BIOINFORMATICS APPLICATIONS NOTE Vol. 22 no. 12 2006, pages 1540–1542
doi:10.1093/bioinformatics/btl117

Data and text mining

Pvclust: an R package for assessing the uncertainty in hierarchical clustering

Ryota Suzuki[1,2,*] and Hidetoshi Shimodaira[1]

[1]Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan and [2]Ef-prime, Inc., 2-17-5 Nihonbashi-Kayabacho, Chuo-ku, Tokyo 103-0025, Japan

Received on August 30, 2005; revised on March 6, 2006; accepted on March 25, 2006
Advance Access publication April 4, 2006
Associate Editor: Satoru Miyano

ABSTRACT

Summary: Pvclust is an add-on package for a statistical software R to assess the uncertainty in hierarchical cluster analysis. Pvclust can be used easily for general statistical problems, such as DNA microarray analysis, to perform the bootstrap analysis of clustering, which has been popular in phylogenetic analysis. Pvclust calculates probability values (*p*-values) for each cluster using bootstrap resampling techniques. Two types of *p*-values are available: approximately unbiased (AU) *p*-value and bootstrap probability (BP) value. Multiscale bootstrap resampling is used for the calculation of AU *p*-value, which has superiority in bias over BP value calculated by the ordinary bootstrap resampling. In addition the computation time can be enormously decreased with parallel computing option.

Availability: The program is freely distributed under GNU General Public License (GPL) and can directly be installed from CRAN (http://cran.r-project.org/), the official R package archive. The instruction and program source code are available at http://www.is.titech.ac.jp/~shimo/prog/pvclust

Contact: ryota.suzuki@is.titech.ac.jp

Cluster analysis is a statistical method which aims to classify several objects into some groups (clusters) according to similarities between them. While it has been widely used in many applications such as DNA microarray analysis, the uncertainty of results caused by sampling error of data has not generally been evaluated in practice. Pvclust is an implementation of bootstrap analysis on a statistical software R to assess the uncertainty in hierarchical cluster analysis.

The importance of uncertainty assessment has been well-recognized in phylogenetic analysis. It is a special form of hierarchical clustering for inferring the history of evolution as a dendrogram. Thousands of bootstrap samples are generated by randomly sampling elements of the data, and bootstrap replicates of the dendrogram are obtained by repeatedly applying the cluster analysis to them (Efron, 1979; Felsenstein, 1985). The bootstrap probability (BP) value of a cluster is the frequency that it appears in the bootstrap replicates. The multiscale bootstrap resampling was developed recently (Efron *et al*., 1996; Shimodaira, 2002, 2004) for calculating approximately unbiased (AU) probability values (*p*-values) as implemented in a software CONSEL (Shimodaira and Hasegawa, 2001).

*To whom correspondence should be addressed.

Although these bootstrap-based approaches are applicable to broad range of statistical problems, their usage have been limited since implementations of these methods are focused only on phylogenetic analysis. Pvclust is designed for general hierarchical clustering problems, so users can easily obtain bootstrap-based *p*-values for their own dataset and preferred clustering method. (Suzuki and Shimodaira, 2004).

R is 'a free software environment for statistical computing and graphics' (the R project website, http://www.r-project.org/), which runs on several platforms such as Windows, MacOS and UNIX/Linux. Several add-on packages are available via CRAN, the official R package archive.

Package pvclust is included in CRAN packages and can be easily installed on the fly. Once the package is installed, pvclust can be run with the command:

```
library (pvclust)
```

In R system data are stored in a 'data object'. For example, to read a data file `data.txt` into a data object `data`, type the following command:

```
data < - read.table(''data.txt'')
```

The bootstrap analysis is performed by applying the function `pvclust` to the object `data` with the number of bootstrap replications being $B = 10\,000$:

```
result < - pvclust(data, nboot=10000)
```

The result is stored in the object `result`, which can be shown graphically with the command:

```
plot(result)
```

We have applied the above commands to the DNA microarray data of Garber *et al*. (2001), and the result is shown in Figure 1. By default, `pvclust` performs bootstrapping at $K = 10$ different data sizes, and the hierarchical clustering is repeated by $K \times B$ times. It took 430 min on a single processor (Athlon MP 2000+). The parallel version `parPvclust` is available with the help of the snow package (Rossini *et al*., 2003), an add-on package of R for parallel computation. It took only 24 min using 20 processors; the speed-up value is 430/24 = 18, indicating very efficient parallel computing.

