Alternatively, M may remain in the original units if each attribute has the same range and units, or if a weighted proximity matrix is desired. Next, let the proximity matrix D of M or M rescaled be a matrix of dimensions $c \times c$ with elements

$$d_{ij} = \|x_i - x_j\|, \quad i, j = 1, 2, \dots, c. \quad (1)$$

One can use a variety of applicable proximity measures (e.g. Euclidean, Mahalanobis, etc.) to create the proximity matrix. If the proximity measure is a Euclidean distance, then classical MDS is equivalent to principal component analysis (PCA). If any other distance function is used, then classical MDS will result in a different transformation that is not equivalent to PCA.

Let I be the identity matrix of dimensions $c \times c$ and K be the constant matrix of dimensions $c \times c$, where $K_{ij} = 1$. Let P be the matrix of the squared proximities:

$$P = \|x_i - x_j\|^2, \quad i, j = 1, 2, \dots, c. \quad (2)$$

The kernel matrix B is derived by first calculating the matrix J from the identity matrix and the number of variables v :

$$J = I - \nu^{-1} K. \quad (3)$$

J is the matrix that double-centers the input matrix by subtracting the row and column means from every element of the matrix. Introducing the constant $-\frac{1}{2}$ creates an equivalence to PCA if d is a Euclidean distance function:

$$B = -\frac{1}{2} J \cdot P \cdot J. \quad (4)$$

The eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_v$ and eigenvectors $e_1, e_2, \dots, e_v$ of B are calculated. To reduce to dimension m with $m < v$, only the first m of a ranked list of the λ and e are selected (e.g. $m = 2$ for a two-dimensional mapping).

The diagonal matrix of the square roots of the eigenvalues Λ_m is

$$\Lambda_m = \begin{pmatrix} \lambda_1 & 0 & & 0 \\ 0 & \lambda_2 & & 0 \\ & & \dots & \\ 0 & 0 & 0 & \lambda_m \end{pmatrix}. \quad (5)$$

The matrix X of relative coordinates in m dimensions is then calculated by the product of the matrix of eigenvectors E_m and the diagonal matrix of the square roots of the eigenvalues Λ_m :

$$X = E_m \cdot \Lambda_m^{1/2}. \quad (6)$$

5.2. Case Study: Airport Location Mapping

We begin with one of the original motivating problems related to MDS, reconstructing the relative locations between cities in two dimensions, knowing only the relative distances between each one. We use the United States airport distance data from Section 3 and reconstruct the cities' relative geographic positions. We begin by squaring the distance functions and calculating the scale of the dataset.

```
In[167]:= Am = airData2;
          scale = Length[Am];
```

Next, we create an identity matrix, with all elements being 0 except the diagonal elements, which are 1, and a constant matrix with all elements equal to 1. Both matrices are of the same dimensions and are used in subsequent calculations to transform data from 28 down to two dimensions.

```
In[169]:= Idm = IdentityMatrix[Length[Am]];
          oneM = ConstantArray[1, Dimensions[Am]];
```

We then calculate the kernel matrix and double-center the new N -dimensional coordinates.

```
In[171]:= J = Idm - 1. / scale oneM;
          Bm = -1 / 2 J.Am.J;
```

With the number of dimensions set to 2, the matrix eigenvectors and eigenvalues are calculated. The sign of the first eigenvector is adjusted to lay out the cities from west to east. (*Mathematica* does not guarantee the direction of an eigenvector.)

```
In[173]:= dim = 2;
          Em = Eigenvectors[Bm, dim] If[$VersionNumber < 10.2, 1,
{-1, 1}]

Out[174]= {{0.129171, -0.164791, 0.101168, -0.0843364, -0.248814, 0.237447,
0.155102, 0.0613687, 0.128977, -0.0244763, -0.122908,
-0.000635832, 0.109603, -0.141688, 0.00345098,
0.0865066,
-0.00785255, 0.0313444, -0.291272, 0.0981005,
0.0577967,
0.210212, -0.00330048, 0.0674643, 0.218743, -0.0249825,
0.207308, -0.212435, 0.150431, -0.310908, 0.187052,
0.0432586, -0.205778, -0.33099, -0.302226, 0.192888},
{-0.138278, 0.17658, -0.154413, 0.210242, 0.117269,
0.230394,
0.184481, 0.0997557, 0.116632, -0.218811, -0.0125633,
0.0620399, 0.131882, -0.267718, -0.314541, 0.0380486,
-0.0115239, -0.144878, -0.182929, -0.00641343,
-0.125285,
-0.349034, 0.168374, -0.282816, 0.154349, 0.0477733,
0.118949, -0.213395, 0.0993995, 0.203908, -0.0304331,
-0.0120272, 0.0201888, -0.0508981, 0.262057,
0.0736327}}
```

```
In[175]:= Ev = Eigenvalues[Bm, dim]
```

```
Out[175]= {2.14663 × 107, 4.77758 × 106}
```

The transformation matrix dMTX is calculated from a diagonal matrix formed by the eigenvalues Ev, and the coordinate matrix EmM is calculated from the eigenvectors Em.

```
In[176]:= dMTX = DiagonalMatrix[Ev];
          EmM = Transpose[Em];
```

```

In[178]:= Column[{Style["Transformation Matrix", FontFamily → "Arial",
Bold],

dMTX / / MatrixForm, " ",

Style["Coordinate Matrix", FontFamily → "Arial",
Bold],

EmM / / MatrixForm}, Alignment → Left]

```

Transformation Matrix

$$\begin{pmatrix} 2.14663 \times 10^7 & 0. \\ 0. & 4.77758 \times 10^6 \end{pmatrix}$$

Coordinate Matrix

$$\begin{pmatrix} 0.129171 & -0.138278 \\ -0.164791 & 0.17658 \\ 0.101168 & -0.154413 \\ -0.0843364 & 0.210242 \\ -0.248814 & 0.117269 \\ 0.237447 & 0.230394 \\ 0.155102 & 0.184481 \\ 0.0613687 & 0.0997557 \\ 0.128977 & 0.116632 \\ -0.0244763 & -0.218811 \\ -0.122908 & -0.0125633 \\ -0.000635832 & 0.0620399 \\ 0.109603 & 0.131882 \\ -0.141688 & -0.267718 \\ 0.00345098 & -0.314541 \\ 0.0865066 & 0.0380486 \\ -0.00785255 & -0.0115239 \\ 0.0313444 & -0.144878 \\ -0.291272 & -0.182929 \\ 0.0981005 & -0.00641343 \\ 0.0577967 & -0.125285 \\ 0.210212 & -0.349034 \\ -0.00330048 & 0.168374 \\ 0.0674643 & -0.282816 \\ 0.218743 & 0.154349 \\ -0.0249825 & 0.0477733 \\ 0.207308 & 0.118949 \\ -0.212435 & -0.213395 \\ 0.150431 & 0.0993995 \\ -0.310908 & 0.203908 \\ 0.187052 & -0.0304331 \\ 0.0432586 & -0.0120272 \\ -0.205778 & 0.0201888 \\ -0.33099 & -0.0508981 \\ -0.302226 & 0.262057 \\ 0.192888 & 0.0736327 \end{pmatrix}$$

Out[178]=

The graph coordinates cMDScoord are calculated and projected to two dimensions.

```

In[179]:= cMDScoord=EmM.  $\sqrt{\text{dMTX}}$ 

```

```

Out[179]= {{598.473, -302.244}, {-763.504, 385.964}, {468.731, -337.511},
           {-390.745, 459.541}, {-1152.8, 256.323}, {1100.13,
           503.588},
           {718.612, 403.233}, {284.332, 218.043}, {597.573,
           254.93},
           {-113.403, -478.27}, {-569.455, -27.4605}, {-2.94592,
           135.605},
           {507.808, 288.264}, {-656.464, -585.169}, {15.989,
           -687.514},
           {400.8, 83.1654}, {-36.3822, -25.1885}, {145.224,
           -316.669},
           {-1349.51, -399.841}, {454.517, -14.0183}, {267.782,
           -273.843},
           {973.947, -762.908}, {-15.2917, 368.028}, {312.574,
           -618.169},
           {1013.47, 337.371}, {-115.748, 104.421}, {960.493,
           259.995},
           {-984.249, -466.433}, {696.974, 217.264}, {-1440.49,
           445.696},
           {866.642, -66.5197}, {200.425, -26.2887}, {-953.404,
           44.128},
           {-1533.53, -111.251}, {-1400.26, 572.796}, {893.684,
           160.944}}

```

Finally, we plot the two-dimensional airport locations in cMDScoord, which shows the airports in the new coordinate system (Figure 2).

Classical multidimensional scaling gives a solution that has an excellent projection of the cases in a lower dimension, preserving the between-cases distances from N dimensions. It is important to note, however, that the solution is not unique with respect to the rotation and reflection of the visualization. It is often the case when one knows the “true” orientation of the case locations (e.g. the previous map example), that classical (and metric) MDS methods yield a visualization that is reflected or rotated. In these cases, use of other boundary conditions (fixing known coordinates) helps constrain the orientation of the solution. Often, however, such boundary conditions are not available.

Given the above solution, how well does classical MDS reproduce the airport coordinates? Because we have a gold standard, the actual airport locations, comparison of the coordinate estimates is possible. However, the airport locations are represented in a geographic coordinate system, making a comparison of calculated versus actual airport locations a bit more complicated. One approximate way to approach this issue is by first examining the correlation between the actual airport locations, in longitude and latitude coordinates, and the derived coordinates from the classical MDS calculation, using the

PearsonCorrelationTest. The units are different, which we will address in a moment, but this does not affect the Pearson correlation test (Figure 3).

