BinMat: a molecular genetics tool for processing binary data obtained from fragment analysis in R

CJM van Steenderen[1]

Department of Zoology and Entomology, Centre for Biological Control (CBC), Rhodes University, Grahamstown, 6140, South Africa

[1] Center for Biological Control, Rhodes University, Grahamstown, 6140, South Africa

Corresponding author: vsteenderen@gmail.com

Running title: Fragment analysis of binary data in R

Abstract

Processing and visualising trends in the binary data obtained from fragment analysis methods in molecular biology can be a time-consuming, and often cumbersome process. Scoring and processing binary data (from methods such as AFLPs, ISSRs, and RFLPs) entails complex workflows that require a high level of computational and/or bioinformatic skills. The application presented here (BinMat) is a free, open-source, and user-friendly R Shiny program that automates the analysis pipeline on one platform. BinMat is presented as a Graphical User Interface (GUI) via the Shiny package in R that is available online across different operating systems. It is also available as an R package. BinMat consolidates replicate sample pairs in a dataset into consensus reads, produces summary statistics, and allows the user to visualise their data as ordination plots and clustering trees without having to use multiple software programs and input files, or rely on previous programming experience.

Keywords

AFLP, binary data scoring, bioinformatics, GUI, ISSR

1 Introduction

Fragment analysis is a broad term used in molecular biology which encompasses the processes by which fragments of DNA are separated by size in order to generate characteristic band-profiles. Bands are detected and scored through either the traditional method of viewing them on polyacrylamide gels (Bassam *et al.*, 1991), or through the use of fluorescent markers (such as FAM™ or ROX®) that tag fragments so that they can be detected by capillary electrophoresis (Dresler-Nurmi *et al.*, 2000; AppliedBiosystems, 2014). There are a number of techniques associated with fragment analysis, including AFLP (Amplified Fragment Length Polymorphism) (Vos *et al.*, 1995), RAPD (Random Amplified Polymorphic DNA) (Koeleman *et al.*, 1998), and ISSR (Inter-Simple Sequence Repeats) (Wolfe and Liston, 1998; Abbot, 2001). Fragment analysis offers a wide range of applications, such as DNA fingerprinting, SNP (single nucleotide polymorphism) genotyping, and microsatellite profiling (AppliedBiosystems, 2014), which are used across a broad range of disciplines.

Processing and analysing the binary data obtained from fragment analysis methods can quickly become challenging due to the large size of datasets and the time required to organise and format them to suit the needs of different programs used down the analysis-pipeline. Common practice is to independently replicate each Polymerase Chain Reaction (PCR) sample in order to consolidate the output into one

consensus read per individual (see for example Taylor *et al.* (2011) and Sutton *et al.* (2017)). The term 'consolidate', as used here, refers to the process of checking the binary value scored at each locus position across every replicate pair, and creating one representative consensus output for that sample. For example, if both replicates show the presence of a band at a particular locus, a '1' is recorded as 'present' at that locus. If a band was absent in both replicates, a '0' is recorded. If one replicate shows the presence of a band, but the other shows an abscence, a '?' is recorded to denote an ambiguous read.

Manually consolidating the replicate pairs of large binary matrices in this way is not only impractical, but it also lends itself to human error. Even after fragments have been scored and processed, the downstream analyses of these data are complex. For example, a number of different programs are often required for different analyses; each of which require a different input file format. This requires a certain level of computational and/or bioinformatic skills, and can be both difficult and time-consuming, and can result in further potential errors when changing between file formats.

The R programming language (R Core Team, 2019) is becoming an increasingly popular means of analysing genetic data (Paradis *et al.*, 2004; Schliep, 2011; Archer *et al.*, 2017), as it can read in multiple file formats and perform a number of analyses all on one platform. Packages in R can, however, often be challenging to utilise for newcomers to programming. The development of GUI (Graphical User Interface) software can address this by collating multiple processing tools into one place, and make complex computational tasks more accessible to researchers (see for example Reyes *et al.* (2019)).

Here I present BinMat, an R package and R Shiny application that automates the analysis of fragment data. Named 'BinMat', from '**Bin**ary **Mat**rix', the application offers researchers a user-friendly, open-source platform that does not require multiple programs and file input formats (Fig. 1). Moreover, a GUI was developed to make data processing easier and more accessible. BinMat is available on three platforms; namely the shinyapps.io server, GitHub, and as an R package on CRAN. The following sections detail the functionality of BinMat, how its output compares to PAST (Hammer *et al.*, 2001) and SplitsTree (Huson, 1998) (which are standalone software typically used to analyse genetic data), and how it can be accessed.

2 Shiny Graphical User Interface (GUI)

2.1 File input

BinMat reads in binary data that has already been processed from raw electropherograms using programs such as GeneMarker (SoftGenetics®) and RawGeno (Arrigo *et al.*, 2012). This needs to be uploaded as a comma-separated values (CSV (Comma delimited)) file in the format shown in Table 1. Column

headings are required, but are not limited to the exact labels shown in the example. If the data consists of replicate pairs, these need to be organised so that they appear consecutively, with a unique name for each sample. It is important to check the data to ensure that there are no single samples without their replicate. When the 'Consolidate matrix' button is clicked, each replicate pair in the dataset is consolidated into a consensus output.

Table 2 shows the output if the data in Table 1 was used as input. The resulting consolidated binary matrix can be downloaded as a CSV file using the 'Download Matrix' button once the message 'COMPLETE. READY FOR DOWNLOAD' appears on the screen. The 'Check my data for unwanted values' button checks the data for any values in the dataset other than a '1', '0', or '?', and returns the column and row index for the unwanted character/s.

2.2 Data analysis and visualisation

Once the data has been consolidated, the user can view and download information in the 'SUMMARY' tab at the top of the window; showing the average number of peaks (± standard deviation (sd)), the maximum and minimum number of peaks, and the total number of loci. The 'ERROR RATES' tab shows the Euclidean (EE) (± sd) and Jaccard (JE) (± sd) error rates. See Bonin *et al.* (2004), Pompanon *et al.* (2005), and Holland *et al.* (2008) for detailed reviews regarding error rates and their calculation.

