



**HAL**  
open science

## Multidimensional scaling with very large datasets

Emmanuel Paradis

► **To cite this version:**

Emmanuel Paradis. Multidimensional scaling with very large datasets. Journal of Computational and Graphical Statistics, 2018, pp.1 - 5. 10.1080/10618600.2018.1470001 . ird-01920130

**HAL Id: ird-01920130**

**<https://ird.hal.science/ird-01920130>**

Submitted on 11 Jul 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Multidimensional scaling with very large data sets

Emmanuel Paradis

April 6, 2018

*ISEM, IRD, Univ. Montpellier, CNRS, EPHE, Montpellier, France*

E-mail: Emmanuel.Paradis@ird.fr

**Abstract:** Multidimensional scaling has a wide range of applications when observations are not continuous but it is possible to define a distance (or dissimilarity) among  
3 them. However, standard implementations are limited when analyzing very large data sets  
because they rely on eigendecomposition of the full distance matrix and require very long  
computing times and large quantities of memory. Here, a new approach is developed based  
6 on projection of the observations in a space defined by a subset of the full data set. The  
method is easily implemented. A simulation study showed that its performance are satis-  
factory in different situations and can be run in a short time when the standard method  
9 takes a very long time or cannot be run because of memory requirements.

**Keywords:** dimension reduction, distance data, projection method, random matrices

# 1 Introduction

12 Multidimensional scaling (MDS) aims to find a set of coordinates in one or several dimen-  
sions, so that the distances derived from these coordinates are the closest possible to the  
observed distances. By contrast to principal component analysis (PCA), MDS (also called  
15 principal coordinates analysis, PCoA) does not require a set of original coordinates. This  
is a very attractive feature since a distance can naturally be defined for many types of  
data whereas these data do not easily define a system of coordinates (e.g., DNA sequences,  
18 graphs, psychological profiles). A search of the string “multidimensional scaling” in the  
Web of Science returned 7313 hits (accessed 2017-07-31) with the highest proportions of  
publications in the fields of environmental sciences & ecology (15.9%), psychology (14.9%),  
21 and computer science (14.7%).

MDS is basically done through a decomposition of the symmetric distance matrix among  
observations. This matrix as thus has many rows and columns than there are observations,  
24 and its decomposition with standard eigenvalue analysis algorithm can be a limiting factor  
when this number is large. In a context where many scientific fields collect large amounts  
of data, this is clearly a severe limitation. Some examples include genomic analyses from  
27 high throughput sequencing technologies which typically handle millions of DNA sequences,  
or environmental data from high-resolution remote sensing which represent variables for  
millions of localities on the Earth’s surface.

30 In this paper, I present an approach to avoid the limitations of the standard procedure  
of MDS. In the next section, I present the different computational procedures considered in  
this paper. Section 3 reports the results from a simulation study. The last section discusses  
33 these results and presents some perspectives.

## 2 Computational procedures

Section 2.1 below describes the standard MDS procedure; Section 2.2 explains how this  
36 procedure can be extended with random matrix algorithms. Sections 2.3 and 2.4 introduce  
new procedures.

## 2.1 Standard multidimensional scaling

39 Let us denote  $n$  the number of observations, and  $\Delta$  the  $n \times n$  symmetric matrix of distances with  $\delta_{ij}$  ( $= \delta_{ji}$ ) being the distance between observations  $i$  and  $j$ . MDS proceeds by doing an eigendecomposition of the doubly-centred distance matrix:

$$-\frac{1}{2}J\Delta^2J, \quad (1)$$

42 with  $J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ . The matrix of coordinates  $Z$  are calculated with:

$$Z = V\Lambda^{1/2},$$

where  $V$  is an  $n \times k$  matrix with the first  $k$  eigenvectors extracted from (1), and  $\Lambda$  is a diagonal  $k \times k$  matrix with the first  $k$  eigenvalues  $(\lambda_1, \dots, \lambda_k)$ .  $Z$  has therefore  $n$  rows and  
45  $k$  columns. The value of  $k$  is the number of dimensions of the projected space and is chosen by the investigator, usually depending on the values of  $\lambda$ .

This procedure involves manipulating matrices with  $n^2$  elements which is very expensive  
48 in terms of memory requirements and computing times when  $n$  is large.