While the correlations are high, they are not perfect, as we can see by the off-diagonal locations of some of the points. To better visualize these differences, we transform the classical MDS coordinates into longitude and latitude units. This is a transformation that uses the pseudoinverse matrix to translate and rotate to find a good alignment between the two coordinate sets.

```
In[184]:= newcoords = Map[# + {c, d} &, TrcMDSCoordinates.{{a, b}, {-b, a}}];
          vec, mat}= CoefficientArrays[Flatten[newcoords], {a, b, c,
          d}];

          res =
            Thread[{a, b, c, d} - >
PseudoInverse[mat].Flatten[geoCoordsAir]]

Out[186]= {a -> 0.0179315, b -> -0.00149652, c -> -94.1752, d -> 38.6731}
```

This type of transformation preserves the relative distances between the classical MDS derived city locations and thus provides a relatively accurate estimate of how close the estimates are to the actual city locations. A more accurate solution would involve mapping of the coordinates to an appropriate spherical projection system; however, this is outside the focus of this article.

The transformation coefficients can now be used to transform the classical MDS derived coordinates into GeoPosition values. To compare the actual (●) with the cMDS calculated (♦) airport locations, we use the GeoListPlot function, which plots both sets of points on a map of North America (Figure 4).

```
In[187]:= transCMDS = newcoords / . res;
          geoMMDS = Map[GeoPosition, Map[Reverse, transCMDS]];
```

The accuracy of the classical MDS estimates is fairly good, but not perfect. Part of this error is due to the approximate transformation from arbitrary MDS coordinates to geographic coordinates, and the complex projection to a flat map.

5.3. Function: Classical Multidimensional Scaling

The preceding code can be summarized in a succinct *Mathematica* function for classical MDS calculation. This function takes a proximity matrix as input and returns coordinates mapped to a space of the specified number of dimensions. The default number of dimensions is two, but the function also works with three or more dimensions.

```
In[190]:= classicalMDS[Dm_, dim_: 2] :=
```

```

Module [{Idm, oneM, A, scale, J, Bm, Em, DmS, Ev, dMTX,
EmM,
coordX}, {
DmS = Dm2;
Idm = IdentityMatrix[Length[Dm]];
oneM = ConstantArray[1, Dimensions[Dm]];
scale = Length[Dm];
J = Idm - 1. / scale oneM;
Bm = -0.5 J.DmS.J;
Em = Eigenvectors[Bm, dim];
Ev = Eigenvalues [Bm, dim];
dMTX = DiagonalMatrix[Ev];
EmM = Transpose[Em];
coordX = EmM. dMTX}] [[1]]

```

5.4. Classical Multidimensional Scaling and the Singular Value Decomposition

It is important to note that if the proximity matrix used for classical MDS is formed from a cases by attributes matrix using a Euclidean distance, then classical MDS is equivalent to principal component analysis. Principal component analysis is a specific case of the more general singular value decomposition method (SVD). Details of the SVD mathematics and computational method is not discussed in detail here, but rather we will outline the computational steps necessary to use the method with the function `SingularValueDecomposition`.

As with classical MDS, a transformation matrix is calculated, and the data is centered to a mean of zero.

```

In[191]:= len = Length[airData];
hmat = IdentityMatrix[len] - 1 / len;
zeromeanData = -hmat.N[airData^2].hmat / 2;

```

The SVD function is then applied to the normalized data and yields three matrices. Here `ww` is the diagonal matrix of eigenvalues and `vv` is the coordinate matrix.

```

In[194]:= dimensions = 2;
{uu, ww, vv} = SingularValueDecomposition[zeromeanData,
dimensions];

```

Next, the two-dimensional projection of the coordinates is calculated.

```

In[196]:= mMDScoordinatesNew = vv.Sqrt[ww];

```

When the dataset to be analyzed is very large, then the computational efficiency degrades quickly if the proximity matrix must be directly calculated. In this case, a matrix transformation can be applied to avoid a cell-by-cell proximity matrix calculation; this greatly increases computational efficiency. The computational efficiency for both classical and metric MDS is discussed, with a direct comparison, in Section 7.

5.5. Synopsis: Classical MDS

Classical MDS is a data mining and data exploration method allowing dimensional reduction to be used to highlight possible clusters, similarities, or differences between cases described by high-dimensional attribute vectors. It is a starting point for more rigorous statistical analysis and hypothesis testing. When the input proximity matrix is composed of Euclidean distances, classical MDS is equivalent to both principal coordinate analysis and singular value decomposition. Computational efficiency and possible limitations of classical MDS methods are discussed in Section 7.

6. MDS Methods: Metric Multidimensional Scaling

Metric multidimensional scaling is a second type of MDS. While classical MDS relies on matrix algebra, metric MDS involves computational minimization of the difference between N -dimensional and two-dimensional proximity measures (e.g. Euclidean distance) of the case coordinates in each dimensional system. In essence, the method attempts to minimize the total error between actual and reduced intercase distances for the group of cases as a whole. Metric MDS is also flexible, accommodating different types of proximity metrics (e.g. not Euclidean), as well as different stress functions (nonlinear transformation and distance metrics other than Euclidean) for minimization. This permits nonlinear dimensional reduction, where the projection of data from an N -dimensional space to lower dimensions can be done using a nonlinear transformation. The end result is a dimensional reduction that allows for data visualization.

As in the previous section, we first describe the mathematics with some references to implementation, then demonstrate a practical example.

6.1. Mathematics

Metric MDS also begins with a $c \times c$ proximity matrix D , which represents the original N -dimensional proximity measure for the cases z_i , where $i=1, \dots, c$. This proximity matrix can be given *de novo*, as in distances between airports, or can be calculated from a cases by attributes matrix as described previously,

$$D = \begin{pmatrix} \delta_{11} & \delta_{21} & & \delta_{1c} \\ \delta_{21} & \delta_{22} & & \delta_{2c} \\ & & \dots & \\ \delta_{c1} & \delta_{c2} & & \delta_{cc} \end{pmatrix}. \quad (7)$$

Initial values for the case coordinates, used to seed the minimization algorithm, can be generated at random or calculated using values from the classical MDS algorithm. So we

next set up the initial coordinate matrix X for a two-dimensional projection, for the minimization algorithm,

$$X = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_c & y_c \end{pmatrix}, \quad (8)$$

where $z_i = (x_i, y_i)$. Next, we create a matrix of variables representing the coordinates that the minimization algorithm estimates:

$$E = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_c & b_c \end{pmatrix}. \quad (9)$$

We then calculate a matrix of distance functions for E using Euclidean distance between any two elements (a_i, b_i) and (a_j, b_j) ,

$$d_{ij} = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}. \quad (10)$$

The distance matrix D is now given by

$$D = \begin{pmatrix} d_{11} & d_{21} & & d_{1c} \\ d_{21} & d_{22} & & d_{2c} \\ & & \dots & \\ d_{c1} & d_{c2} & & d_{cc} \end{pmatrix}. \quad (11)$$

For the minimization, several different types of stress functions may be used. Here we select the stress function to be the sum of the squares of the errors (SSE):

$$E(D_m, \Delta_m) = \sum_{i=0}^c \sum_{j=0}^c (d_{ij} - \delta_{ij})^2. \quad (12)$$

We use the efficient and robust “Differential Evolution” method, with sufficient search points to avoid local minima. Some aspects of the choice of minimization methods are discussed later.

6.2. Case Study: Mapping Antibody Reactivity Differences between Influenza Strains

This example is motivated by the need to determine if an influenza strain is sufficiently different, as determined by the immune system, so that even if a person has prior immunity from vaccination against other similar strains, it will be insufficient to protect against the new strain. Influenza vaccination relies on the immune system’s ability to generate

protective antibodies, which are proteins that bind to the virus and block its ability to infect cells in the respiratory system. The major protection comes from antibodies that bind the viral surface protein hemagglutinin, the protein responsible for attaching the virus to the target cells. Variations in the hemagglutinin protein structure between influenza strains allow the viruses to evade the immune system and cause an infection. This is why the influenza vaccine composition is changed every year. Figuring out which strains to include in the influenza vaccine is an annual problem for the international organizations that recommend changes in the seasonal influenza vaccine.

Metric MDS provides a graphical way of visualizing influenza strain similarity, derived from experiments measuring the ability of serum from animals infected with influenza to bind to the target virus hemagglutinin. There has been extensive literature over the last decade on the use of metric MDS for this purpose, referred to as antigenic cartography [8–11]. Metric MDS was chosen for dimensional reduction and visualization by several groups, as classical MDS methods could not be easily adapted to solve several issues [8, 9]. These included the need for complete datasets [10, 11], where in some cases data was missing due to experimental considerations. Metric MDS methods could be adapted to impute relationships [10]. In addition, metric MDS was viewed as a more accurate estimator of influenza strain antigenic distance due to correlations with the binary logarithm of the hemagglutinin inhibition assay serum titers [8, 9]. Use of metric MDS continues in the influenza literature [10, 11], although newer methods of measuring antibody reactivity do not have the same issues as older assays, and classical MDS could be used to the same end. To illustrate the method of metric MDS, however, we will use it in this example. We discuss method selection in Section 7, as well as considerations of computational efficiency.