The 'Remove samples with a jaccard error greater than:' button removes samples with a Jaccard error (ranging from 0 to 1) greater than or equal to a specified value. This can give the user an idea of how filtering their data can affect overall error rates. The default value is set at zero.

Clustering methods, such as the UPGMA (Unweighted Pair Group Method with Arithmetic Mean) and neighbour-joining, are frequently used in the analyses of fragment data to create dendrograms (see for example Van Eldere *et al.* (1999); Ticknor *et al.* (2001); Liu *et al.* (2009); Timm *et al.* (2010)). Additionally, ordination methods such as those offered by non-metric multidimensional scaling (nMDS) plots are also often used (see for example Denaro *et al.* (2005); Zhang *et al.* (2008); Vašek *et al.* (2017)).

2.2.1 Hierarchical clustering tree: UPGMA

The 'UPGMA TREE' tab in BinMat allows the user to upload a consolidated binary matrix as a CSV file (in the format shown in Table 2), specify the number of bootstrap replications, and download the resulting hierarchical clustering tree as a scalable vector graphics (SVG) file. This function makes use of the pvclust function in the pvclust package (Suzuki *et al.*, 2019), and uses the UPGMA clustering method. The uploaded binary data is converted into a distance matrix applying the Jaccard transformation (dJ_i) (Jaccard, 1908) shown below. f_{11} represents the total number of times that a band occurred at the same

locus in both samples, f_{00} represents the shared absence of bands, and f_{10} and f_{01} represents the number of times that a band was present in only one of the two sample replicates. The Jaccard transformation was applied using the .dist function, applying the 'binary' method. This transformation was prefered because it does not treat the shared absence of bands as being biologically meaningful.

$$dJ_{\mathrm{i}} = \frac{f_{01} + f_{10}}{f_{01} + f_{10} + f_{11}}$$

2.2.2 Ordination: nMDS Plot

The 'nMDS PLOT' tab allows the user to upload a consolidated binary matrix with grouping information as a CSV file. The input file format is shown in Table 3, where grouping information needs to appear in the second column. The distance methods available are 'binary' (Jaccard's distance), 'euclidean', 'maximum', 'manhattan', 'canberra', and 'minkowski'. The 'No. of dimensions (k)' option can be set at '2' or '3', and can be determined using the 'nMDS Validation' tab using the 'Scree plot' and 'Shepard plot' buttons. The resulting distance matrix can be downloaded as a CSV file, and the plot itself as a SVG file. Once the user has uploaded their data, an editable table will appear to allow for the selection of colours and symbols for each group. The user can adjust symbol size, and can select whether sample labels should appear on the graph or not. The nMDS plot is created using the isoMDS function in the MASS package (Venables and Ripley, 2002).

2.2.3 Scree plot

The optimal number of dimensions to use for the nMDS plot should minimise the resulting stress value. Clarke (1993) suggest that stress values < 0.05 = excellent, < 0.10 = good, < 0.20 = usable, > 0.20 = not acceptable, while Dugard *et al.* (2010) suggest that a stress value below 0.15 represents a good fit for the data. BinMat indicates the 0.15 threshold as a dotted red line on the resulting scree plot.

2.2.4 Shepard plot

Shepard plots are graphical representations of how well the ordination fits the original distance data (Leeuw and Mair, 2014). BinMat plots the original Jaccard distances (x-axis) against the transformed distances used to create the nMDS ordination plot (y-axis). R^2-values are shown on the plot for the regression line of best fit.

2.2.5 Filter data

The 'Filter data' tab allows the user to filter their dataset by setting a threshold value for the number of peaks present. The new subsetted data, and the removed samples, can be downloaded as a CSV file and re-uploaded to create a new nMDS plot and/or hierarchical clustering tree.

2.3 Testing BinMat

2.3.1 Comparing BinMat's output to PAST and SplitsTree

Two AFLP datasets were downloaded from the Dryad Digital Repository, available at https://datadryad.org/stash/dataset/doi:10.5061/dryad.b5d6b and https://datadryad.org/stash/dataset/doi:10.5061/dryad.c3g80. These comprised data generated by Arias *et al.* (2014) and Tewes *et al.* (2018) for *Heliconius* (Lepidoptera: Nymphalidae) and *Bunias orientalis* L. (Brassicaceae) specimens, respectively. With the authors' permission, a subset of each were used to compare output from BinMat to that of PAST v4.0 (Paleontological Statistics Software Package for Education and Data Analysis) (Hammer *et al.*, 2001) and SplitsTree v4.14.6 (Huson, 1998) (raw data are available as supplementary files). Replicate pairs were consolidated in BinMat where applicable, and used to create nMDS plots and UPGMA hierarchical clustering trees (1000 bootstrap repetitions). The lowest number of dimensions were used for nMDS plots (k = 2), and their stress- and R^2 values recorded. SplitsTree was used to create a NeighborNet tree applying Jaccard's distance transformation.

The nMDS plots created by BinMat and PAST showed comparable clustering patterns (Fig. 2 A1-A2, and B1-B2). The SplitsTree output for the data taken from Tewes *et al.* (2018) (Fig. 2 B4) corroborated the corresponding nMDS plot from the original paper (Fig. 2 B3), and from that created by BinMat (Fig. 2 B1). Both hierarchical clustering trees using the UPGMA method showed equivalent topologies and bootstrap support values for clades (Fig. 3). BinMat, PAST, and SplitsTree perform equally as well for the visualisation of fragment analysis output, where BinMat offers the advantage of a quicker, automated process on one platform.