It is recommended to use `nboot = 1000` (default) for testing at first, followed by 10 000 for smaller errors. To determine an appropriate size of B (`nboot`), standard errors of *p*-values are helpful.

 © The Author 2006. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oxfordjournals.org

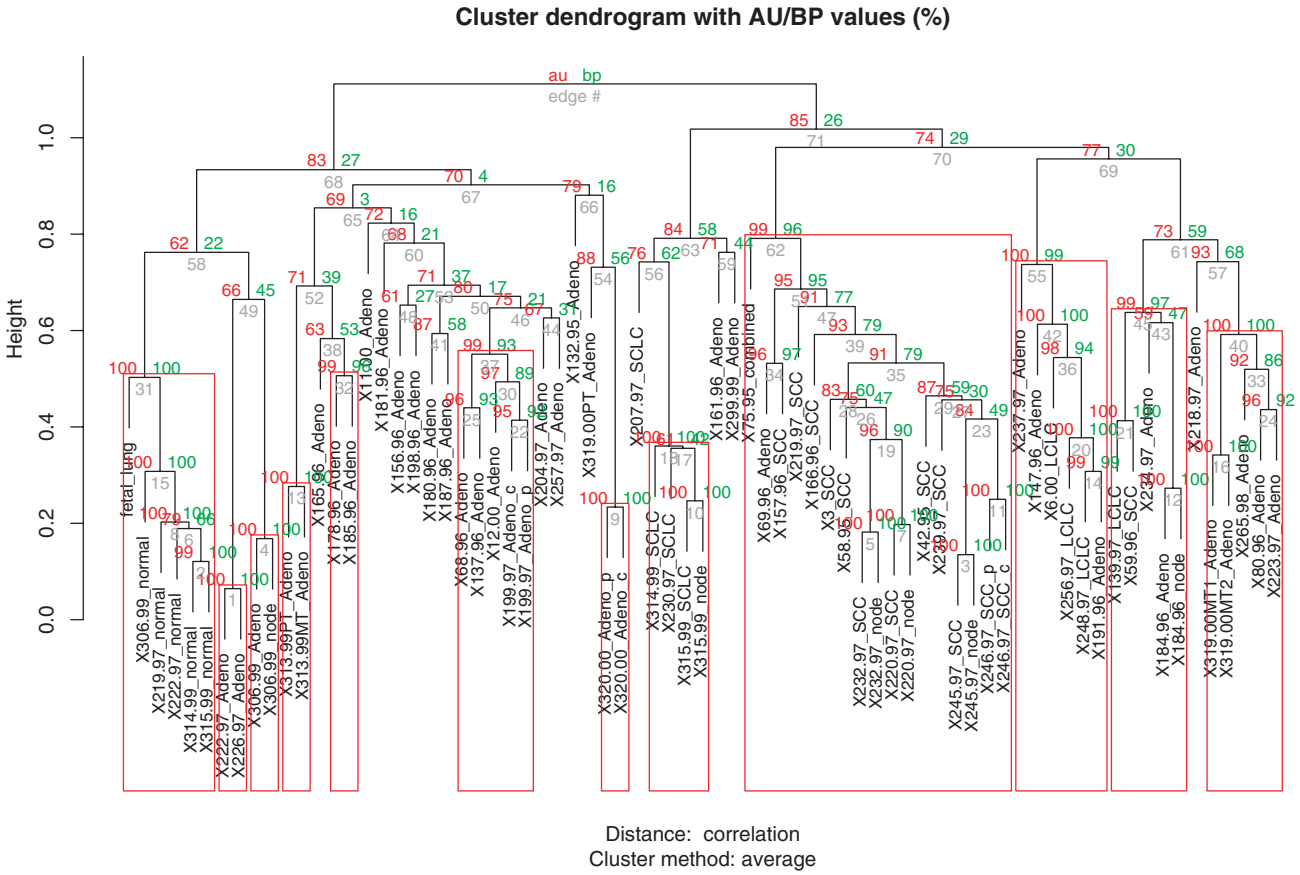

Fig. 1. Hierarchical clustering of 73 lung tumors. The data are expression pattern of 916 genes of Garber *et al*. (2001). Values at branches are AU p-values (left), BP values (right), and cluster labels (bottom). Clusters with AU ≥ 95 are indicated by the rectangles. The fourth rectangle from the right is a cluster labeled 62 with AU $= 0.99$ and BP $= 0.96$.