## 2.2 Eigendecomposition with random matrices

Halko et al. (2011) presented several algorithms to decompose very large matrices. These  
51 algorithms are based on random matrices and can extract several eigenvalues in a few seconds while it would take several hours to perform the same operation with classical eigendecomposition. The eigenvalues are extracted sequentially, by contrast to the standard  
54 MDS where all eigenvalues are calculated at once, even if only some of them are used to calculate the coordinates in  $Z$ . Therefore, random matrix algorithms make possible to decompose very large matrices which could not be analyzed with the standard approach  
57 because of memory limitations. Here, we use the implementation from Abraham & Inouye and their package `flashpcaR` (Abraham & Inouye 2014).

## 2.3 1-D projection

60 The principle of this method is quite simple. In a first step,  $m$  observations are selected ( $m < n$ ) and a standard MDS is performed on them setting  $k = 1$ . We denote as  $z_i$  the

coordinate of the  $i$ th observation ( $i = 1, \dots, m$ ). In a second step, for each observation  $j$   
 63 not among the  $m$  selected in step 1, the coordinate  $z_j$  is found by minimizing:

$$f = \sum_{i=1}^m (\delta_{ij} - d_{ij})^2, \quad (2)$$

with  $d_{ij} = |z_i - z_j|$ . We may write the latter as:

$$d_{ij} = \gamma_i(z_i - z_j) \quad \gamma_i = \begin{cases} +1 & z_j < z_i \\ -1 & z_j \geq z_i, \end{cases}$$

A simple algebraic development leads to:

$$f = \sum_{i=1}^m \delta_{ij}^2 - 2 \sum_{i=1}^m (\delta_{ij} \gamma_i z_i - \delta_{ij} \gamma_i z_j) + \sum_{i=1}^m z_i^2 - 2z_j \sum_{i=1}^m z_i + mz_j^2.$$

66 We can now write the partial derivative of  $f$  with respect to  $z_j$ :

$$\frac{\partial f}{\partial z_j} = 2 \sum_{i=1}^m \delta_{ij} \gamma_i - 2 \sum_{i=1}^m z_i + 2mz_j.$$

Writing  $\bar{z} = \frac{1}{m} \sum_{i=1}^m z_i$  and solving the last expression leads us to find the value  $z_j$  that  
 minimizes  $f$ :

$$\frac{1}{m} \sum_{i=1}^m \delta_{ij} \gamma_i - \bar{z} + z_j = 0.$$

69 Given that there are only  $m$  distances  $\delta_{ij}$  to calculate, and that  $\bar{z}$  is calculated only once,  
 this equation can be solved numerically relatively easily. During the first step,  $m(m-1)/2$   
 distances are computed for the classical MDS. Thus, the memory requirements of this  
 72 method are very small. Overall, instead of calculating  $n(n-1)/2$  distances for a full MDS,  
 this method requires to calculate  $nm - m(m+1)/2$  distances.

The choice of the  $m$  observations is important for this projection procedure to work  
 75 as best as possible. They should represent as much as possible the distribution of the  $n$   
 observations. This is not straightforward since the projection of these  $n$  points in the MDS  
 space is not known a priori. In order to solve this issue, the following algorithm selects  $m$   
 78 observations so their distances are representative of the whole set of distances:

1. Select one observation  $i$  at random among the  $n$  ones; store  $i$ .
2. Compute the distances  $\delta_{ij}$  with  $j$  among the values not yet stored.

- 81 3. Find  $j'$  so that  $\delta_{ij'}$  is max of the distances calculated at step 2; store  $j'$ .
4. Compute the distances  $\delta_{j'i}$  with  $j$  among the values not yet stored.
5. Select  $i$  so that  $\delta_{j'i}$  is median of the distances calculated at step 4; store  $i$ .
- 84 6. Repeat steps 2–5 until  $m$  values are stored.

This algorithm may be used whatever the number of dimensions  $k$ . There could be some variants of this algorithm: for instance, instead of the median in step 5, one could select  
87 a random distance  $\delta_{j'i}$  so that the observations will be represented in proportion of their relative frequencies (somehow similar to an MCMC procedure).