3 BinMat as an R package on CRAN

There are two example binary matrices embedded in the BinMat package, called "BinMatInput_reps" and "BinMatInput_ordination" that can be accessed by creating objects with names such as:

```
> data1 = BinmatInput_reps
> data2 = BinmatInput_ordination
```

which can be used to test the various functions as a demonstration example, as shown in the vignette supplied with the package.

3.1 Worked example

3.1.1 Binary matrix comprising replicate pairs

The `data1` object contains a binary data frame with replicate pairs (i.e. two replicate reads per sample). The `check.data()`, `consolidate()`, `peaks.original()`, `peak.remove()`, and `upgma()` functions can be applied to this object.

> `check.data(data1)` checks the matrix for any possible unwanted characters. If found, the function returns the row and column index where they occur. The output for the above line is

> `None found.`

The next step is to consolidate the replicate pairs in the matrix using the `consolidate()` function.

```
> data("BinMatInput_reps")
> data1 = BinMatInput_reps
> data1
  Sample X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14
1     A1  0  0  1  1  1  0  1  0  0   1   1   1   0   1
2     A2  0  1  1  1  1  0  1  0  1   1   1   1   0   1
3     B1  0  0  1  1  1  1  1  0  0   1   1   1   1   1
4     B2  1  1  1  1  1  0  1  1  1   1   1   1   0   1
5     C1  0  1  1  1  1  0  0  0  1   1   1   1   0   0
6     C2  1  0  1  1  0  0  0  1  0   1   1   0   0   0
7     D1  1  0  0  1  1  0  1  0  0   1   0   1   0   1
8     D2  1  1  1  1  1  0  1  0  1   1   0   1   0   0
> consolidate(data1)
      X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14
A1+A2  0  ?  1  1  1  0  1  0  ?   1   1   1   0   1
B1+B2  ?  ?  1  1  1  ?  1  ?  ?   1   1   1   ?   1
C1+C2  ?  ?  1  1  ?  0  0  ?  ?   1   1   ?   0   0
D1+D2  1  ?  ?  1  1  0  1  0  ?   1   0   1   0   ?
```

A summary of peak information can be obtained using the `peaks.original()` function. This averages the peak number across all replicates in the data set. If the user has a data set that does not need to be consolidated by BinMat, and they want to find the peak summary for it, this same function can be used.

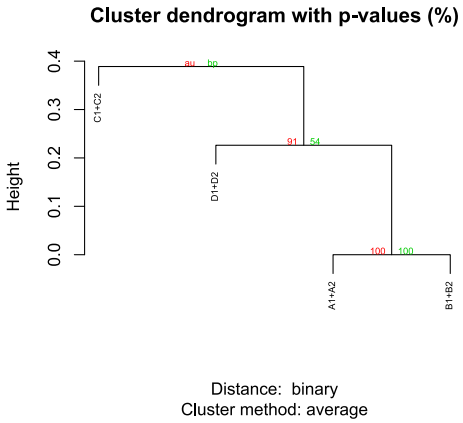
```
> peaks.original(data1)
                Summary
1  Average no. peaks:
2                  8.75
3                  sd:
4                1.9086
5      Max. no. peaks:
6                    12
7      Min. no. peaks:
8                     6
9            No. loci:
10                   14
```

Once the matrix has been consolidated, a UPGMA hierarchical clustering tree can be created. The de-

fault bootstrap repetition is set to 10. If the user wishes to upload their own matrix that they want to use to create the clustering tree, this needs to be read in first, and then specified in the `upgma()` function as `fromFile = TRUE`. For example:

```
> mydata = read.csv("C:/Users/General/Desktop/mydata.csv")
> upgma(mydata, fromFile = TRUE)
```

```
> cons = consolidate(data1)
> upgma(cons)
Bootstrap (r = 0.5)... Done.
Bootstrap (r = 0.57)... Done.
Bootstrap (r = 0.64)... Done.
Bootstrap (r = 0.79)... Done.
Bootstrap (r = 0.86)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.07)... Done.
Bootstrap (r = 1.14)... Done.
Bootstrap (r = 1.29)... Done.
Bootstrap (r = 1.36)... Done.
NULL
```



A peak summary for the consolidated matrix created in BinMat can be obtained by using the `peaks.consolidated()` function.

```
> peaks.consolidated(cons)
                Summary
1  Average no. peaks:
2                   6.5
3                   sd:
4                1.9149
5     Max. no. peaks:
6                     8
7     Min. no. peaks:
8                     4
9           No. loci:
10                 14
>
```

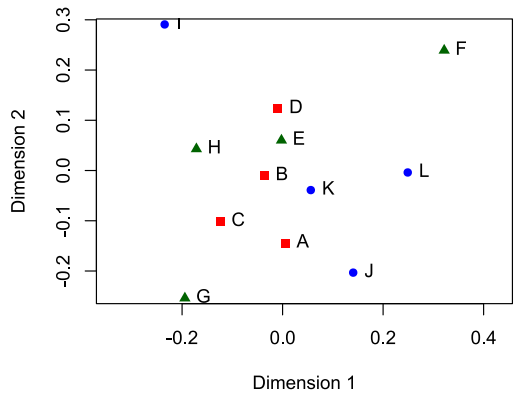
3.1.2 Consolidated binary matrix with grouping information