Function `seplot` provides a graphical interface for examining standard errors, while `print` gives more detailed information about p-values in text-based format. See online instruction on our website for the usage of these facilities.

In the multiscale bootstrap resampling, we intentionally alter the data size of bootstrap samples to several values. Let N be the original data size, and N' be that for bootstrap samples. In the example of Figure 1, $N = 916$, and $N' = 458$, 549, 641, 732, 824, 916, 1007, 1099, 1190 and 1282. For each cluster, an observed BP value is obtained for each value of N', and we look at change in $z = -\Phi^{-1}$ (BP) values, where $\Phi^{-1}(\cdot)$ is the inverse function of $\Phi(\cdot)$, the standard normal distribution function. For the cluster labeled 62 in Figure 1, the observed BP values are 0.8554, 0.8896, 0.9132, 0.9335, 0.9498, 0.9636, 0.9656, 0.9756, 0.9795 and 0.9859 (Fig. 2). Then, a theoretical curve $z(N') = v\sqrt{N'/N} + c\sqrt{N/N'}$ is fitted to the observed values, and the coefficients v, c are estimated for each cluster. The AU p-value is computed by AU $= \Phi(-v + c)$. For the cluster labeled 62, $v = -2.01$, $c = 0.26$, and thus AU $= \Phi(2.01 + 0.26) = \Phi(2.27) = 0.988$, where BP $= 0.964$ for $N' = N$. An asymptotic theory proves that the AU p-value is less biased than the BP value.

Currently only the simplest form of the bootstrapping, i.e. the non-parametric bootstrap resampling, is implemented in pvclust. More elaborate models designed for specific applications, such

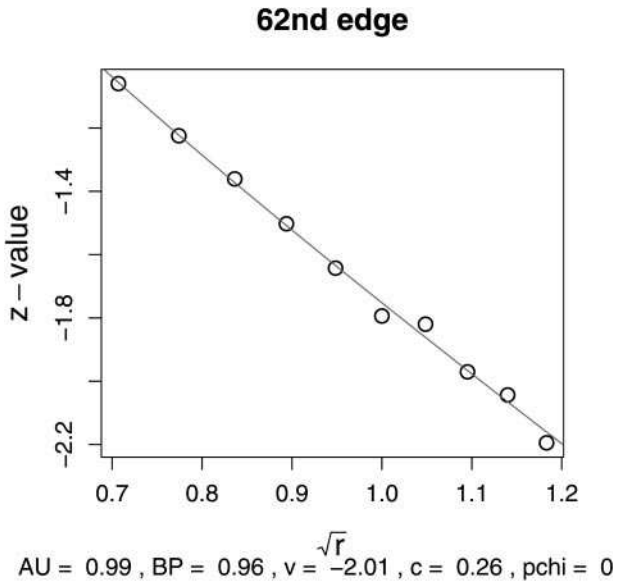

Fig. 2. Diagnostic plot of the multiscale bootstrap for the cluster labeled 62. The observed z-values are plotted for $\sqrt{N'/N}$, and the theoretical curve is obtained by the weighted least squares fitting. This plot is obtained by command: `msplot(result, edges=62)`. When the curve fitting is poor, a breakdown of the asymptotic theory may be suspected.

as that of Kerr and Churchill (2001) for DNA microarray analysis, should be incorporated into the program in a future work.

ACKNOWLEDGEMENTS

This work is supported in part by Grant KAKENHI (14702061, 17700276) from MEXT of Japan.

Conflict of Interest: none declared.

REFERENCES

Efron,B. (1979) Bootstrap methods: another look at the jackknife. *Ann. Stat.*, **7**, 1–26.

Efron,B. *et al.* (1996) Bootstrap confidence levels for phylogenetic trees. *Proc. Natl Acad. Sci., USA*, **93**, 13429–13434.

Felsenstein,J. (1985) Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, **39**, 783–791.