## 2.4 Higher dimension projection

90 The method developed above can be generalized to more than one dimension in a straightforward way. The distances  $d_{ij}$  in (2) would then be calculated as Euclidean distances in the projection space:

$$d_{ij} = \sqrt{\sum_{l=1}^k (z_{il} - z_{jl})^2}$$

93 In this case, the partial derivatives can be derived but are too complicated to be useful, so it is more efficient to rely on a standard minimization procedure to minimize (2). Two procedures were tested here: the classical BFGS method (Broyden 1970, Fletcher 1970,  
96 Goldfarb 1970, Shanno 1970) and the PORT routine (Gay 1990).

## 3 Simulation study

A simulation study was conducted to answer two questions: What are the computing times  
99 of the different procedures described above? and, What are their respective accuracy? The data were simulated from a standard normal distribution with one or two dimensions and with different samples sizes  $n$  (1000, 5000, 10,000, and 50,000). The projection procedures  
102 were performed with  $m = 100$  points chosen randomly. To assess the accuracy of the results, two quantities were calculated. First, the classical stress was calculated with Kruskal's formula (Kruskal 1964):

$$S = \sqrt{\frac{\sum_{i,j}^n (\delta_{ij} - d_{ij})^2}{\sum_{i,j}^n \delta_{ij}^2}}.$$

105 This quantity varies between 0 (complete mismatch of the distances) and 1 (perfect match).  
 Second, a measure of the accuracy of the inferred distances was calculated with:

$$S' = \sum_{i,j}^n \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}}.$$

This second quantity is similar to Sammon’s criterion (Sammon 1969) and gives more  
 108 emphasis on the precision of each distance whereas Kruskal’s stress puts emphasis on the  
 overall precision of the distances. In addition, in order to assess whether the observed  
 results were better than simply randomly positioning the points in the MDS space, both  
 111 quantities were calculated after randomizing the simulated data.

The simulations were run with  $k = 1$  and  $k = 2$ . In the second case, the two variables  
 were either independent or with a correlation of 0.7. Additionally, skewed distributions  
 114 were generated with  $k = 1$  in order to simulate aggregation of points. Two cases were  
 considered: an exponential distribution with rate equal to one, and a mixture of two  
 normal distributions with means  $-5$  and  $5$  and proportions 0.9 and 0.1, respectively. In  
 117 these two cases, the analyses were performed with  $m = 100$  observations selected randomly  
 like previously, and using the algorithm described above. All simulations were replicated  
 100 times and run on a computer equipped with a duo-core, 2.1 GHz processor and 16 GB  
 120 of RAM, and running Ubuntu 16.04. All computations were implemented in R version 3.4.1  
 (R Core Team 2017); the code is available as supplementary material with this article.

### 3.1 1-D MDS

123 With a moderate sample size  $n = 1000$ , the three methods considered here completed in less  
 than one second (Table 1). However, with  $n = 10,000$ , the standard MDS took almost four  
 minutes while the same procedure with a random decomposition took slightly more than  
 126 eleven seconds, and the projection method took less than one second. Most importantly,  
 the computing times of the latter appeared to be linearly related with  $n$  whereas the two  
 others seemed to run proportionally to  $n^3$  for the standard MDS, or to  $n^2$  for the random

Table 1: Computing times of different 1-D MDS methods (in seconds).

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	0.33	26.47	220.94	–
Random eigendecomposition	0.09	1.09	11.66	–
1-D Projection ( $m = 100$ )	0.11	0.41	0.79	3.91

Table 2: Stress ( $S$ ) from different 1-D MDS methods.

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	$1.6 \times 10^{-15}$	$2.7 \times 10^{-15}$	$3.6 \times 10^{-15}$	–
Random eigendecomposition	$1.7 \times 10^{-2}$	$7.6 \times 10^{-3}$	$6.9 \times 10^{-3}$	–
1-D Projection ( $m = 100$ )	$9.1 \times 10^{-6}$	$9.5 \times 10^{-6}$	$9.3 \times 10^{-6}$	$9.7 \times 10^{-6}$
“random”	$9.5 \times 10^{-1}$	$8.5 \times 10^{-1}$	$8.5 \times 10^{-1}$	$8.5 \times 10^{-1}$

129 decomposition. The computing time of the two decomposition-based methods was not  
assessed with  $n = 50,000$ .

132 Unsurprisingly, the standard MDS performed the best considering either the stress  $S$   
(Table 2) or the accuracy of the inferred distances  $S'$  (Table 3). The projection method  
performed the second best and better than the random decomposition method. All methods  
performed better than randomly projecting the points.