The `data2` object contains a binary data frame with a consolidated matrix and grouping information in the second column. The `errors()`, `group.names()`, `nmds()`, `peak.remove()`, `scree()`, and `shepard()` functions can be applied to this object.

```
> data2
   Sample     Group X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14
1       A    Africa  0  0  1  ?  1  0  ?  0  0   1   1   1   0   1
2       B    Africa  0  1  1  1  1  0  1  0  1   1   1   1   ?   1
3       C    Africa  0  ?  1  1  1  1  1  0  0   1   1   1   1   1
4       D    Africa  1  1  1  1  1  0  1  1  1   ?   1   1   0   1
5       E    Europe  ?  1  1  1  1  0  ?  0  1   1   1   1   0   0
6       F    Europe  1  0  1  1  0  0  0  1  0   1   1   0   0   0
7       G    Europe  1  0  0  1  1  0  1  0  0   1   0   1   0   1
8       H    Europe  1  1  1  1  1  ?  1  0  1   1   0   1   0   0
9       I Australia  ?  1  1  ?  0  0  1  0  0   0   1   1   0   0
10      J Australia  0  0  1  1  1  0  0  0  1   1   0   1   ?   0
11      K Australia  0  1  1  1  1  0  0  0  1   1   1   1   0   1
12      L Australia  0  0  1  1  ?  0  0  1  1   1   1   ?   0   1
> errors(data2)
                    Errors
1 Average Euclidean Error:
2                   0.0774
3 Euclidean error St. dev:
4                   0.0566
5         Average Jaccard:
6                   0.1222
7    Jaccard error St.dev:
8                   0.0992
> group.names(data2)
[1] "Africa"    "Australia" "Europe"
```

In order to create an nMDS plot, the user needs to create an object specifying colours and shapes of their choice, shown below as `clrs` and `shps`. These are passed as the `colours` and `shapes` arguments in the `nmds()` function, and are assigned to groups in the order appearing in the output from the `group.names()` function. In this example, Africa, Australia, and Europe will be red, blue, and dark green, respectively. Labels can be displayed by adding `labs = TRUE` as an argument.

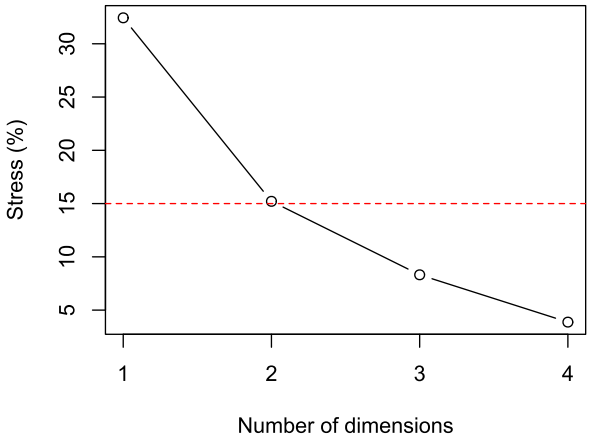
```
> clrs = c("red", "blue", "darkgreen")
> shps = c(15, 16, 17)
> nmds(data2, colours = clrs, shapes = shps, labs = TRUE)
initial  value 23.145184
iter   5 value 15.519401
iter  10 value 15.224397
iter  10 value 15.216857
iter  10 value 15.210646
final  value 15.210646
converged
NULL
>
```

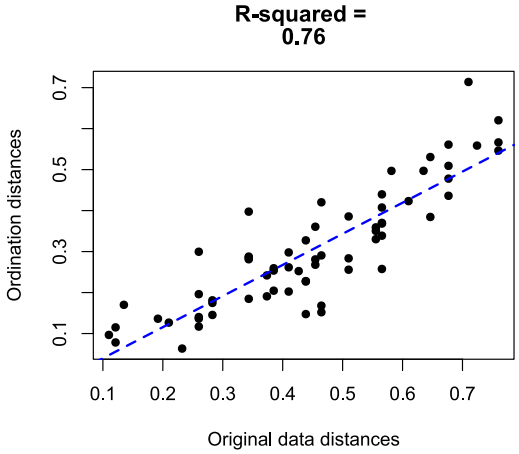


Scree and shepard plots are used to confirm the number of dimensions chosen to create the nMDS plot. The scree plot shows a dotted red line at y = 15% to indicate that dimensions with stress values below this are acceptable to use to create an nMDS plot.

```
> scree(data2)
initial  value 7.158898
iter   5 value 4.062583
iter  10 value 3.951154
iter  15 value 3.911452
final  value 3.873089
converged
initial  value 17.256889
iter   5 value 13.318548
iter  10 value 10.281815
iter  15 value 9.885853
iter  20 value 9.230169
iter  25 value 8.712386
iter  30 value 8.413392
final  value 8.314316
converged
initial  value 23.145184
iter   5 value 15.519401
iter  10 value 15.224397
iter  10 value 15.216857
iter  10 value 15.210646
final  value 15.210646
converged
initial  value 40.284721
iter   5 value 33.922426
final  value 32.437297
converged
NULL
>

> shepard(data2)
initial  value 23.145184
iter   5 value 15.519401
iter  10 value 15.224397
iter  10 value 15.216857
iter  10 value 15.210646
final  value 15.210646
converged
NULL
>
```





4 Obtaining BinMat

The R Shiny application platform allocates a maximum memory of 1 GB, and is accessible at https://clarkevansteenderen.shinyapps.io/BINMAT/. The online version may time-out due to insufficient memory if a particularly large file is uploaded. In such a case, the program can alternatively be run directly from R on the user's local machine by typing

> `shiny::runGitHub("BinMat", "CJMvS")`

into the console. The program's code is freely available via Github at https://github.com/CJMvS/BinMat. The BinMat R package is also available on CRAN (Comprehensive R Archive Network), and is command-line driven. More information about the package can be obtained by typing

> `library(help = BinMat)`

after it has been installed. This details all the functions available (Table 4). More information about each function, and the parameters it requires, can be accessed by typing

> `?functionName`

into the console.

To my knowledge, this is the only freely-available application offering the functionality presented here. Suggestions for improvement (for example via pull-requests on GitHub), and feedback from the community, are welcomed and encouraged.

Acknowledgements

I wish to thank Rhodes University and the Centre for Biological Control in the Department of Zoology and Entomology for the provision of funding and facilities. Guy Sutton is thanked for his valuable advice and suggestions in the writing of this manuscript. Megan Reid is thanked for testing the program's functionality and providing feedback. The supervisors for my M.Sc. project, Dr. Iain Paterson and Dr. Shelley Edwards, are thanked for their assistance throughout the course of the degree.