Garber,M. *et al.* (2001) Diversity of gene expression in adenocarcinoma of the lung. [Erratum (2002) *Proc. Natl Acad. Sci. USA,* 99, 1098.] *Proc. Natl Acad. Sci. USA*, **98**, 13784–13789.

Kerr,M.K. and Churchill,G.A. (2001) Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments. *Proc. Natl Acad. Sci. USA*, **98**, 8961–8965.

Rossini,A. *et al.* (2003) Simple parallel statistical computing in R. *UW Biostatistics Working Paper Series. Paper 193*, University of Washington, WA.

Shimodaira,H. and Hasegawa,M. (2001) CONSEL: for assessing the confidence of phylogenetic tree selection. *Bioinformatics*, **17**, 1246–1247.

Shimodaira,H. (2002) An approximately unbiased test of phylogenetic tree selection. *Syst. Biol.*, **51**, 492–508.

Shimodaira,H. (2004) Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling. *Ann. Stat.*, **32**, 2616–2641.

Suzuki,R. and Shimodaira,H. (2004) An application of multiscale bootstrap resampling to hierarchical clustering of microarray data: how accurate are these clusters? In *proceedings by the Fifteenth International Conference on Genome Informatics (GIW 2004)*, p. P034.

BIOINFORMATICS APPLICATIONS NOTE Vol. 22 no. 21 2006, pages 2695–2696
doi:10.1093/bioinformatics/btl461

Structural bioinformatics

Bio3d: an R package for the comparative analysis of protein structures

Barry J. Grant[1,*], Ana P. C. Rodrigues[2], Karim M. ElSawy[3], J. Andrew McCammon[1,4] and Leo S. D. Caves[3]

[1]Department of Chemistry and Biochemistry, University of California, San Diego, La Jolla, CA 92093, USA, [2]The Burnham Institute for Medical Research, La Jolla, CA 92037, USA, [3]Department of Biology, University of York, York YO10 5YW, UK and [4]Howard Hughes Medical Institute, University of California, San Diego, La Jolla, CA 92093, USA

Received on July 18, 2006; revised on August 22, 2006; accepted on August 23, 2006
Advance Access publication August 29, 2006
Associate Editor: Anna Tramontano

ABSTRACT

Summary: An automated procedure for the analysis of homologous protein structures has been developed. The method facilitates the characterization of internal conformational differences and inter-conformer relationships and provides a framework for the analysis of protein structural evolution. The method is implemented in bio3d, an R package for the exploratory analysis of structure and sequence data.

Availability: The bio3d package is distributed with full source code as a platform-independent R package under a GPL2 license from: http://mccammon.ucsd.edu/~bgrant/bio3d/

Contact: bgrant@mccammon.ucsd.edu

1 INTRODUCTION

The detailed comparison of homologous protein structures can be used to infer pathways for evolutionary adaptation and, at closer evolutionary distances, mechanisms for conformational change. Traditionally, such investigations have involved careful visual inspection combined with structural alignment methods. These procedures are both time consuming and labor intensive, and require expert insight into the systems studied. With the growing number of determined protein structures, the availability of automatic procedures for analyzing the differences and similarities between structures becomes increasingly desirable.

The bio3d package contains utilities to process, organize and explore structure and sequence data. Features include the ability to read and write structure, sequence and dynamic trajectory data, perform atom summaries, atom selection, re-orientation, superposition, rigid core identification, clustering, distance matrix analysis, structure and sequence conservation analysis, and principal component analysis (PCA). Bio3d takes advantage of the extensive graphical and statistical capabilities of the R environment (R development core team, 2006; http://www.R-project.org), and thus represents a useful framework for exploratory analysis of structural data.

*To whom correspondence should be addressed.

2 COMPARATIVE ANALYSIS OF PROTEIN STRUCTURES WITH BIO3D

The bio3d package employs refined structural superposition and PCA to examine the relationship between different conformers. Conventionally, structural superposition of protein structures minimizes the root mean square difference between their full set of equivalent residues. However, for the current application such a superposition procedure can be inappropriate. For example, in the comparison of a multi-domain protein that has undergone a hinge-like rearrangement of its domains, standard 'all atom' superposition would result in an underestimate of the true atomic displacement by attempting superposition over all domains (whole structure superposition). A more appropriate and insightful superposition would be anchored at the most invariant region and hence more clearly highlight the domain rearrangement (sub-structure superposition). To avoid such problems, the current protocol includes an iterated superposition procedure, where residues displaying the largest positional differences are excluded at each round until only the invariant 'core' residues remain (Gerstein and Altman, 1995).