Table 3: Accuracy of inferred distances ( $S'$ ) from different 1-D MDS methods.

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	$6.7 \times 10^{-24}$	$1.3 \times 10^{-21}$	$9.2 \times 10^{-21}$	–
Random eigendecomposition	$2.8 \times 10^2$	$1.4 \times 10^3$	$3.7 \times 10^3$	–
1-D Projection ( $m = 100$ )	$3.8 \times 10^{-4}$	$9.7 \times 10^{-3}$	$3.8 \times 10^{-2}$	$9.8 \times 10^{-1}$
“random”	$8.7 \times 10^6$	$4.3 \times 10^8$	$1.1 \times 10^9$	$3.4 \times 10^{10}$

Table 4: Computing times of different 2-D MDS methods (in seconds).

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	0.32	26.32	221.60	–
Random eigendecomposition	0.07	1.13	12.35	–
2-D Projection ( $m = 100$ ) with BFGS	0.59	3.01	6.06	33.58
with PORT	0.33	1.58	3.20	17.62

Table 5: Stress ( $S$ ) from different 2-D MDS methods.

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	$1.6 \times 10^{-15}$	$2.4 \times 10^{-15}$	$4.5 \times 10^{-15}$	–
Random eigendecomposition	$2.0 \times 10^{-2}$	$9.2 \times 10^{-3}$	$6.1 \times 10^{-3}$	–
2-D Projection ( $m = 100$ ) with BFGS	$5.4 \times 10^{-8}$	$5.7 \times 10^{-8}$	$5.7 \times 10^{-8}$	$5.7 \times 10^{-8}$
with PORT	$4.5 \times 10^{-10}$	$4.9 \times 10^{-10}$	$4.5 \times 10^{-10}$	$4.7 \times 10^{-10}$
“random”	$6.6 \times 10^{-1}$	$6.5 \times 10^{-1}$	$6.5 \times 10^{-1}$	$6.5 \times 10^{-1}$

### 135 3.2 2-D MDS

The two methods based on matrix decomposition showed similar computing times than in one dimension (Table 4). On the other hand, the projection method was slower but its  
138 computing times still scaled proportionally to  $n$ . The PORT-based projection method was almost twice faster than the BFGS-based one.

With two independent variables, the accuracy of the projection methods were less than  
141 the standard MDS but these methods appeared still accurate (Tables 5 and 6). The PORT-based variant was more accurate than the BFGS-based one. By contrast, the random decomposition method performed poorly. When the two variables were correlated, the  
144 performance of the standard MDS and the projection methods were similar to the previous situation; however, the performance of the random decomposition method deteriorated considerably and were only slightly better than randomly positioning the data (Tables 7  
147 and 8).

Table 6: Accuracy of inferred distances ( $S'$ ) from different 2-D MDS methods.

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	$3.1 \times 10^{-24}$	$2.0 \times 10^{-22}$	$4.2 \times 10^{-21}$	–
Random eigendecomposition	$4.3 \times 10^2$	$2.2 \times 10^3$	$3.5 \times 10^3$	–
2-D Projection ( $m = 100$ ) with BFGS	$5.1 \times 10^{-9}$	$1.2 \times 10^{-7}$	$5.0 \times 10^{-7}$	$1.2 \times 10^{-5}$
with PORT	$4.0 \times 10^{-13}$	$1.1 \times 10^{-11}$	$4.0 \times 10^{-11}$	$1.1 \times 10^{-9}$
“random”	$8.8 \times 10^5$	$2.2 \times 10^7$	$8.9 \times 10^7$	$2.2 \times 10^9$

Table 7: Stress ( $S$ ) from different 2-D MDS methods with correlated variables.

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	$1.4 \times 10^{-15}$	$2.2 \times 10^{-15}$	$2.7 \times 10^{-15}$	–
Random eigendecomposition	$5.5 \times 10^{-1}$	$5.5 \times 10^{-1}$	$5.5 \times 10^{-1}$	–
2-D Projection ( $m = 100$ ) with BFGS	$5.4 \times 10^{-8}$	$5.9 \times 10^{-8}$	$5.5 \times 10^{-8}$	$6.0 \times 10^{-8}$
with PORT	$1.9 \times 10^{-10}$	$2.1 \times 10^{-10}$	$2.0 \times 10^{-10}$	$2.1 \times 10^{-10}$
“random”	$7.4 \times 10^{-1}$	$7.4 \times 10^{-1}$	$7.5 \times 10^{-1}$	$7.4 \times 10^{-1}$

Table 8: Accuracy of inferred distances ( $S'$ ) from different 2-D MDS methods with correlated variables.