References

Abbot, P., (2001). Individual and population variation in invertebrates revealed by Inter-simple Sequence Repeats (ISSRs). *Journal of Insect Science*, 1(1):8.

AppliedBiosystems. *DNA Fragment Analysis by Capillary Electrophoresis*. Applied Biosystems, (2014).

Archer, F. I., Adams, P. E., and Schneiders, B. B., (2017). stratag: An R package for manipulating, summarizing and analysing population genetic data. *Molecular Ecology Resources*, 17(1):5–11. doi:10.1111/1755-0998.12559.

Arias, C. F., Salazar, C., Rosales, C., Kronforst, M. R., Linares, M., Bermingham, E., and McMillan, W. O., (2014). Phylogeography of *Heliconius cydno* and its closest relatives: disentangling their origin and diversification. *Molecular Ecology*, 23(16):4137–4152. doi:10.1111/mec.12844.

Arrigo, N., Holderegger, R., and Alvarez, N., (2012). Automated scoring of AFLPs using RawGeno v 2.0, a free R CRAN library. In *Data Production and Analysis in Population Genomics*, pages 155–175. Springer. doi:10.1007/978-1-61779-870-2_10.

Bassam, B. J., Caetano-Anollés, G., and Gresshoff, P. M., (1991). Fast and sensitive silver staining of DNA in polyacrylamide gels. *Analytical Biochemistry*, 196(1):80–83.

Bonin, A., Bellemain, E., Bronken Eidesen, P., Pompanon, F., Brochmann, C., and Taberlet, P., (2004). How to track and assess genotyping errors in population genetics studies. *Molecular Ecology*, 13(11): 3261–3273. doi:10.1111/j.1365-294X.2004.02346.x.

Clarke, K. R., (1993). Non-parametric multivariate analyses of changes in community structure. *Australian Journal of Ecology*, 18(1):117–143. doi:10.1111/j.1442-9993.1993.tb00438.x.

Denaro, R., D'auria, G., Di Marco, G., Genovese, M., Troussellier, M., Yakimov, M., and Giuliano, L., (2005). Assessing terminal restriction fragment length polymorphism suitability for the description of bacterial community structure and dynamics in hydrocarbon-polluted marine environments. *Environmental Microbiology*, 7(1):78–87. doi:10.1111/j.1462-2920.2004.00685.x.

Dresler-Nurmi, A., Terefework, Z., Kaijalainen, S., Lindström, K., and Hatakka, A., (2000). Silver stained polyacrylamide gels and fluorescence-based automated capillary electrophoresis for detection of amplified fragment length polymorphism patterns obtained from white-rot fungi in the genus *Trametes*. *Journal of Microbiological Methods*, 41(2):161–172. doi:10.1016/S0167-7012(00)00153-6.

Dugard, P., Todman, J., and Staines, H., (2010). *Approaching multivariate analysis. A practical introduction*. CRC Press, Taylor and Francis, Routledge, New York.

Hammer, O., Harper, D. A. T., and Ryan, P. D., (2001). PAST: Paleontological Statistics Software Package for Education and Data Analysis. *Palaeontologia Electronica*, 4:9.

Holland, B. R., Clarke, A. C., and Meudt, H. M., (2008). Optimizing automated AFLP scoring parameters to improve phylogenetic resolution. *Systematic Biology*, 57(3):347–366. doi:10.1080/10635150802044037.

Huson, D. H., (1998). SplitsTree: analyzing and visualizing evolutionary data. *Bioinformatics (Oxford, England)*, 14(1):68–73. doi:10.1093/bioinformatics/14.1.68.

Jaccard, P., (1908). Nouvelles recherches sur la distribution florale. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 44:223–70. doi:10.5169/seals-268384.

Koeleman, J. G., Stoof, J., Biesmans, D. J., Savelkoul, P. H., and Vandenbroucke-Grauls, C. M., (1998). Comparison of amplified ribosomal DNA restriction analysis, random amplified polymorphic DNA analysis, and amplified fragment length polymorphism fingerprinting for identification of *Acinetobacter* genomic species and typing of *Acinetobacter baumannii*. *Journal of Clinical Microbiology*, 36(9):2522–2529.

Leeuw, J. D. and Mair, P., (2014). Shepard diagram. *Wiley StatsRef: Statistics Reference Online*, pages 1–3. doi:10.1002/9781118445112.stat06268.pub2.

Liu, L.-j., Meng, Z.-Q., Wang, B., Wang, X.-x., Yang, J.-Y., and Peng, D.-x., (2009). Genetic diversity among wild resources of the genus Boehmeria Jacq. from west China determined using inter-simple sequence repeat and rapid amplification of polymorphic DNA markers. *Plant Production Science*, 12 (1):88–96. doi:10.1626/pps.12.88.

Paradis, E., Claude, J., and Strimmer, K., (2004). APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, 20(2):289–290. doi:10.1093/bioinformatics/btg412.

Pompanon, F., Bonin, A., Bellemain, E., and Taberlet, P., (2005). Genotyping errors: causes, consequences and solutions. *Nature Reviews Genetics*, 6(11):847. doi:10.1038/nrg1707.

R Core Team. R: A language and environment for statistical computing, (2019).

Reyes, A. L. P., Silva, T. C., Coetzee, S. G., Plummer, J. T., Davis, B. D., Chen, S., Hazelett, D. J., Lawrenson, K., Berman, B. P., Gayther, S. A., *et al.*, (2019). GENAVi: a shiny web application for gene expression normalization, analysis and visualization. *BMC Genomics*, 20(1):745. doi:10.1186/s12864-019-6073-7.

Schliep, K. P., (2011). Phangorn: Phylogenetic analysis in R. *Bioinformatics*, 27(4):592–593. doi:10.1093/bioinformatics/btq706.

Sutton, G. F., Paterson, I. D., and Paynter, Q., (2017). Genetic Matching of Invasive Populations of the African Tulip Tree, *Spathodea Campanulata* Beauv. (Bignoniaceae), to Their Native Distribution: Maximising the Likelihood of Selecting Host-Compatible Biological Control Agents. *Biological Control*, 114:167–175. doi:10.1016/j.biocontrol.2017.08.015.