Following core identification and subsequent superposition, PCA is employed to examine the relationship between different conformers/structures based on their equivalent residues. The application of PCA to both distributions of experimental structures and Molecular Dynamics trajectories, along with its ability to provide considerable insight into the nature of conformational differences in a range of protein families and other biomolecules, has been discussed previously (Abseher *et al*., 1998; Caves *et al*., 1998; Elsawy *et al*., 2005; van Aalten *et al*., 1997). Briefly, the resulting principal components (orthogonal eigenvectors) describe the axes of maximal variance of the distribution of structures. Projection of the distribution onto the subspace defined by the largest principal components results in a lower dimensional representation of the structural dataset. The percentage of the total mean square displacement (or variance) of atom positional fluctuations captured in each dimension is characterized by their corresponding eigenvalue. Experience suggests that 3–5 dimensions are often sufficient to capture over 70% of the total variance in a given family of structures. Thus, a handful of principal components are sufficient to

© 2006 The Author(s)
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/2.0/uk/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

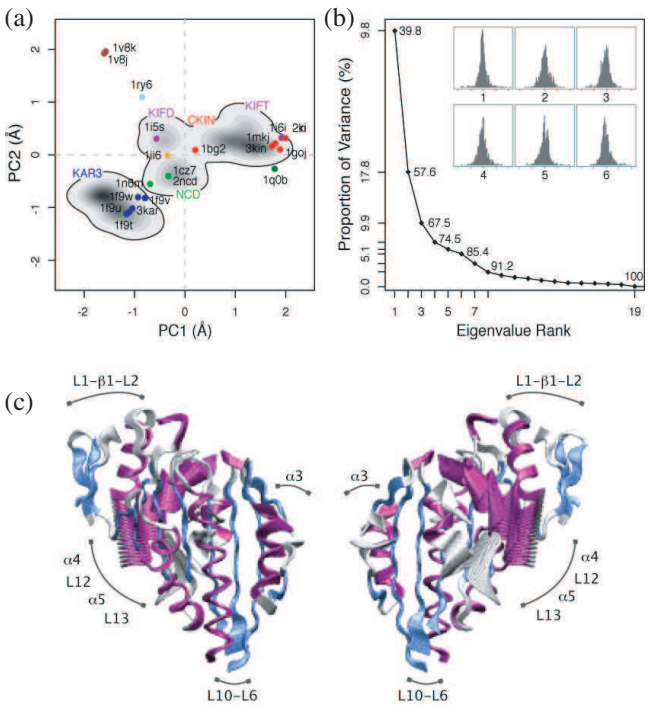

Fig. 1. Results of PCA on the kinesin molecular motor using standard Euclidean distance. (**a**) Conformer plot: Projection of the kinesin X-ray structures (circles) and transient MD conformers (shaded density contours) onto the principal planes obtained from analysis of all kinesin X-ray structures. (**b**) Eigenvalue spectrum: Results obtained from diagonalization of the atomic displacement correlation matrix of Cα atom coordinates from the kinesin crystal structures. Inset shows histograms for the projection of the distribution of structures onto the first six principal components. (**c**) Interpolation: Front and back views of the kinesin motor domain, with the first principal component represented as equidistant atomic displacements from the mean structure. Displacements are scaled by two times the standard deviation of the distribution along the first principal component. Molecular figure was generated using VMD (Humphrey *et al*., 1996).

provide a useful description while still retaining most of the variance in the original distribution. These low-dimensional representations, here termed 'conformer plots', succinctly display the relationships between different conformers, highlight the major differences between structures and enable the interpretation and characterization of multiple interconformer relationships (see example conformer plot, Fig. 1).

To further aid interpretation, a graphic 'trajectory' can be produced that interpolates between the most dissimilar structures in the distribution, as determined from the conformer plots. This involves dividing the difference between the conformers into a number of evenly spaced steps along the principal components, forming the frames of the trajectory. Such trajectories can be directly visualized in a molecular graphics program, such as VMD (Humphrey *et al*., 1996). Furthermore, the interpolated structures can be analyzed for possible domain and shear movements with the DynDom package (Hayward and Berendsen, 1998), or used as initial seed structures for more advanced reaction path refinement methods such as Conjugate Peak Refinement (Fischer and Karplus, 1992).