We begin with the proximity matrix derived in the previous section from the antibody reactivity to influenza hemagglutinin data. We next calculate the relative positions of each influenza strain with respect to the entire set by minimizing a stress function. The Array function is used to create an array of variables that specify the coordinates for each case as a 2D set of points.

```
In[197]:= dimensions =2;
          Y =Array[sr, {lenDm =Length[Dm], dimensions}]

Out[198]= {{sr[1, 1], sr[1, 2]}, {sr[2, 1], sr[2, 2]}, {sr[3, 1], sr[3, 2]},
           {sr[4, 1], sr[4, 2]}, {sr[5, 1], sr[5, 2]}, {sr[6, 1],
sr[6, 2]},
           {sr[7, 1], sr[7, 2]}, {sr[8, 1], sr[8, 2]}, {sr[9, 1],
sr[9, 2]},
           {sr[10, 1], sr[10, 2]}, {sr[11, 1], sr[11, 2]},
           {sr[12, 1], sr[12, 2]}, {sr[13, 1], sr[13, 2]},
           {sr[14, 1], sr[14, 2]}, {sr[15, 1], sr[15, 2]}}
```

Each sr variable pair will hold a pair of coordinates in two-dimensional space. Finally, we flatten the list of variable pairs to input into the function NMinimize.

```
In[199]:= minVar =Flatten[Y]
```

```
Out[199]= {sr[1, 1], sr[1, 2], sr[2, 1], sr[2, 2], sr[3, 1], sr[3, 2],
           sr[4, 1], sr[4, 2], sr[5, 1], sr[5, 2], sr[6, 1], sr[6,
           2],
           sr[7, 1], sr[7, 2], sr[8, 1], sr[8, 2], sr[9, 1], sr[9,
           2],
           sr[10, 1], sr[10, 2], sr[11, 1], sr[11, 2], sr[12, 1],
           sr[12, 2],
           sr[13, 1], sr[13, 2], sr[14, 1], sr[14, 2], sr[15, 1],
           sr[15, 2]}
```

The next section of code creates a series of Euclidean distance calculations for the distances between all combinations of case locations, using the estimated coordinates for each point. For succinctness, only one function within the full matrix is displayed.

```
In[200]:= mY = Table[Sqrt[(Y[[i, 1]] - Y[[j, 1]]) ^2 + (Y[[i, 2]] - Y[[j,
           2]]) ^2],
           {i, lenDm}, {j, lenDm}];
```

```
mY[[4, 3]]
```

```
Out[201]=  $\sqrt{(-sr[3, 1] + sr[4, 1])^2 + (-sr[3, 2] + sr[4, 2])^2}$ 
```

The stress function is a very large least-squares expression.

```
In[202]:= stress =Total[Table[(Dm[[i, j]] - mY[[i, j]]) ^2, {i, lenDm},
           {j, lenDm}], 2];
```

The first nonzero element of the stress function is shown here. The reader may expand the formula to see the entire function, if desired.

```
In[203]:= stress[[2]]
```

```
Out[203]=  $\left(185.642 - \sqrt{(sr[1, 1] - sr[2, 1])^2 + (sr[1, 2] - sr[2, 2])^2}\right)^2$ 
```

Finally, we must choose the method for minimizing the stress function. We discuss two options, FindMinimum and NMinimize. A known issue with metric MDS is the existence of local minima, and thus identifying a global minimum cannot be guaranteed [7]. While FindMinimum may be computationally more efficient, it also lacks the ability to specify the

number of search points and is prone to finding local minima instead of global minima. These issues are discussed in some detail in Section 7. In contrast, the NMinimize function allows specification of “SearchPoints” to address this issue and may be substantially more robust. For these reasons, we chose NMinimize to optimize the stress function.

```
In[204]:= minSol =NMinimize[stress, minVar,
               Method -> {"DifferentialEvolution", "SearchPoints" ->
30}];

               mMDScoordinates =Y / . minSol[[2]]

Out[205]= {{-1402.38, 607.16}, {-1266.42, 469.947}, {-1606.71, 264.926},
               {-1126.87, -53.3209}, {-1765.09, -577.1}, {-1071.78,
-647.575},
               {951.956, -251.19}, {834.507, -1216.65}, {513.712,
-1011.4},
               {349.17, -1063.04}, {395.113, -1555.98}, {0.554891,
1164.23},
               {-439.06, 135.563}, {-135.863, 672.142}, {-369.433,
984.485}}
```

The viral strains are then plotted in the two-dimensional space and are color-coded for clarity (H1, H3, and H5). Note that the coordinate system is arbitrary, in the sense that what is preserved and important are the relative distances between the data points. Thus, we have omitted the axes, which have the same scale in each dimension (Figure 5).

Examining the plot, it is immediately apparent that the influenza strains fall into four distinct clusters. Two of these correspond to major hemagglutinin protein strains (H3, H5). In addition, we can see differences between temporally distinct strains in the H1 influenza strains, giving perhaps two clusters. For example, note the antigenic distance between the pandemic A/California/07/2009 (CAL709), A/Utah/20/2009 (UT2009), and A/Mexico/41/2009 (MEX4109) strains and the other H1 influenza strains. This likely reflects molecular mutations in the hemagglutinin proteins for the 2009 strains. These mutations resulted in decreased binding of antibodies from ferrets infected with earlier influenza strains. This finding was consistent with the decreased population immunity observed in humans to A/California/07/2009 and demonstrates the pandemic nature of that particular influenza strain.

Given that the influenza strains seem to cluster together, what is the relationship between MDS and methods of unsupervised data clustering used in data mining? The answer lies in the use of the proximity matrix, which is used by both hierarchical and agglomerative clustering methods to determine relatedness. This relationship was noted by the developers of the MDS methods [1, 2, 8]. To briefly demonstrate, we apply hierarchical clustering to the same proximity matrix D_m , using the DirectAgglomerate function. Ward's minimum variance method is used for determining cluster linkage.

```

In[207]:= virusLabelsColor =
      Table[Style[seraLabShort[[i]], seraColor[[i]], Bold,
              FontFamily -> "Arial", 9], {i, 1,
Length[seraLabShort]}}];
      clustTR =DirectAgglomerate[Dm, virusLabelsColor, Linkage ->
"Ward"];

```

The resulting dendrogram is displayed in Figure 6. Note the grouping of the different virus types (H1, H3, and H5) based on the reactivity of ferret serum after infection with a single virus and the resulting antibodies against the hemagglutinin proteins.

To explore which viruses are antigenically closer to each other, a heat map of the relative distances from the calculated coordinates (or the original proximity matrix) can then be created with DistanceMatrix and ArrayPlot. Note that more antigenically similar influenza strains have smaller distances between each other (Figure 7).

```

In[210]:= DmC = DistanceMatrix[mMDScoordinates];

```

6.3. Function: Metric Multidimensional Scaling

As in the previous section, we end with a general function for the metric MDS calculation. The function requires several input variables, including precision, accuracy, the minimization method, and the maximum number of iterations. This function uses the classical MDS routine for the initial coordinate estimates. Some suggested defaults are given in the function definition to be passed to NMinimize. The metric MDS function is written for two-dimensional mapping that could easily be generalized for three dimensions.

```

In[216]:= metricMDS [dmT_,
      method_: {"DifferentialEvolution", "SearchPoints" -> 30},
      precGoal_: 5, accurGoal_: 4, itr_: 100] :=
Module [ {m, n, Y, mY, stress, minVar, minSol, stval, sr},
      { {m, n} = {Length [dmT], 2};

      Y = Array[sr, {Length [dmT], 2}];
      minVar = Flatten[Y];

      mY =
      Table [Sqrt [ (Y[ [i, 1] ] - Y[ [j, 1] ] ) ^2 +
stress = Total [Table [ (dmT [ [i, j] ] - mY [ [i,
j] ] ) ^2, {i, m},
      {j, m} ], 2];

      minSol = NMinimize[stress, minVar, Method -> method,

```

```

PrecisionGoal → precGoal, AccuracyGoal → accurGoal,
MaxIterations → itr];
Y /. minSol [ [2] ]

}] [ [1] ]

```

6.4. Nonlinear Stress Functions

One of the advantages of metric MDS methods is the ability to use nonlinear mappings from N -dimensional to visualizable space. We now use this function to create an MDS map comparing the influenza hemagglutinin proteins by sequence similarity. Recall that the proximity measure for sequence comparison is the Smith–Waterman sequence distance, which is not a Euclidean distance function. Thus, this violates the assumptions of classical MDS and makes metric MDS the appropriate method to use, albeit at computational cost. Taking the hemagglutinin protein proximity matrix, fluPm, we apply the metricMDS function and obtain the coordinates. We also apply hierarchical clustering to the proximity matrix fluPm.

One of the advantages of metric MDS methods is that one can use nonlinear stress functions to emphasize particular regions of data relationships. One of the first nonlinear metric MDS methods was that of Sammon's mapping [12, 13]. Sammon's mapping can be useful in revealing underlying data structure or differences with nonlinear relationships. This mapping minimizes the following general nonlinear form of stress functions:

$$E(D_m, \Delta_m) = C \sum_{i=0}^c \sum_{j=0}^c (d_{ij} - \delta_{ij})^2 F(d_{ij}), \quad (13)$$

where F is a weighting function and C is a constant. In the case of Sammon's mapping, the stress function is defined by $F(x) = 1/x$, $k = 2$, with C specified as [12]:

$$C = \sum_{i=0}^c \sum_{j=0}^c d_{ij}. \quad (14)$$

Many related nonlinear mappings exist of the same form [12, 14]. For the purposes of this example, we explore applying a nonlinear exponential MDS mapping with a stress function defined by $F(x) = e^{-\beta d_{ij}}$, $k = 2$, $\alpha = 9$, $\beta = \alpha/\max(d_{ij})$, and $C = 1$:

$$E(D_m, \Delta_m) = C \sum_{i=0}^c \sum_{j=0}^c (d_{ij} - \delta_{ij})^2 e^{\frac{-9 d_{ij}}{\max(d_{ij})}}. \quad (15)$$

Note that α is an empirically specified tuning factor. This nonlinear mapping function decreases the contribution to the overall stress function of larger d_{ij} and has the effect of expanding the mapped distances between data points with smaller d_{ij} . The advantage of this mapping is that the weight of any point in the minimization is inversely proportional to its

magnitude. Thus, smaller differences between data elements are spread out. The coding of the exponential MDS function lets you specify α .

```
n[217]:= expMDS [dmT_, alpha_,
  method_: {"DifferentialEvolution", "SearchPoints" → 30},
  precGoal_: 5, accurGoal_: 4, itr_: 100]:=
Module [ {m, n, Y, mY, stress, minVar, minSol, stval, sr},
  { {m, n} = {Length [dmT], 2};
    Y = Array [sr, {Length [dmT], 2}];
    minVar = Flatten[Y];
    mY =
      Table [Sqrt [ (Y[ [i, 1] ] - Y[ [j, 1] ] ) ^2 +
        (Y[ [i, 2] ] - Y[ [j, 2] ] ) ^2], {i, m}, {j, m} ];
    stress =
      Total [Table [ ( (dmT[ [i, j] ] - mY[ [i, j] ] ) ^2)
        Exp[-alpha dmT[ [i, j]] / Max[dmT] ], {i, m}, {j,
m}], 2];
    minSol = NMinimize [stress, minVar, Method → method,
      PrecisionGoal → precGoal, AccuracyGoal → accurGoal,
      MaxIterations → itr];
    Y /. minSol [ [2] ]
  } ] [ [1] ].
```

To demonstrate, we apply the metric nonlinear MDS method to the above fluPm dataset of protein sequence comparisons. First, we apply the standard metric MDS function defined in the previous section, metricMDS, and generate a dendrogram to highlight the sequence differences (Figure 8).