Suzuki, R., Terada, Y., and Shimodaira, H. *pvclust: Hierarchical Clustering with P-Values via Multiscale Bootstrap Resampling*, (2019).

Taylor, S. J., Downie, D. A., and Paterson, I. D., (2011). Genetic Diversity of Introduced Populations of the Water Hyacinth Biological Control Agent *Eccritotarsus catarinensis* (Hemiptera: Miridae). *Biological Control*, 58(3):330–336. doi:10.1016/j.biocontrol.2011.05.008.

Tewes, L. J., Michling, F., Koch, M. A., and Müller, C., (2018). Intracontinental plant invader shows matching genetic and chemical profiles and might benefit from high defence variation within populations. *Journal of Ecology*, 106(2):714–726. doi:10.1111/1365-2745.12869.

Ticknor, L. O., Kolstø, A.-B., Hill, K. K., Keim, P., Laker, M. T., Tonks, M., and Jackson, P. J., (2001). Fluorescent amplified fragment length polymorphism analysis of Norwegian *Bacillus cereus* and *Bacillus thuringiensis* soil isolates. *Applied and Environmental Microbiology*, 67(10):4863–4873. doi:10.1128/AEM.67.10.4863–4873.2001.

Timm, A., Geertsema, H., and Warnich, L., (2010). Population genetic structure of economically important Tortricidae (Lepidoptera) in South Africa: a comparative analysis. *Bulletin of Entomological Research*, 100(4):421–431. doi:10.1017/S0007485309990435.

Van Eldere, J., Janssen, P., Hoefnagels-Schuermans, A., van Lierde, S., and Peetermans, W. E., (1999). Amplified-Fragment Length Polymorphism Analysis versus Macro-Restriction Fragment Analysis for Molecular Typing of *Streptococcus pneumoniae* Isolates. *Journal of Clinical Microbiology*, 37(6):2053–2057.

Vašek, J., Čepková, P. H., Viehmannová, I., Ocelak, M., Huansi, D. C., and Vejl, P., (2017). Dealing with AFLP genotyping errors to reveal genetic structure in *Plukenetia volubilis* (Euphorbiaceae) in the Peruvian Amazon. *Public Library of Science (PloS) one*, 12(9). doi:10.1371/journal.pone.0184259.

Venables, W. N. and Ripley, B. D., (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Vos, P., Hogers, R., Bleeker, M., Reijans, M., Lee, T. v. d., Hornes, M., Friters, A., Pot, J., Paleman, J., Kuiper, M., *et al.*, (1995). AFLP: a new technique for DNA fingerprinting. *Nucleic Acids Research*, 23 (21):4407–4414.

Wolfe, A. D. and Liston, A., (1998). Contributions of pcr-based methods to plant systematics and evolutionary biology. In *Molecular Systematics of Plants II*, pages 43–86. Springer.

Zhang, R., Thiyagarajan, V., and Qian, P.-Y., (2008). Evaluation of terminal-restriction fragment length polymorphism analysis in contrasting marine environments. *Federation of European Microbiological Societies (FEMS) Microbiology Ecology*, 65(1):169–178. doi:10.1111/j.1574-6941.2008.00493.x.

Data Availability Statement

Code accessibility:

R CRAN:

`https://cran.r-project.org/web/packages/BinMat/index.html`

`https://github.com/CJMvS/BinMatPackage`

R Shiny:

`https://github.com/CJMvS/BinMat`

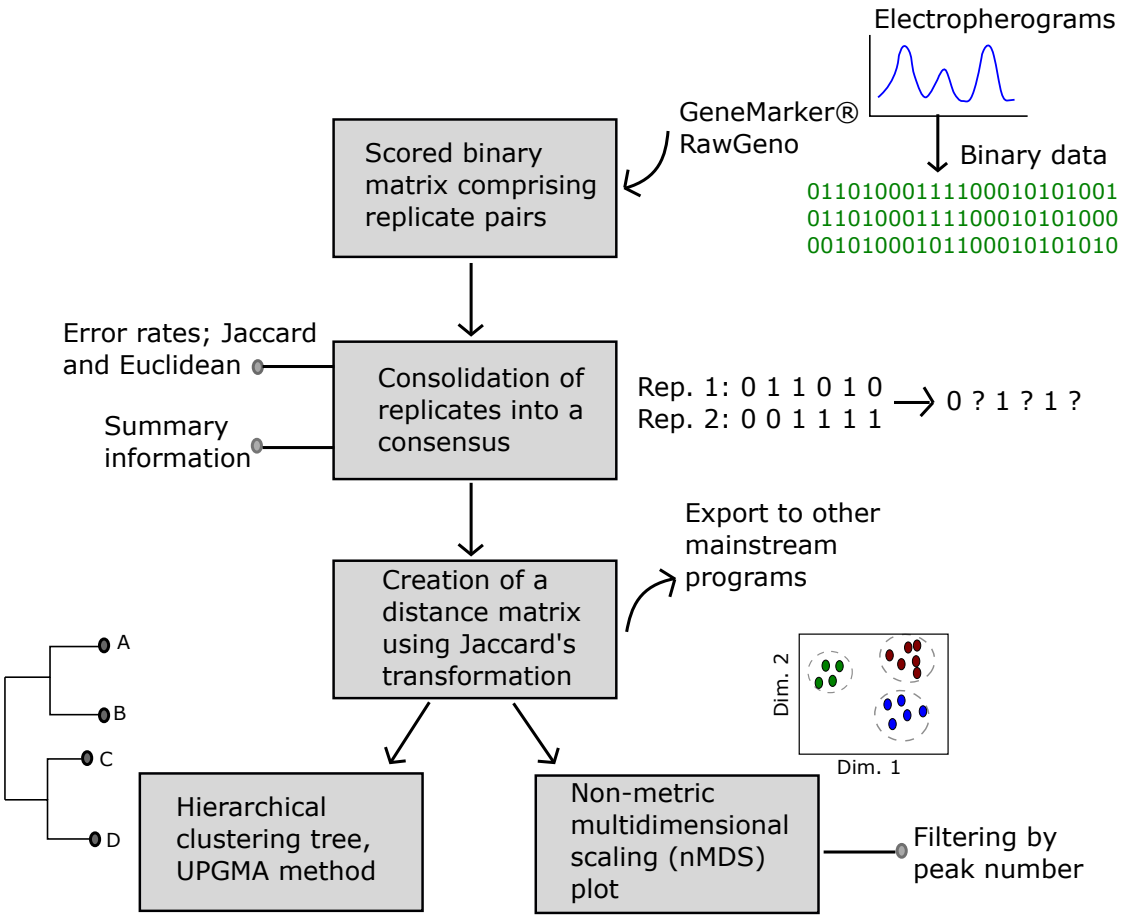


Figure 1: Flowchart of the utility of the BinMat program, starting with input that has been processed in programs such as GeneMarker® and RawGeno, to the rapid visualisation of a hierarchical clustering tree and non-metric dimensional scaling (nMDS) plot.

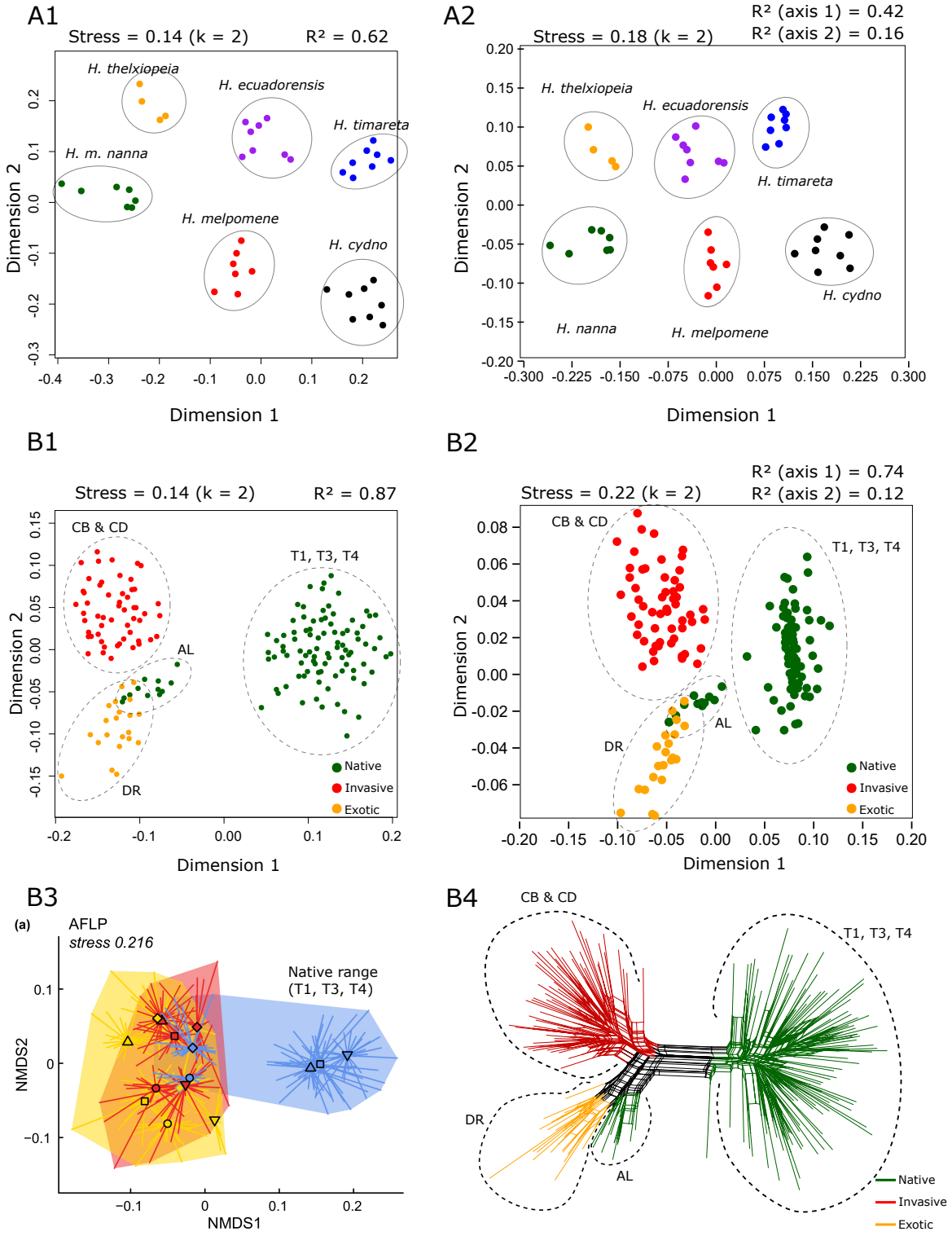


Figure 2: Comparisons of non-metric multidimensional scaling (nMDS) plots in BinMat (A1 and B1), and PAST (A2 and B2). Both nMDS plots are plotted for k = 2 dimensions. Data were taken from Arias *et al.* (2014) (A1 and A2) and Tewes *et al.* (2018) (B1, B2, and B4). Stress-and R^2 values are shown above each plot. Diagram B3 shows the original nMDS plot presented by Tewes *et al.* (2018), which depicts the same clustering pattern of the native range samples (T1, T3, and T4). Diagram B4 shows the SplitsTree representation of the same data (NeighborNet, Jaccard distance).