Method	$n$			
	1000	5000	10,000	50,000
Standard MDS	$3.3 \times 10^{-24}$	$2.7 \times 10^{-22}$	$1.6 \times 10^{-21}$	–
Random eigendecomposition	$2.9 \times 10^5$	$7.0 \times 10^6$	$2.9 \times 10^7$	–
2-D Projection ( $m = 100$ ) with BFGS	$8.9 \times 10^{-9}$	$2.5 \times 10^{-7}$	$7.7 \times 10^{-7}$	$2.3 \times 10^{-5}$
with PORT	$1.1 \times 10^{-14}$	$2.8 \times 10^{-12}$	$1.0 \times 10^{-11}$	$2.8 \times 10^{-10}$
“random”	$1.5 \times 10^6$	$3.7 \times 10^7$	$1.5 \times 10^8$	$3.7 \times 10^9$

Table 9: Results of simulations with an exponential variable. CT: computing times (in seconds). Sampling  $m = 100$  observations with the algorithm presented in this paper (A) or uniformly (U).

		$n$			
Sampling		1000	5000	10,000	50,000
CT	A	0.06	0.35	0.72	3.86
	U	0.06	0.36	0.73	3.93
$S$	A	$8.3 \times 10^{-6}$	$8.6 \times 10^{-6}$	$8.9 \times 10^{-6}$	$8.6 \times 10^{-6}$
	U	$1.0 \times 10^{-5}$	$1.1 \times 10^{-5}$	$1.1 \times 10^{-5}$	$1.1 \times 10^{-5}$
$S'$	A	$5.6 \times 10^{-4}$	$1.4 \times 10^{-2}$	$6.0 \times 10^{-2}$	1.33
	U	$8.7 \times 10^{-4}$	$2.1 \times 10^{-2}$	$8.3 \times 10^{-2}$	2.16

### 3.3 Skewed Distributions

For the two methods based on matrix decomposition, the results with the skewed distributions were very similar to the previous ones, and are thus not reported here. For the 1-D projection method, the analyses were performed with a uniform random sample and using the algorithm described above. Both algorithms resulted in similar computing times; however, the uniform sampling resulted in decreased accuracy while the above algorithm yielded performance comparable to the previous ones with non-aggregated data. This difference in accuracy was small for the data simulated from an exponential distribution (Table 9) whereas it was important for the mixture of normal variables where the skewness of the data was much more pronounced (Table 10).

## 4 Discussion

This article presents a method to perform MDS on very large data sets in short times and with small memory requirements. The objective of the present study was to develop a method easily implemented and generally applicable. One initial motivation was to avoid the need to perform a matrix decomposition of the full distance matrix as used in standard MDS.

Two approaches were considered in this study: the first one used matrix decomposition algorithms based on random matrices, and the second one used a projection algorithm

Table 10: Results of simulations with a mixture of two random variables. CT: computing times (in seconds). Sampling  $m = 100$  observations with the algorithm presented in this paper (A) or uniformly (U).

		$n$			
Sampling		1000	5000	10,000	50,000
CT	A	0.06	0.35	0.70	3.68
	U	0.07	0.36	0.73	3.85
$S$	A	$2.9 \times 10^{-6}$	$3.0 \times 10^{-6}$	$3.0 \times 10^{-6}$	$2.9 \times 10^{-6}$
	U	$4.1 \times 10^{-2}$	$4.3 \times 10^{-2}$	$4.2 \times 10^{-2}$	$4.2 \times 10^{-2}$
$S'$	A	$3.0 \times 10^{-4}$	$7.4 \times 10^{-3}$	$2.9 \times 10^{-2}$	$7.3 \times 10^{-1}$
	U	$9.8 \times 10^3$	$2.1 \times 10^5$	$7.8 \times 10^5$	$1.8 \times 10^7$

based on a subset of points. The first approach did not appear as a viable solution to handle large data sets: it was too slow for sample sizes larger than 10,000 and was very inaccurate in some situations. This method performed poorly with correlated variables, which was an unexpected result. It is unclear whether this a pathological specific case or a more general problem with random decomposition of distance matrices. Further tests will be needed to clarify this point.