While the major influenza viral subtypes (H1, H3, and H5) all cluster together, minimal differences can be observed with respect to sequences differences within each subtype. Using the nonlinear mapping, as shown in Figure 9, accentuates the small differences between the hemagglutinin protein sequences. Note the negative exponential weighting function, with the large distances between hemagglutinins being given less weight in the minimization. We are now able to visualize the division of the H1 influenza hemagglutinins into two major clusters and the split between clusters of the H3 substrains within the visualization. It is worth noting that the hierarchical clustering may capture these differences.

6.5. Synopsis: Metric MDS

Metric multidimensional scaling is a more flexible method compared to classical MDS. While computationally less efficient, it allows nonlinear dimensional reduction that is not possible with the SVD or PCA method of classical MDS. This functionality can be used to highlight possible clusters, similarities, or differences between cases described by high-dimensional attribute vectors and to weight or penalize the stress function based on data

attributes. As with classical MDS, complementary visualization methods allow for a fuller picture of case differences but must be carefully interpreted when nonlinear stress functions are used.

7. Some Considerations on Selecting a Method for MDS

From a computational perspective, several features of classical and metric MDS methods should be considered when selecting which method to use for a specific analysis.

Classical MDS is computationally straightforward but has some subtle mathematical constraints. For a solution to exist in Euclidean space, the distance matrix must follow a set of necessary and sufficient conditions [1, 15]. In practice, this is rarely an issue. When it is an issue, it often manifests by a computational result expressed in imaginary numbers. If, however, the proximity matrix is composed of Euclidean distances, classical MDS becomes principal component analysis, and the support vector machine (SVM) function can be used. This is much more computationally efficient, as we demonstrate in Section 7.3, and is the preferred method of MDS for visualization of most data.

Metric MDS has somewhat more relaxed constraints than classical MDS. For very high numbers of initial dimensions N , metric MDS is computationally and memory intensive. This is primarily due to the need to use iterative optimization to minimize the stress function, whereas classical MDS uses more efficient matrix algebra operations. In addition, one needs to pay attention to the algorithms used to perform metric MDS to ensure that minimization finds global rather than local minima.

We next examine the selection of minimization algorithms for metric MDS, compare the computational efficiency of classical and metric MDS methods, and discuss the particular circumstances where metric MDS may be the method of choice.

7.1 Convergence, Accuracy, and Selection of Minimization Algorithms for Metric MDS

With metric MDS, selecting a minimization algorithm with an appropriate Method setting for *Mathematica*'s built-in function NMinimize may require some investigation. In addition, a known issue with metric MDS is the existence of local minima, so that a global minimum cannot be guaranteed [7]. Using a large number of starting points or selecting the appropriate minimization function and method may help. In this article we have used NMinimize with Method set to the "RandomSearch" option. We also specified sufficient search points to avoid local minima. This can be an important issue when no other external data is available to constrain the solution and when it is not known if other minimization methods will routinely converge to a solution.

It is worth exploring this tradeoff between convergence, accuracy, and computational efficiency. For example, the execution time of NMinimize increases with the number of initial search points. The next example uses NMinimize on the influenza antibody dataset distance matrix D_m and the method "RandomSearch". In this case, however, the number of search points appears to make little difference in the residual after optimizing the stress function (Figure 10).

```

In[227]:= sp={1, 5, 10, 20, 30, 40, 50, 100, 150, 200, 250, 300};
minSolT =
  Parallelize[
    Timing[NMinimize[stress, minVar,
      Method → {"RandomSearch", "SearchPoints" →
#}]]& /@sp];

```

In the case of RandomSearch, there appears to be no tradeoff with this particular dataset. In contrast, the theoretically more robust setting Method → DifferentialEvolution finds a small number of varying minima, which improves after specifying a larger number of search points. Also note the improved execution times, which are an order of magnitude less than with RandomSearch (Figure 11).

```

In[232]:= minSolTde =
  Parallelize[
    Timing[NMinimize[stress, minVar,
      Method → {"DifferentialEvolution",
"SearchPoints" → #}]]& /@
    sp];

```

Another consideration is the speed of computation. The FindMinimum function is often much faster than NMinimize, but the tradeoff is the risk that the algorithm will not converge or that it will find local instead of global minima. For the influenza data, FindMinimum can suffer from both nonconvergence and local minima, although the former issue may be sporadic. The sporadic nature of convergence can be frustrating and problematic. The issue of finding local minima is potentially more serious. To explore this further, we minimized 300 times with FindMinimum.

```

In[236]:= my1 =
  Table[Sqrt[(Y[[i, 1]] - Y[[j, 1]])^2 + (Y[[i, 2]] -
Y[[j, 2]])^2],
    {i, lenDm}, {j, lenDm}];
stress1 = Total[Table[((Dm[[i, j]]/10.)^2 - my1[[i, j]])^2,
    {i, lenDm}, {j, lenDm}], 2];

simMin =
  Quiet[
    Table[
      {start =
        Thread[{minVar, RandomReal[{-1000, 1000},
          Length[minVar]]}];
      test =

```



```

Check[
  mSol2 = Timing[FindMinimum[stress1, start,
    Method -> "LevenbergMarquardt",
    AccuracyGoal -> 6,
    PrecisionGoal -> 5]], {}];
If[test == 0, {"Converges", mCoords2 = Y /.
  mSol2[[2, 2]],
  timing2 = mSol2[[1]],
  residual2 = mSol2[[2, 1]]}, {"Does Not Converge", mCoords2 = Y /.
  mSol2[[2, 2]],
  timing2 = mSol2[[1]],
  residual2 = mSol2[[2, 1]]}][[1]], {i, 1,
  300}];

sim2 = simMin[All, 1];

```

The results are displayed as a histogram in Figure 13. We found that a number did not converge with the influenza antibody-reactivity dataset. Of even more concern is that FindMinimum was not consistent, and often appeared to provide local rather than global minima. This may in some cases be dataset specific. The starting points selected, here a RandomReal number between -1000 and 1000 for each coordinate, are also critical for FindMinimum.

The convergence can be improved by minimizing the sum of the squares rather than the Euclidean distance. Note that the minimum residual is higher than when minimizing the sum of the differences between distances rather than the square of the difference between distances. This may be due to computational considerations, in that there is differentiability at the optima when minimizing the sum of the squares of the distances, while this is lacking if minimizing the difference between the distances. Still, a moderate number (2% to 15% from 300 attempts) of the solution attempts did not converge or find global minima. Whether this is an issue for your particular application depends on both the level of precision required and the computational efficiency required (Figure 13).

```

In[246]:= mY3 = Table[(Y[[i, 1]]-Y[[j, 1]])^2 +(Y[[i, 2]]-Y[[j, 2]])^2,
  {i, lenDm}, {j, lenDm}];
stress3 = Total[Table[(Dm[[i, j]]/ 10.)^2 -mY3[[i, j]]^2,
  {i, lenDm}, {j, lenDm}], 2];

```

Given the low percentage of nonconvergence and the presence of local minima, one could execute FindMinimum several times and take the results that converge with the lowest minimum. This does increase the time for execution. An alternative is to use the function NMinimize. The tradeoff is an increased computational time for consistent convergence on

global minima, as shown here. This demonstration takes several minutes to perform on this dataset, even with parallelization (Figure 14).

```

In[256]:= mY3 =Table[(Y[[i, 1]]-Y[[j, 1]])^2 +(Y[[i, 2]]-Y[[j, 2]])^2,
               {i, lenDm}, {j, lenDm}];
stress3 =Total[Table[(Dm[[i, j]]/ 10.)^2 -mY3[[i, j]]^2,
               {i, lenDm}, {j, lenDm}], 2];

Quiet [
  simMin4 =ParallelTable[
    {test4 =
      Check[
        mSol4 =Timing[minSol =NMinimize[stress3,
minVar,
                                     Method ->
{"DifferentialEvolution",
                                     "SearchPoints" -> 30},
AccuracyGoal -> 6,
                                     PrecisionGoal -> 5]], {}];
      If[test4 {}, {"Converges", mCoords4 =Y /.
mSol4[[2, 2]],
          timing4 =mSol4[[1]],
          residual4 =mSol4[[2, 1]]},
        {"Does Not Converge", mCoords4 =Y /.
mSol4[[2, 2]],
          timing4 =mSol4[[1]],
          residual4 =mSol4[[2, 1]]}]]][[1]], {i, 1,
300}]]];

sim4 =simMin4[[All, 1]];

```

A single minimum residual is obtained for all 300 separate runs of NMinimize. Thus, unless local minima are known not to be an issue, NMinimize may be a more consistent minimization method for metric MDS routines.