3 SUMMARY

The bio3d comparative analysis results are in good agreement with descriptions established by human experts (Grant, 2004). In addition, the tools provide quantitative and visual information allowing for a more complete appreciation of interconformer relationships. Access to the open source software, full documentation, quick start guide and example data are available at http://mccammon.ucsd.edu/~bgrant/bio3d/

4 CONCLUSIONS AND PERSPECTIVES

The structure comparison procedures described here should facilitate the examination of diverse protein families, helping to identify common structural and dynamic features. Such analysis of structural homologues can provide invaluable conformational landmarks useful for assessing both new crystallographic structures and the results of theoretical methods. More generally, the current analysis methods may prove valuable to any study where knowledge of backbone flexibility must be modeled. For example, in flexible protein–protein docking and the generation of homology models where sampling along identified principal components may generate plausible alternative conformations. Another important area of research is deciphering possible networks of communication within proteins and, in particular, understanding allosteric mechanisms that appear to be preserved in distant relatives. Theoretical studies combined with comparative analysis of structural homologues are an initial step in this direction.

ACKNOWLEDGEMENTS

We would like to thank members of the Caves and McCammon groups for fruitful and entertaining discussions. This work was supported in part by the National Institutes of Health, National Science Foundation, the Howard Hughes Medical Institute, the National Biomedical Computation Resource and the National Science Foundation Center for Theoretical Biological Physics. Funding to pay the Open Access publication charges was provided by The Howard Hughes Medical Institute.

Conflict of Interest: none declared.

REFERENCES

Abseher,R. *et al*. (1998) Essential spaces defined by NMR structure ensembles and molecular dynamics simulation show significant overlap. *Proteins*, **31**, 370–382.

Caves,L.S.D. *et al*. (1998) Locally accessible conformations of proteins: multiple molecular dynamics simulations of crambin. *Protein Sci*., **7**, 649–666.

Elsawy,K.M. *et al*. (2005) The physical determinants of the DNA conformational landscape. *Nucleic Acids Res*., **33**, 5749–5762.

Fischer,S. and Karplus,M. (1992) Conjugate peak refinement: an algorithm for finding reaction paths and accurate transition states in systems with many degrees of freedom. *Chem. Phys. Lett*., **194**, 252–261.

Gerstein,M. and Altman,R.B. (1995) Average core structures and variability measures for protein families: application to the immunoglobulins. *J. Mol. Biol*., **251**, 161–175.

Grant,B.J. (2004) Kinesin sequence, structure and dynamics. PhD Thesis. University of York, York, UK.

Hayward,S. and Berendsen,H. (1998) Systematic analysis of domain motions in proteins from conformational change: new results on citrate synthase and T4 lysozyme. *Proteins*, **30**, 144–154.

Humphrey,W. *et al*. (1996) VMD: visual molecular dynamics. *J. Mol. Graph*., **14**, 33–38.

R Development Core Team (2006) R: a language and environment for statistical computing. Vienna, Austria.

van Aalten,D.M.F. *et al*. (1997) Protein dynamics derived from clusters of crystal structures. *Biophys. J*., **73**, 2891–2896.

BIOINFORMATICS APPLICATIONS NOTE Vol. 24 no. 11 2008, pages 1403–1405
doi:10.1093/bioinformatics/btn129

Genetics and population analysis

adegenet: a R package for the multivariate analysis of genetic markers

Thibaut Jombart*

Université de Lyon, Université Lyon 1, CNRS, UMR 5558, Laboratoire de Biométrie et Biologie Evolutive, France

Received on February 12, 2008; revised on April 2, 2008; accepted on April 4, 2008

Advance Access publication April 8, 2008

Associate Editor: Alex Bateman

ABSTRACT

Summary: The package *adegenet* for the R software is dedicated to the multivariate analysis of genetic markers. It extends the *ade4* package of multivariate methods by implementing formal classes and functions to manipulate and analyse genetic markers. Data can be imported from common population genetics software and exported to other software and R packages. *adegenet* also implements standard population genetics tools along with more original approaches for spatial genetics and hybridization.