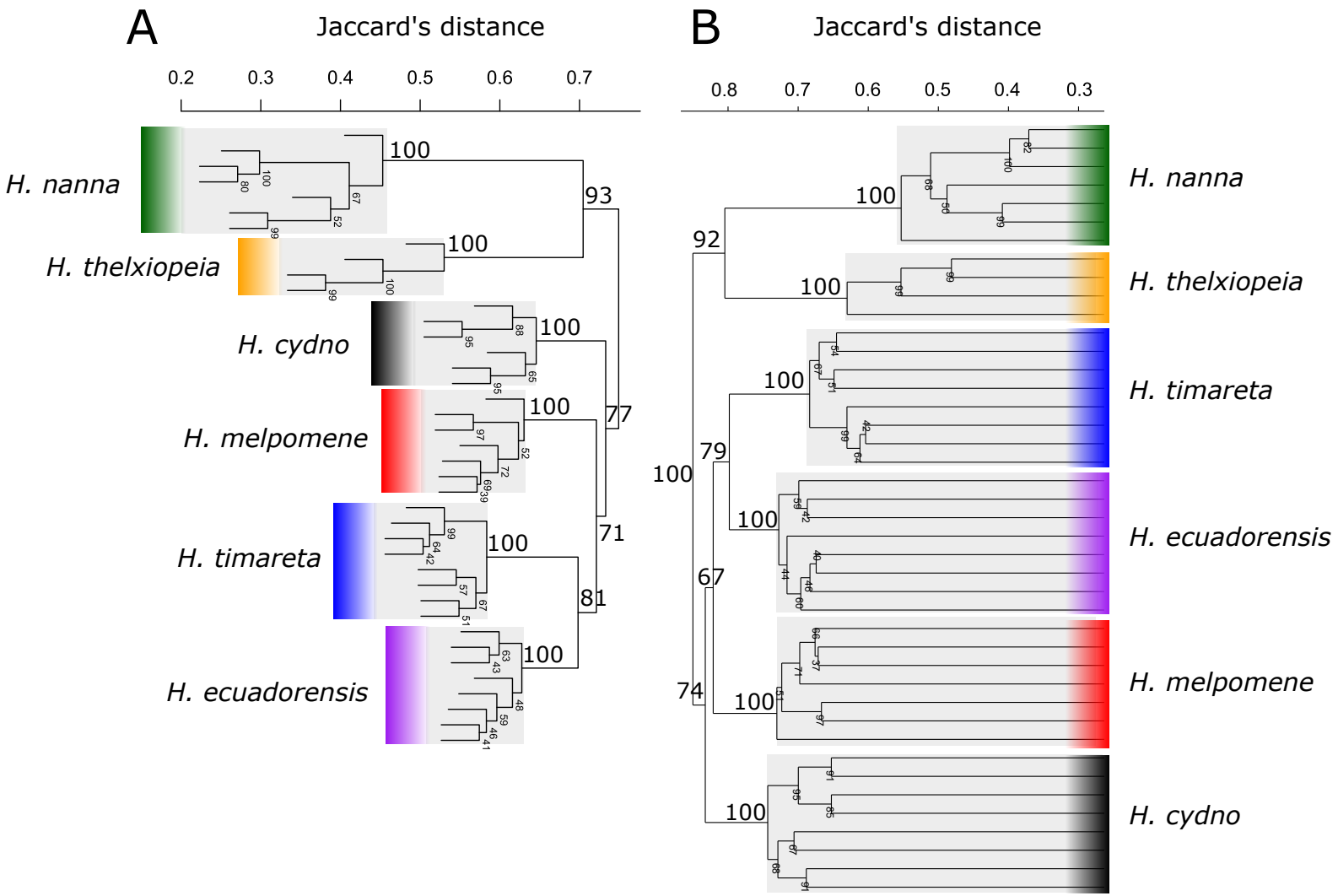


Figure 3: Comparison of hierarchical clustering trees in A) BinMat, and B) PAST. Both programs applied Jaccard's transformation to create a distance matrix, and used the UPGMA clustering method. Bootstrap probabilities are shown on the branches, resulting from 1000 repetitions.

Table 1: File input for a dataset containing replicate pairs that needs to be consolidated.

Sample label	Locus 1	Locus 2	Locus 3	Locus 4	Locus 5
Sample A rep. 1	0	0	1	1	1
Sample A rep. 2	0	0	1	1	1
Sample B rep. 1	1	1	0	0	0
Sample B rep. 2	0	1	0	0	1

Table 2: Consolidated matrix if Table 1 was used as input.

Sample label	Locus 1	Locus 2	Locus 3	Locus 4	Locus 5
Sample A rep. 1 + rep. 2	0	0	1	1	1
Sample B rep. 1 + rep. 2	?	1	0	0	?

Table 3: Data input required for the creation of a non-metric multidimensional scaling (nMDS) plot. Grouping information needs to be in the second column. Data represents binary replicate pairs that have already been consolidated into a consensus.

Sample label	Group	Locus 1	Locus 2	Locus 3	Locus 4	Locus 5
Sample A	Africa	0	0	1	1	1
Sample B	Asia	?	1	0	0	?

Table 4: BinMat R package functions, available on CRAN. Typing `?functionName` into the console provides more information about each function.

Function	**Description**
`check.data()`	Checks for unwanted characters.
`consolidate()`	Consolidates replicate pairs. 1&1 →1; 1&0 →?; 0&0 →0
`errors()`	Calculates Jaccard and Euclidean error rates.
`group.names()`	Outputs groups in the uploaded binary matrix.
`nmds()`	Creates a non-metric multidimensional scaling (nMDS) plot.
`peak.remove()`	Removes samples with peaks equal to, or less than, a specified threshold value.
`peaks.consolidated()`	Peak summary for a consolidated binary matrix.
`peaks.original()`	Peak summary for replicate data, or consolidated data from file.
`scree()`	Creates a scree plot of stress values vs ordination dimensions.
`shepard()`	Creates a shepard plot for goodness of fit for ordination data.
`upgma()`	Draws a hierarchical clustering tree (UPGMA) with bootstrapping.