7.2. Computational Efficiency

The issues of optimization and computational efficiency make it useful to benchmark classical and metric MDS methods with respect to execution time. Therefore, we compared SVD with classical and metric MDS across the three example datasets. For metric MDS, we include both FindMinimum and NMinimize. The measured execution times encompass all the steps from the input of the proximity matrix through final coordinate output, but not plotting. The previously created classicalMDS and metricMDS functions defined previously were used, along with new functions created for singular value decomposition (mdsSVD) and metric MDS using FindMinimum (findMinMDS).

```

In[267]:= mdsSVD[data_, dimensions_: 2] :=
  Module[{hmat, len, zeromeanData, uu, ww, vv}, {
    len = Length[data];
    hmat = IdentityMatrix[len] -1 / len;
    zeromeanData = -hmat.N[data^2] .hmat / 2;
    {uu, ww, vv} =
SingularValueDecomposition[zeromeanData,
  dimensions]}][[1]]

  findMinMDS[dmT_, start_: {-1000, 1000},
    method_: "LevenbergMarquardt", precGoal_: 5, accurGoal_:
6,
    itr_: 100] :=
  Module[{m, n, Y, mY, stress, minVar, minSol, stval, sr},
    {{m, n} = {Length[dmT], 2};
    Y = Array[sr, {Length[dmT], 2}];
    minVar = Flatten[Y];
    mY =
      Table[((Y[[i, 1]] - Y[[j, 1]]) ^2 + (Y[[i, 2]]
- Y[[j, 2]]) ^2),
        {i, m}, {j, m}];
    stress = Total[Table[(dmT[[i, j]] - mY[[i, j]])
^2, {i, m},
        {j, m}], 2];
    minSol = FindMinimum[stress,
      Transpose[{minVar, RandomReal[start,
Length[minVar]]}],
      Method → method, PrecisionGoal → precGoal,
      AccuracyGoal → accurGoal, MaxIterations
→ itr];
    Y / . minSol[[2]]
  ]][[1]]

In[269]:= times =
  Transpose[
    Parallelize[
      {Timing[mdsSVD[#, 2]][[1]], Timing[classicalMDS[#]]
[[1]],
        Timing[findMinMDS[#]][[1]],
        Timing[metricMDS[#, {"DifferentialEvolution",
          "SearchPoints" → 5}]]][[1]]}& /@
{airData, Dm, fluPm}];

```

```
(kernel 11) FindMinimum::cvmit :
Failed to converge to the requested accuracy or precision within 100 iterations.

(kernel 12) FindMinimum::cvmit :
Failed to converge to the requested accuracy or precision within 100 iterations.
```

Note that FindMinimum failed to converge for the airport dataset but did converge for both the flu antibody and sequence datasets, a behavior that may vary between executions. Timings are shown here (Table 6).

Overall, there is a speed advantage to using classical MDS and SVD, which are several orders of magnitude faster than metric MDS using either FindMinimum or NMinimize. This advantage assumes that the proximity matrix can be formulated using a Euclidean distance, and thus other measurements (e.g. Manhattan distance, etc.) may require other methods. It also assumes that you require a linear mapping from the N -dimensional space to the visualizable space. SVD has the advantage that, for many applications, the proximity matrix may not have to be directly calculated, saving additional memory and computational time [8]. With respect to metric MDS, there are also tradeoffs. For moderately large datasets, if one can optimize the conditions to avoid nonconvergence and local minima, FindMinimum is a good choice. Note that the execution time for FindMinimum in the airport and influenza antibody dataset examples occurred in the setting of failure to converge. Although slower, metric MDS using NMinimize had no issues with nonconvergence or finding local instead of global minima.

7.3. Some Thoughts on Method Selection: Classical versus Metric MDS

Given the computational advantages of classical MDS in its various forms (principal component analysis, singular value decomposition, etc.), why would you choose metric MDS, which requires numerical optimization? For those wishing an in-depth and lucid explanation, we refer you to the initial work of Gower [15] and the outstanding article by Shiens [16]. In short, classical MDS methods are usually preferable to and always computationally more efficient than metric MDS methods. The method is computationally efficient and robust; there are very few cases where classical MDS methods fail to provide a visualization. For some datasets, classical MDS methods may not provide optimal visualization, and metric MDS should be considered. Some examples include datasets whose variables have a non-Gaussian distribution, those where the distance metric is not Euclidean, those requiring imputation of missing data [10, 11], or cases with a parametrization in a nonorthogonal coordinate system [16]. For nonlinear dimensional reduction, which is often used in graph layout algorithms, metric MDS with stress function minimization is the method of choice [14].

Thus, selection of an MDS method for visualization should consider several factors. If the distance function is Euclidean, and especially if the dataset is large, SVD or classical MDS are the most appropriate and computationally efficient methods. In the less common case where metric MDS is used, careful consideration should be given to the choice of stress function minimization method (in this example NMinimize versus FindMinimum) to avoid local minima.

8. Summary and Conclusion

We have demonstrated the application of two methods of multidimensional scaling, classical and metric, for the visualization of similarity/proximity of high-dimensional data, with reduction to two or three dimensions. We have shown how these methods can be used to visualize the relatedness of influenza virus strains with respect to antibody-mediated immunity, as well as their utility in reconstructing relative spatial-geographic locations using only interlocation distances. These MDS methods are, however, quite generalizable. Both classical and metric MDS rely on a proximity matrix. While the examples in this article use continuous variables and sequences or case attributes, a variety of data types with appropriate proximity metrics can be visualized with MDS methods, such as molecular DNA or protein sequence data (Smith–Waterman similarity, Needleman–Wunsch similarity, or Damerau–Levenshtein distance), Boolean data (Hamming distance), and images (image distance, color distance). For cases with a single data attribute (e.g. sequence similarity, distance), no data scaling is necessary. For cases with multiple attributes having disparate units, standardization (e.g. z -score) and rescaling are needed to equally weight each attribute.

In our examples, we used reduction down to two dimensions for multidimensional data visualization. Reduction to three dimensions is easily accomplished with both classical and metric MDS, although the computational cost may increase with the added dimension, depending on the method used. While we have emphasized data visualization, the classical MDS method can also be used for reduction to four or more dimensions as a method of identifying the weighted combinations of components that contribute the most to the data variance. In these cases, the goal is to select a small set of variables that explain the most variance in the dataset, such that this minimum set of variables can be further used for statistical or other modeling.

We have also shown how MDS methods are related to clustering algorithms, which also use proximity matrices to compare and classify cases by their attributes. This relationship also allows creative graphical display of multidimensional data from several vantage points. For example, one can use the MDS plot to display the relative proximity of cases to each other and plot marker coloring or other methods to add information regarding other case attributes. Some caution is in order, however, as different proximity measures and data transformations may give different clustering and classification. Parallel display of dendrograms and heat maps may also enhance understanding of the relationship of data clusters to each other. Similarly, heat maps, combined with MDS displays, are particularly helpful for data exploration, in that they enhance visual identification of those data attributes (dimensions) that contribute the most to differentiating between case clusters.

Care should be taken when selecting the MDS method. In most cases, classical MDS will be the most computationally efficient method, especially for very large datasets. In the cases where metric MDS is optimal, such as the use of nonlinear mapping, care should be taken to choose a minimization method that is robust and avoids local minima. Performing some testing on a subset of the data can be very informative regarding convergence, accuracy, and computational efficiency. While we did not discuss in detail how constraining optimization

problems can improve computational efficiency and accuracy, this should also be considered whenever boundary conditions or other information is available.

Finally, one must remain aware that these methods reveal only associative patterns. Further analysis with rigorous statistical inference methods is needed to test the validity and specify the error boundaries of these associations. Mechanistic studies should be performed, if possible, to confirm suspected causal relationships. Overall, however, MDS methods are excellent for dimensional reduction and data exploration with the goal of creating comprehensible and informative quantitative graphical representations of multidimensional data.

Acknowledgments

We would like to thank Ollivier Hyrien, Alex Rosenberg, Chris Fucile, and Melissa Trayhan for their suggestions and critical reading of the manuscript. We would also like to acknowledge the reviewer at the *Mathematica Journal*, who suggested several improvements and additions, including the evaluation of execution time and accuracy for the MDS methods, and the discussion regarding the relative merits of SingularValueDecomposition, NMinimize, and FindMinimum. This work was supported by grants to the authors from the Philip Templeton Foundation, as well as the National Institute of Allergy and Infectious Diseases and the National Institute of Immunology, grant and contract numbers HHSN272201000055C, AI087135, AI098112, AI069351.

References

1. Gower JC. Some Distance Properties of Latent Root and Vector Methods Used in Multivariate Analysis. *Biometrika*. 1966; 53(3–4):325–338.10.1093/biomet/53.3-4.325
2. Torgerson WS. Multidimensional Scaling: I. Theory and Method. *Psychometrika*. 1952; 17(4):401–419.10.1007/BF02288916
3. Torgerson, WS. *Theory and Methods of Scaling*. New York: Wiley; 1958.
4. Cox, TF.; Cox, MAA. *Multidimensional Scaling*. Boca Raton: Chapman and Hall; 2001.
5. Borg, I.; Groenen, P. *Modern Multidimensional Scaling: Theory and Applications*. 2. New York: Springer; 2005. p. 207–212.
6. Wish, M.; Carroll, JD. Multidimensional Scaling and Its Applications. In: Krishnaiah, PR.; Kanai, LN., editors. *Handbook of Statistics, Vol. 2: Classification, Pattern Recognition and Reduction of Dimensionality*. Amsterdam: North-Holland; 1982. p. 317–345.
7. Saltenis, V. Constrained Optimization of the Stress Function for Multidimensional Scaling. In: Alexandrov, V.; van Albada, G.; Sloot, PA.; Dongarra, J., editors. *Computational Science–ICCS 2006*. Berlin: Springer; 2006. p. 704–711.link.springer.com/chapter/10.1007/11758501_94
8. Smith DJ, Lapedes AS, de Jong JC, Bestebroer TM, Rimmelzwaan GF, Osterhaus ADME, Fouchier RAM. Mapping the Antigenic and Genetic Evolution of Influenza Virus. *Science*. 2004; 305(5682): 371–376.10.1126/science.1097211 [PubMed: 15218094]
9. Bedford T, Suchard MA, Lemey P, Dudas G, Gregory V, Hay AJ, McCauley JW, Russell CA, Smith DJ, Rambaut A. Integrating Influenza Antigenic Dynamics with Molecular Evolution. *eLife*. 2014; 3:e01914.10.7554/eLife.01914 [PubMed: 24497547]
10. Cai Z, Zhang T, Wan X-F. Concepts and Applications for Influenza Antigenic Cartography. *Influenza and Other Respiratory Viruses*. 2011; 5(Suppl s1):204–207. www.ncbi.nlm.nih.gov/pmc/articles/PMC3208348. [PubMed: 21761589]
11. Fanning DW, Smith JA, Rose GD. Molecular Cartography of Globular Proteins with Application to Antigenic Sites. *Biopolymers*. 1986; 25(5):863–883.10.1002/bip.360250509 [PubMed: 2424521]
12. Lespinats, S.; Aupetit, M. False Neighbourhoods and Tears are the Main Mapping Defaults. How to Avoid It? How to Exhibit Remaining Ones?. QIMIE/PAKDD 2009; PAKDD Workshops; Thailand. Bangkok: 2009. p. 55–64.conferences.telecom-bretagne.eu/data/qimie09/lespinats_aupetit_QIMIE_2009.pdf