Availability: Stable version is available from CRAN: http://cran.r-project.org/mirrors.html. Development version is available from *adegenet* website: http://adegenet.r-forge.r-project.org/. Both versions can be installed directly from R. *adegenet* is distributed under the GNU General Public Licence (v.2).

Contact: jombart@biomserv.univ-lyon1.fr

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

Genetic markers are now widely used in many fields of population biology, and can be analysed using various approaches. Among these, multivariate methods such as principal component analysis (PCA) are compelled to play an important role because they can summarize the genetic variability without making strong assumptions about an evolution model: they do not rely on Hardy–Weinberg equilibrium, nor do they suppose the absence of linkage disequilibrium. This is especially valuable when no or very little information is known about the system under study, as is frequent in landscape genetics (Manel *et al*., 2003). Recently, multivariate methods have proven useful to assess the consensus genetic structuring among a set of genetic markers (Laloë *et al*., 2007), as well as to investigate the spatial pattern of the genetic variability (Jombart *et al*., in press). However, multivariate methods currently available in population genetics software are very restricted, despite the fairly large number of these programs (Excoffier and Heckel, 2006). An exception to this is the R software (R Development Core Team, 2008) which contains both packages devoted to multivariate methods like *ade4* (Chessel *et al*., 2004; Dray *et al*., 2007), and packages dedicated to the analysis of genetic markers (http://cran.r-project.org/web/views/Genetics.html). Currently there are no bridges between multivariate analysis packages and genetic marker packages, and genetic markers data cannot be readily analysed using multivariate approaches. The purpose of *adegenet* is to build this connection. This package aims at extending the *ade4* package so that genetic markers can be analysed using multivariate methods. This is achieved by defining new classes of objects to represent genetic markers, and providing functions to import, export and manipulate these objects. Moreover, *adegenet* also implements some usual population genetics methods, as well as more original tools for spatial genetics and data simulation. This article presents an overview of these functionalities.

2 CONTENT

2.1 Data representation

Basic genetic markers data are genotypes obtained for a set of markers, each allele being coded by a character string (Warnes, 2003). In order to use statistical methods, such information cannot be used directly, and needs to be recoded numerically into a matrix of allelic frequencies. In *adegenet*, allelic frequencies of genotypes are stored inside objects of the class `genind`, which is the basic class of the package. In addition to allelic frequencies stored in a `@tab` component (the '@' designs a slot), a `genind` object stores other useful information whose description is provided by the R command `class?genind`. This class `genind` was designed to allow flexibility (it can virtually store any relevant information about genotypes) but also to be robust. As `genind` is a formal class (or 'S4 class' in R language), it is naturally robust: the content of an object is checked for validity when it is created and each time it is modified, which considerably limits the risks of having wrong or missing items in it. Moreover, `genind` internally uses generic labels for markers, alleles and genotypes, so that missing or redundant user-defined labels cannot originate an error in further analyses. Whenever the study involves groups of genotypes (say, 'populations') rather than genotypes, `genpop` objects are used. This formal class is very similar to `genind`, except that `@tab` contains counts of alleles per population instead of allelic frequencies of genotypes. Objects of both classes can be analysed by multivariate methods using

*To whom correspondence should be addressed.

© The Author 2008. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oxfordjournals.org

the @tab slot as input. Main available functions to import to, export from, manipulate and analyse genind and genpop objects are listed in Supplementary Material.

2.2 Functionalities

Great attention was devoted to developing input/output functions, because interoperability of data is crucial to facilitate data analysis. Until now, data could only be imported into R from FSTAT (Goudet, 2002) using the *hierfstat* package (Goudet, 2005). Currently, *adegenet* can read files from the software GENETIX (Belkhir *et al*., 1996–2004), STRUCTURE (Pritchard *et al*., 2000), FSTAT (Goudet, 2002), and Genepop (Raymond and Rousset, 1995), which are among the most common data formats in population genetics software (Excoffier and Heckel, 2006). Data can also be read inside R from a data.frame of genotypes coded by character strings (using df2genind), and exported back (genind2df). Outputs are possible from genind to the R packages *genetics* (Warnes, 2003) and *hierfstat* (Goudet, 2005), using genind2genotype and genind2hierfstat, respectively. Note that the data representation in the *genetics* package was intended to be consensual, and is used by many other R packages. Moreover, the output of genind2df is customisable and can be designed to fit usual formats like those of GENETIX or STRUCTURE.