One issue not treated in depth in the present work is how to select the number of dimensions ( $k$ ). In standard MDS, this value is selected depending on the eigenvalues extracted from the decomposition of the distance matrix. Typically, in practical applications of MDS two dimensions are selected in order to provide an interpretable graphical display. With the projection method proposed in this paper, since the number of dimensions determines the algorithm used, this number may be selected with respect to the eigenvalues of the standard MDS done on the subset of size  $m$ .

Another issue not explored here is the choice of the size of the subset ( $m$ ). It was found that a value  $m = 100$  is appropriate in the situations considered here: it makes possible to perform the projection easily since a larger value would make this procedure slower and more complicated. An interesting result was the good performance of the selection algorithm presented in this paper, particularly if the data were aggregated. This also deserves further study.

The present approach can have a wide range of practical applications. Many applied

186 researchers need to analyze increasingly larger data sets, such as in ecological habitat  
modeling based on remote sensing (Hansen et al. 2008) or in genomic analysis (Erlich  
2015). It will thus be interesting to see how the present method behaves and performs in  
189 practical applications.

## Acknowledgments

I am grateful to an anonymous reviewer for helpful comments on a previous version of this  
192 paper. This is publication ISEM 2017-201.

## SUPPLEMENTARY MATERIAL

**functions\_MDS.R:** R functions used to perform the methods described in this paper.

195 **sim.R:** R functions used to perform the simulations reported in this paper.

## References

- Abraham, G. & Inouye, M. (2014), ‘Fast principal component analysis of large-scale  
198 genome-wide data’, *PLoS ONE* **9**(4), e93766.
- Broyden, C. G. (1970), ‘The convergence of a class of double-rank minimization algorithms  
1. General considerations’, *Journal of the Institute of Mathematics and its Applications*  
201 **6**(1), 76–90.
- Erlich, Y. (2015), ‘A vision for ubiquitous sequencing’, *Genome Research* **25**(10), 1411–  
1416.
- 204 Fletcher, R. (1970), ‘A new approach to variable metric algorithms’, *Computer Journal*  
**13**(3), 317–322.
- Gay, D. M. (1990), Usage summary for selected optimization routines, Technical Report  
207 Computing Science Technical Report No. 153, AT&T Bell Laboratories, Murray Hill, NJ  
07974, USA.  
**URL:** <http://netlib.bell-labs.com/netlib/port/>
- 210 Goldfarb, D. (1970), ‘A family of variable metric updates derived by variational means’,  
*Mathematics of Computation* **24**, 23–26.
- Halko, N., Martinsson, P. G. & Tropp, J. A. (2011), ‘Finding structure with randomness:

- 213 probabilistic algorithms for constructing approximate matrix decompositions’, *SIAM Review* **53**(2), 217–288.
- Hansen, M. C., Stehman, S. V., Potapov, P. V., Loveland, T. R., Townshend, J. R. G.,  
216 DeFries, R. S., Pittman, K. W., Arunarwati, B., Stolle, F., Steininger, M. K., Carroll,  
M. & DiMiceli, C. (2008), ‘Humid tropical forest clearing from 2000 to 2005 quantified  
by using multitemporal and multiresolution remotely sensed data’, *Proceedings of the*  
219 *National Academy of Sciences USA* **105**(27), 9439–9444.
- Kruskal, J. B. (1964), ‘Multidimensional scaling by optimizing goodness of fit to a nonmetric  
hypothesis’, *Psychometrika* **29**(1), 1–27.
- 222 R Core Team (2017), *R: A Language and Environment for Statistical Computing*, R Found-  
ation for Statistical Computing, Vienna, Austria.  
**URL:** <http://www.R-project.org>
- 225 Sammon, Jr, J. W. (1969), ‘A nonlinear mapping for data structure analysis’, *IEEE Trans-*  
*actions on Computers* **C-18**(5), 401–409.
- Shanno, D. F. (1970), ‘Conditioning of quasi-Newton methods for function minimization’,  
228 *Mathematics of Computation* **24**, 647–656.