13. Sammon JW. A Nonlinear Mapping for Data Structure Analysis. IEEE Transactions on Computers. 1969; C-18(5):401–409.10.1109/T-C.1969.222678
14. Chen L, Buja A. Stress Functions for Nonlinear Dimension Reduction, Proximity Analysis, and Graph Drawing. Journal of Machine Learning Research. 2013; 14(1):1145–1173. dl.acm.org/citation.cfm?id=2502616.
15. Gower JC. A Comparison of Some Methods of Cluster Analysis. Biometrics. 1967; 23(4):623–637.10.2307/2528417 [PubMed: 6080200]
16. Shiens, J. A Tutorial on Principal Component Analysis. arxiv.org/pdf/1404.1100.pdf
17. Zand MS, Wang J, Hilchey S. Graphical Representation of Proximity Measures for Multidimensional Data. The Mathematica Journal. 201510.3888/tmj.17-8

Biographies

Martin S. Zand is a professor of medicine and director of the Rochester Center for Health Informatics at the University of Rochester. His research includes the application of informatics, graph theory, and computational modeling to vaccine immunology, gene regulatory networks, and health care delivery.

Jiong Wang is a molecular virologist and research assistant professor at the University of Rochester. She works on influenza vaccination responses and developing high-dimensional measurements of vaccine immune responses.

Shannon Hilchey is a cellular immunologist and research associate professor at the University of Rochester. He studies human immune responses to vaccination, hematologic cancers, and organ transplants in human subjects.

```

In[159]:= labels = Partition[Riffle[Range[Length[seraLabLong]], seraLabLong],
2];
cfunc = Blend[{Lighter[Blue, 0.5], White, Lighter[Red, 0.5]}, #1] &;
clegend = Magnify[colorbarL[{0, 1000}, cfunc, 150, "Distance"],
0.9];

p11 = ArrayPlot[Dm, ColorFunction -> cfunc, Frame -> True,
FrameTicks -> {labels, None, None,
Table[{i, Rotate[labels[[i, 2]], - $\frac{\pi}{2}$ ]}, {i, 1, Length[labels]}}],
ImageSize -> {300, 300},
BaseStyle -> Directive[FontFamily -> "Arial", 8, Bold]];
scale1 = colorbarL[{Min[Dm], Max[Dm]}, cfunc, 150, "Distance",
{200, 80}];

Column[{p11, scale1}, Alignment -> Right]

```

Figure 1.
Heat map of proximity matrix data.


```

In[180]:= Framed[
Graphics[
Table[
{Text[Style[airLabsShort[[i]], Red, Bold, FontFamily -> "Arial",
9], cMDScoord[[i]]]}, {i, 1, Length[cMDScoord]}],
ImageSize -> Medium], FrameStyle -> LightGray]

```

Figure 2.
Classical MDS display of relative airport locations.

```

In[279]:= TrcMDSCoordinates = cMDSCoord;
geoCoordsAir = Table[{airGeo[[i, 1, 2]], airGeo[[i, 1, 1]]},
  {i, Length[airGeo]};

correl =
  Correlation[geoCoordsAir[[All, #]],
    TrcMDSCoordinates[[All, #]] & /@ {1, 2};

pLabel =
  Style[
    Column[
      {Row[{"Longitude", "Latitude"}[[#]], " versus ",
        {Style["X", Italic], Style["Y", Italic]}[[#]]},
      Row[{Style["r", Italic], " = ", correl[[#]]}], Center],
    FontFamily -> "Arial", Bold, 10, Black] & /@ {1, 2};

Grid[
  Transpose[
    {Labeled[
      ListPlot[Table[{geoCoordsAir[[i, #]],
        TrcMDSCoordinates[[i, #]]}, {i, Length[geoCoordsAir]}],
      Axes -> False, Frame -> {{True, False}, {True, False}},
      BaseStyle -> Directive[FontFamily -> "Arial", 9],
      FrameLabel -> {{Column[{"", "Metric MDS Coordinates"}],
        ""}, {"True Coordinates", ""}},
      ImageSize -> {250, Automatic}], pLabel[[#]], Top] & /@
    {1, 2}}]]

```

Figure 3.

Comparison of actual and estimated coordinates.

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

```
In[277]:= Magnify[GeoListPlot[{airGeo, geoMDS}, PlotStyle -> {Black, Red},
  PlotLegends -> {"Actual", "cMDS"}, ImageSize -> {350, Automatic}],
  0.9]
```

Figure 4.
Comparison of airport geo locations.

```

In[206]:= Magnify[
  Framed[
    Graphics[
      Table[
        {Text[Style[seraLabShort[[i]], seraColor[[i]], Bold,
          FontFamily -> "Arial", 10],
          {mMDScoordinates[[i, 1]], mMDScoordinates[[i, 2]]}],
        {i, 1, Length[seraLabShort]}], FrameStyle -> LightGray],
    0.7]

```

Figure 5.
Relative antigenic distances for influenza strains.

`In[278]:= DendrogramPlot[clustTR, LeafLabels -> {# &}, Orientation -> Left]`

Figure 6.
Dendrogram of influenza virus relationships.

```

In[211]:= cfunc = Blend[{Lighter[Blue, 0.5], White, Lighter[Red, 0.5]}, #] &;
clegend = Magnify[colorbarL[{0, 1000}, cfunc, 150, "Distance"],
0.9];
labels = Partition[Riffle[Range[Length[seraLabLong]], seraLabLong],
2];

p17 = ArrayPlot[DmC, ColorFunction -> cfunc, Frame -> True,
FrameTicks -> {labels, None, None,
Table[{i, Rotate[labels[[i, 2]], -Pi/2]}],
{i, 1, Length[labels]}}], ImageSize -> {300, Automatic},
BaseStyle -> Directive[FontFamily -> "Arial", 9, Bold]];

Column[{p17, clegend}, Alignment -> Right]

```

Figure 7.
Heat map of proximity matrix data.

```

In[271]:= fluSeqCoords = metricMDS[fluPm];

im1 =
  Framed[
    Graphics[
      Table[
        {Text[Style[seraLabShort[[i]], seraColor[[i]], Bold,
          FontFamily -> "Arial", 8], fluSeqCoords[[i]]]},
        {i, Length[fluSeqCoords]}], ImageSize -> {350, 150}],
    FrameStyle -> LightGray];

clustTR = DirectAgglomerate[fluPm, virusLabelsColor,
  Linkage -> "Average"];

im2 = DendrogramPlot[clustTR, LeafLabels -> {# &}, Orientation -> Left,
  ImageSize -> Medium];

Column[{Magnify[im1, 0.9], " ", Magnify[im2, 0.9]}]

```

Figure 8.
Influenza virus relationships from protein sequences.


```

In[314]:= nlpla =
  ListLogPlot[
    Flatten[
      Table[{fluPm[[i, j]], Exp[-9. fluPm[[i, j]] / Max[fluPm]]},
        {i, Length[fluPm]}, {j, Length[fluPm]}], 1],
    Frame → {{True, False}, {True, False}},
    FrameStyle → Directive[FontFamily → "Arial", 8, Bold],
    FrameLabel → {{{"Weighting", None},
      {"Protein pair dissimilarity", Null}},
    ImageSize → {300, Automatic}];

nlpCoords = expMDS[fluPm, -4.5];

map =
  Framed[
    Graphics[
      Table[
        {Text[Style[seraLabShort[[i]], seraColor[[i]], Bold,
          FontFamily → "Arial", 9],
          {nlpCoords[[i, 1]], -nlpCoords[[i, 2]]}],
        {i, Length[nlpCoords]}], ImageSize → {350, Automatic}],
    FrameStyle → LightGray];

Magnify[Column[{nlpla, "      ", map}], 0.8]

```

Figure 9.

Weighting function and nonlinear mapping of protein sequences.

```

In[229]:= p11a = ListLinePlot[Partition[Riffle[sp, minSolt[[All, 1]]], 2],
  PlotMarkers → Automatic, Frame → {{True, False}, {True, False}},
  FrameStyle → Directive[FontFamily → "Arial", 8, Bold],
  FrameLabel → {"Execution Time (seconds)", None},
  {"Search Points", Null}}, ImageSize → 200];

p11b =
  ListLinePlot[
    Partition[Riffle[sp, minSolt[[All, 2, 1]] / 10 000 000], 2],
    PlotMarkers → Automatic, PlotRange → {{0, 320}, {1, 3.0}},
    Frame → {{True, False}, {True, False}},
    FrameStyle → Directive[FontFamily → "Arial", 8, Bold],
    FrameLabel → {"Residual x 107", None}, {"Search Points", Null}},
    ImageSize → 200];

Row[{p11a, p11b}]

```

Figure 10.
Efficiency of NMinimize (RandomSearch).

Figure 11.
Efficiency of NMinimize (Differential Evolution).

Figure 12.
Residuals and convergence using FindMinimum.

```

In[248]:= simMin3 =
  Quiet[
    Table[
      {start3 =
        Thread[{minVar, RandomReal[{-1000, 1000},
          Length[minVar]]}];
      test =
        Check[
          mSol3 = Timing[FindMinimum[stress3, start3,
            Method -> "LevenbergMarquardt", AccuracyGoal -> 6,
            PrecisionGoal -> 5]], {}];
      If[test != {}, {"Converges", mCoords3 = Y /. mSol3[[2, 2]],
        timing3 = mSol3[[1]],
        residual3 = mSol3[[2, 1]]},
        {"Does Not Converge", mCoords3 = Y /. mSol3[[2, 2]],
        timing3 = mSol3[[1]],
        residual3 = mSol3[[2, 1]]}][[1]], {i, 1, 300}];

    sim3 = simMin3[[All, 1]];
    mins3 = First[Sort[Tally[SetPrecision[simMin3[[All, 4]], 12]]]];
    minPct3 = {"Global Min %",
      ToString[SetPrecision[100. mins3[[2]] / 300., 3]] <> " %"};
    gMin3 = {"Global Min", ToString[Round[Min[simMin3[[All, 4]]]]];

    tf3 =
      Style[
        TableForm[Append[Tally[sim3], Transpose[{minPct3, gMin3}]]],
        FontFamily -> "Arial", 12];

    his3 = Histogram[simMin3[[All, 4]], {6. × 109, 5. × 1010, 1. × 109},
      PlotRangePadding -> 2, AxesOrigin -> {1. × 109, 0},
      BaseStyle -> Directive[FontFamily -> "Arial", FontSize -> 6],
      Frame -> {{True, False}, {True, False}},
      FrameLabel -> {{ "Counts", None}, {"Residual", None}}];

    Column[{Magnify[his3, 1.7], tf3}]

```

Figure 13.

FindMinimum and minimizing the sum of the squares.

Figure 14.
Residuals and convergence using NMinimize.

Table 1

Airport distance proximity matrix.

$In[302] := \text{Magnify}[\text{scrollTable}[\text{airData}, \text{airLabsShort}, \text{airLabs}, \{\text{Automatic}, \text{Automatic}\}, \{\text{False}, \text{False}\}], 0.7]$

	ATL	BIL	BHM	BIS	BOI	BOS	BUF	ORD	CLE	DFW	DEN	DSM	DTW	ELP	HOU
Atlanta (ATL)	0	1525	134	1247	1838	946	712	606	555	731	1199	743	594	1282	696
Billings (BIL)	1525	0	1428	379	409	1866	1481	1060	1366	1081	455	800	1274	973	1327
Birmingham (BHM)	134	1428	0	1171	1727	1050	779	584	604	597	1083	667	625	1148	570
Bismark (BIS)	1247	379	1171	0	786	1490	1109	716	1007	977	516	504	913	1075	1217
Boise (BOI)	1838	409	1727	786	0	2265	1876	1437	1750	1272	649	1156	1660	972	1501
Boston (BOS)	946	1866	1050	1490	2265	0	396	867	563	1562	1754	1165	632	2067	1609
Buffalo (BUF)	712	1481	779	1109	1876	396	0	473	192	1212	1359	771	241	1693	1297
Chicago (ORD)	606	1060	584	716	1437	867	473	0	316	802	888	299	234	1236	945
Cleveland (CLE)	555	1366	604	1007	1750	563	192	316	0	1021	1201	613	95	1509	1107
Dallas (DFW)	731	1081	597	977	1272	1562	1212	802	1021	0	641	624	986	551	247
Denver (DEN)	1199	455	1083	516	649	1754	1359	888	1201	641	0	589	1123	563	883
Des Moines (DSM)	743	800	667	504	1156	1165	771	299	613	624	589	0	534	972	825
Detroit (DTW)	594	1274	625	913	1660	632	241	234	95	986	1123	534	0	1455	1092
El Paso (ELP)	1282	973	1148	1075	972	2067	1693	1236	1509	551	563	972	1455	0	677
Houston (HOU)	696	1327	570	1217	1501	1609	1297	945	1107	247	883	825	1092	677	0
Indianapolis (IND)	432	1202	425	875	1563	818	452	177	261	761	976	407	231	1250	861
Kansas City (MCI)	692	835	594	599	1152	1256	870	403	694	460	533	164	629	835	666
Little Rock (LIT)	453	1148	324	942	1419	1260	920	552	729	304	771	476	705	843	393
Los Angeles (LAX)	1946	970	1815	1280	674	2611	2217	1745	2053	1235	862	1447	1979	714	1390
Louisville (SDF)	321	1281	323	967	1630	829	494	287	304	733	1024	481	306	1247	803
Memphis (MEM)	332	1224	211	984	1516	1139	813	491	623	431	872	490	610	972	484
Miami (MIA)	595	2083	661	1831	2358	1258	1185	1197	1080	1121	1709	1327	1145	1631	955
Minneapolis (MSP)	907	748	854	386	1142	1124	734	334	622	852	680	232	528	1147	1057
New Orleans (MSY)	425	1472	321	1287	1706	1367	1097	837	917	448	1062	818	926	965	302
New York (JFK)	760	1776	866	1407	2167	187	301	740	425	1391	1626	1038	509	1907	1428
Omaha (OMA)	821	706	732	448	1048	1282	887	416	729	583	472	117	650	875	804

$Out[302] =$

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Philadelphia (PHL)	666	1727	772	1363	2113	280	279	678	363	1303	1558	972	453	1824	1336
Phoenix (PHX)	1587	873	1455	1094	735	2300	1912	1440	1737	868	602	1149	1671	347	1020
Pittsburgh (PIT)	526	1469	598	1112	1850	496	186	412	106	1067	1290	705	201	1572	1131
Portland (PDX)	2172	679	2064	1048	344	2537	2157	1739	2046	1616	991	1471	1953	1288	1843
Raleigh Durham (RDU)	356	1690	480	1360	2045	612	487	646	416	1061	1436	892	501	1606	1050
St. Louis (STL)	484	1048	411	764	1383	1047	674	258	487	550	770	259	440	1022	687
Salt Lake City (SLC)	1590	387	1472	696	290	2105	1710	1250	1565	989	391	955	1481	693	1214
San Francisco (SFO)	2139	909	2013	1271	522	2704	2309	1846	2161	1464	967	1549	2079	993	1649
Seattle (SEA)	2182	664	2079	1015	399	2496	2122	1721	2021	1660	1024	1464	1927	1368	1894
Washington (DCA)	547	1671	653	1316	2048	399	296	612	310	1192	1476	897	405	1719	1218

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 2

Influenza antibody binding assay data.

$In[313]:= \text{Magnify}[\text{ScrollTable}[\text{fluData}, \text{virusLabShort}, \text{seralabLong}, \{\text{Automatic}, \text{Automatic}\}, \{\text{False}, \text{False}\}], 0.7]$															
	CAL709-H1	SC0118-H1	PR34-H1	USSR77-H1	TX91-H1	PER1609-H3	VIC361-H3	TX12-H3	WISC05-H3	HIRO05-H3	PC73-H3	HK68-H3	VIE105-H5	INDO05-H5	
A/California/07/2009 (H1)	5586	274	491	246	73	0	0	0	0	16	0	0	61	47	
A/Mexico/4108/2009 (H1)	4876	332	441	215	131	74	27	59	51	8	54	78	94	158	
A/Utah/20/2009 (H1)	5428	819	238	165	426	55	20	33	33	26	88	78	97	109	
A/Brisbane/59/2007 (H1)	260	720	412	707	951	47	2	12	4	22	15	33	281	223	
A/USSR/90/1977 (H1)	303	805	530	7472	1204	135	109	92	77	75	206	213	451	447	
A/Taiwan/1/1986 (H1)	53	236	195	527	5791	96	32	9	28	28	115	103	193	129	
A/Indiana/10/2011 (H3)	137	119	157	81	135	719	1547	1551	2045	3914	1701	1775	116	220	
A/Victoria/361/2011 (H3)	151	100	76	118	89	3403	5910	4609	1006	6561	1324	1821	49	142	
A/Wisconsin/15/2009 (H3)	86	49	97	147	52	3391	5159	3621	717	3972	1352	1578	74	124	
A/Perth/16/2009 (H3)	84	13	95	145	56	3784	5520	3443	703	3561	1119	1292	54	112	
A/Victoria/210/2009 (H3)	0	0	0	0	0	4727	7192	5422	491	4336	832	1284	0	0	
A/Vietnam/1203/2004 (H5)	59	157	147	335	434	48	33	7	33	0	101	121	8806	5432	
A/India/NIV/2006 (H5)	184	304	183	350	211	290	232	147	271	210	442	474	1601	1538	
A/Anhui/01/2005 (H5)	24	59	76	179	55	12	29	1	44	17	108	124	3044	4330	
A/Hubei/1/2010 (H5)	94	581	133	147	317	60	17	8	46	16	80	95	5223	5164	

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 5

Proximity matrix for influenza sequence comparisons.

<div>In[165]:= fluPm = ParallelTable[(2 x 10^6 / SmithWatermanSimilarity[seqFluHA[[i]], seqFluHA[[j]], SimilarityRules -> "PAM70" - 528), {i, Length[seqFluHA]}, {j, Length[seqFluHA]}];</div>																																																																																																																																																																																																																																
<div>In[303]:= Magnify[ScrollTable[Round[fluPm], serialabLong, serialabLong, {Automatic, Automatic}, {False, False}], 0.65]</div>																																																																																																																																																																																																																																
<div>Out[303]=</div> <table><thead><tr><th></th><th>A/California/07/2009 (H1)</th><th>A/Mexico/4/08/2009 (H1)</th><th>A/Utah/20/2009 (H1)</th><th>A/Brisbane/59/2007 (H1)</th><th>A/USSR/90/1977 (H1)</th><th>A/Taiwan/1/1986 (H1)</th><th>A/Indiana/01/20/11 (H1)</th><th>A/Vietnam/15/2006 (H1)</th><th>A/Vietnam/21/02/2009 (H1)</th><th>A/Vietnam/12/03/2004 (H1)</th><th>A/India/NV/2006 (H1)</th><th>A/Ahmad/01/2005 (H1)</th><th>A/Hubei/1/2010 (H1)</th></tr></thead><tbody><tr><td>A/California/07/2009 (H1)</td><td>10</td><td>13</td><td>19</td><td>148</td><td>142</td><td>138</td><td>1302</td><td>1300</td><td>1293</td><td>1750</td><td>370</td><td>353</td><td>352</td></tr><tr><td>A/Mexico/4/08/2009 (H1)</td><td>13</td><td>11</td><td>17</td><td>153</td><td>144</td><td>140</td><td>1295</td><td>1298</td><td>1287</td><td>1747</td><td>369</td><td>352</td><td>351</td></tr><tr><td>A/Utah/20/2009 (H1)</td><td>19</td><td>17</td><td>12</td><td>160</td><td>149</td><td>147</td><td>1292</td><td>1297</td><td>1285</td><td>1734</td><td>364</td><td>349</td><td>350</td></tr><tr><td>A/Brisbane/59/2007 (H1)</td><td>148</td><td>153</td><td>160</td><td>10</td><td>65</td><td>47</td><td>1375</td><td>1317</td><td>1312</td><td>1734</td><td>376</td><td>360</td><td>355</td></tr><tr><td>A/USSR/90/1977 (H1)</td><td>142</td><td>144</td><td>149</td><td>65</td><td>8</td><td>28</td><td>1414</td><td>1406</td><td>1416</td><td>1853</td><td>370</td><td>363</td><td>358</td></tr><tr><td>A/Taiwan/1/1986 (H1)</td><td>138</td><td>140</td><td>147</td><td>47</td><td>28</td><td>8</td><td>1410</td><td>1368</td><td>1379</td><td>1814</td><td>370</td><td>358</td><td>352</td></tr><tr><td>A/Indiana/01/20/11 (H1)</td><td>1302</td><td>1295</td><td>1292</td><td>1375</td><td>1414</td><td>1410</td><td>1</td><td>81</td><td>72</td><td>140</td><td>1464</td><td>1437</td><td>1431</td></tr><tr><td>A/Vietnam/15/2006 (H1)</td><td>1300</td><td>1298</td><td>1297</td><td>1317</td><td>1406</td><td>1368</td><td>81</td><td>3</td><td>17</td><td>71</td><td>1444</td><td>1429</td><td>1404</td></tr><tr><td>A/Wisconsin/15/2009 (H1)</td><td>1329</td><td>1322</td><td>1320</td><td>1350</td><td>1416</td><td>1379</td><td>75</td><td>19</td><td>9</td><td>68</td><td>1431</td><td>1427</td><td>1401</td></tr><tr><td>A/Vietnam/12/03/2004 (H1)</td><td>1283</td><td>1287</td><td>1285</td><td>1312</td><td>1384</td><td>1348</td><td>72</td><td>17</td><td>4</td><td>66</td><td>1401</td><td>1397</td><td>1371</td></tr><tr><td>A/Vietnam/21/02/2009 (H1)</td><td>1750</td><td>1747</td><td>1734</td><td>1734</td><td>1853</td><td>1814</td><td>140</td><td>71</td><td>66</td><td>58</td><td>1920</td><td>1923</td><td>1882</td></tr><tr><td>A/India/NV/2006 (H1)</td><td>355</td><td>354</td><td>354</td><td>368</td><td>372</td><td>364</td><td>1478</td><td>1435</td><td>1390</td><td>1902</td><td>21</td><td>24</td><td>29</td></tr><tr><td>A/Ahmad/01/2005 (H1)</td><td>370</td><td>369</td><td>364</td><td>376</td><td>370</td><td>370</td><td>1464</td><td>1431</td><td>1401</td><td>1920</td><td>3</td><td>23</td><td>34</td></tr><tr><td>A/Hubei/1/2010 (H1)</td><td>353</td><td>352</td><td>349</td><td>360</td><td>363</td><td>358</td><td>1437</td><td>1429</td><td>1397</td><td>1923</td><td>24</td><td>5</td><td>31</td></tr><tr><td></td><td>352</td><td>351</td><td>350</td><td>355</td><td>358</td><td>352</td><td>1431</td><td>1404</td><td>1371</td><td>1882</td><td>34</td><td>31</td><td>2</td></tr></tbody></table>		A/California/07/2009 (H1)	A/Mexico/4/08/2009 (H1)	A/Utah/20/2009 (H1)	A/Brisbane/59/2007 (H1)	A/USSR/90/1977 (H1)	A/Taiwan/1/1986 (H1)	A/Indiana/01/20/11 (H1)	A/Vietnam/15/2006 (H1)	A/Vietnam/21/02/2009 (H1)	A/Vietnam/12/03/2004 (H1)	A/India/NV/2006 (H1)	A/Ahmad/01/2005 (H1)	A/Hubei/1/2010 (H1)	A/California/07/2009 (H1)	10	13	19	148	142	138	1302	1300	1293	1750	370	353	352	A/Mexico/4/08/2009 (H1)	13	11	17	153	144	140	1295	1298	1287	1747	369	352	351	A/Utah/20/2009 (H1)	19	17	12	160	149	147	1292	1297	1285	1734	364	349	350	A/Brisbane/59/2007 (H1)	148	153	160	10	65	47	1375	1317	1312	1734	376	360	355	A/USSR/90/1977 (H1)	142	144	149	65	8	28	1414	1406	1416	1853	370	363	358	A/Taiwan/1/1986 (H1)	138	140	147	47	28	8	1410	1368	1379	1814	370	358	352	A/Indiana/01/20/11 (H1)	1302	1295	1292	1375	1414	1410	1	81	72	140	1464	1437	1431	A/Vietnam/15/2006 (H1)	1300	1298	1297	1317	1406	1368	81	3	17	71	1444	1429	1404	A/Wisconsin/15/2009 (H1)	1329	1322	1320	1350	1416	1379	75	19	9	68	1431	1427	1401	A/Vietnam/12/03/2004 (H1)	1283	1287	1285	1312	1384	1348	72	17	4	66	1401	1397	1371	A/Vietnam/21/02/2009 (H1)	1750	1747	1734	1734	1853	1814	140	71	66	58	1920	1923	1882	A/India/NV/2006 (H1)	355	354	354	368	372	364	1478	1435	1390	1902	21	24	29	A/Ahmad/01/2005 (H1)	370	369	364	376	370	370	1464	1431	1401	1920	3	23	34	A/Hubei/1/2010 (H1)	353	352	349	360	363	358	1437	1429	1397	1923	24	5	31		352	351	350	355	358	352	1431	1404	1371	1882	34	31	2
	A/California/07/2009 (H1)	A/Mexico/4/08/2009 (H1)	A/Utah/20/2009 (H1)	A/Brisbane/59/2007 (H1)	A/USSR/90/1977 (H1)	A/Taiwan/1/1986 (H1)	A/Indiana/01/20/11 (H1)	A/Vietnam/15/2006 (H1)	A/Vietnam/21/02/2009 (H1)	A/Vietnam/12/03/2004 (H1)	A/India/NV/2006 (H1)	A/Ahmad/01/2005 (H1)	A/Hubei/1/2010 (H1)																																																																																																																																																																																																																			
A/California/07/2009 (H1)	10	13	19	148	142	138	1302	1300	1293	1750	370	353	352																																																																																																																																																																																																																			
A/Mexico/4/08/2009 (H1)	13	11	17	153	144	140	1295	1298	1287	1747	369	352	351																																																																																																																																																																																																																			
A/Utah/20/2009 (H1)	19	17	12	160	149	147	1292	1297	1285	1734	364	349	350																																																																																																																																																																																																																			
A/Brisbane/59/2007 (H1)	148	153	160	10	65	47	1375	1317	1312	1734	376	360	355																																																																																																																																																																																																																			
A/USSR/90/1977 (H1)	142	144	149	65	8	28	1414	1406	1416	1853	370	363	358																																																																																																																																																																																																																			
A/Taiwan/1/1986 (H1)	138	140	147	47	28	8	1410	1368	1379	1814	370	358	352																																																																																																																																																																																																																			
A/Indiana/01/20/11 (H1)	1302	1295	1292	1375	1414	1410	1	81	72	140	1464	1437	1431																																																																																																																																																																																																																			
A/Vietnam/15/2006 (H1)	1300	1298	1297	1317	1406	1368	81	3	17	71	1444	1429	1404																																																																																																																																																																																																																			
A/Wisconsin/15/2009 (H1)	1329	1322	1320	1350	1416	1379	75	19	9	68	1431	1427	1401																																																																																																																																																																																																																			
A/Vietnam/12/03/2004 (H1)	1283	1287	1285	1312	1384	1348	72	17	4	66	1401	1397	1371																																																																																																																																																																																																																			
A/Vietnam/21/02/2009 (H1)	1750	1747	1734	1734	1853	1814	140	71	66	58	1920	1923	1882																																																																																																																																																																																																																			
A/India/NV/2006 (H1)	355	354	354	368	372	364	1478	1435	1390	1902	21	24	29																																																																																																																																																																																																																			
A/Ahmad/01/2005 (H1)	370	369	364	376	370	370	1464	1431	1401	1920	3	23	34																																																																																																																																																																																																																			
A/Hubei/1/2010 (H1)	353	352	349	360	363	358	1437	1429	1397	1923	24	5	31																																																																																																																																																																																																																			
	352	351	350	355	358	352	1431	1404	1371	1882	34	31	2																																																																																																																																																																																																																			

Table 6

MDS computational efficiency (seconds).

```
In[270]:= TableFigure[times, {"Airport", "Flu Antibody", "Flu Sequence"},
                           {"SVD", "cMDS", "mMDS(FindMinimum)", "mMDS(NMinimize)"},
                           12]
```

		Airport	Flu Antibody	Flu Sequence
<i>Out[270]=</i>	SVD	0.002294	0.000580	0.000619
	cMDS	0.001568	0.000429	0.000679
	mMDS(FindMinimum)	1.238080	0.121800	0.061153
	mMDS(NMinimize)	13.598350	0.962116	0.920443