To perform analyses at a population level, a genind object can be translated into a genpop object using genind2genpop. Other data manipulations include splitting information by marker (seploc) or by population (seppop), computing allelic frequencies for populations (makefreq), or subsetting genotypes, populations or alleles according to a given criterion. Basic methods are implemented such as the Hardy–Weinberg equilibrium test for all combinations of populations and markers (HWE.test.genind), a matrix of *P*-values allowing a quick overview of the results. Observed and expected heterozygosity, number of alleles by marker or population, sample sizes and other miscellaneous information are provided by summary functions. Missing data can be replaced in different ways—which is required by most statistical methods—using na.replace. Several genetic distances among populations can be computed using dist.genpop. Goudet's *G* statistic (Goudet *et al*., 1996) can be tested by a Monte–Carlo procedure to assess the hierarchical structuring of a set of genotypes (gstat.randtest).

The last goal of *adegenet* is to implement more original methods, either by extending existing ones, or by proposing new methods. Hybridization between individuals from two genind objects can be simulated using hybridize, which can be useful to evaluate the power of methods based on genetic differentiation. Monmonier's algorithm (Monmonier, 1973), which is used to infer genetic boundaries among geo-referenced genotypes (Manni *et al*., 2004), has been extended to include different degrees of connectivity among genotypes (monmonier) and implemented with an optimization function (optimize.monmonier). Finally, recently developed methods to investigate spatial patterns of the genetic variability (Jombart *et al*., in press) are also part of the package (functions spca, global.rtest and local.rtest).

3 EXAMPLE

This example illustrates how a theoretical hybrid population would appear on a typology provided by a multivariate method. First, we load the required packages, and the dataset microbov containing 30 microsatellite markers for 704 genotypes of 15 cattle breeds, described in Laloë *et al*. (2007).

```
> library(adegenet)
> library(ade4)
> data(microbov)
```

To simulate a hybrid population, two parent breeds (Salers and Zebu) are isolated:

```
> temp <- seppop(microbov)
> salers <- temp$Salers
> zebu <- temp$Zebu
```

The hybrid population ('Zebler') is obtained using the hybridize function (with n = 40 simulated genotypes). All data are pooled in a new object newbov:

```
> zebler <- hybridize(salers, zebu,
+ pop = "Zebler", n = 40)
> newbov <- repool(microbov, zebler)
```

Now we seek a typology displaying the diversity between breeds. For this, the inter-class PCA (Dolédec *et al*., 1987) is appropriate: this modification of PCA maximizes the variance between populations (here, breeds), instead of the total variance. Missing data are replaced (na.replace) before performing a centred PCA (dudi.pca) and an inter-class PCA (between):

```
> newbov <- na.replace(newbov, method =
"mean")
> pca1 <- dudi.pca(newbov$tab, center = TRUE,
+ scale = FALSE, scannf = FALSE)
> bet1 <- between(pca1, fac = newbov$pop,
+ scannf = FALSE, nf = 3)
```

The resulting typology (Fig. 1) is obtained by:

```
> s.class(bet1$ls, fac = newbov$pop,
+ clab = 1.2, lab = newbov$pop.names)
> add.scatter.eig(bet1$eig, nf = 2, xax = 1,
+ yax = 2, pos = "bottomright", csub = 1.2)
```

The first principal axis of the analysis (Fig. 1) differentiates African and French breeds, while the second axis expresses the genetic variability between African breeds. Interestingly enough, the simulated hybrid population (Zebler) appears between its parent populations (Salers and Zebu).

4 CONCLUSION

The first contribution of the R package *adegenet* is to implement classes and functions to facilitate the multivariate analysis of genetic markers. This led to define new formal classes for genotypes (genind) or groups of genotypes (genpop), which can be used as input to multivariate methods proposed in the R software. Several functions are also implemented to manipulate and analyse these objects, including recent development in spatial genetics and data simulation. By assuring a good interoperability of data, *adegenet* contributes to making the R software a unifying platform for the analysis of genetic markers.

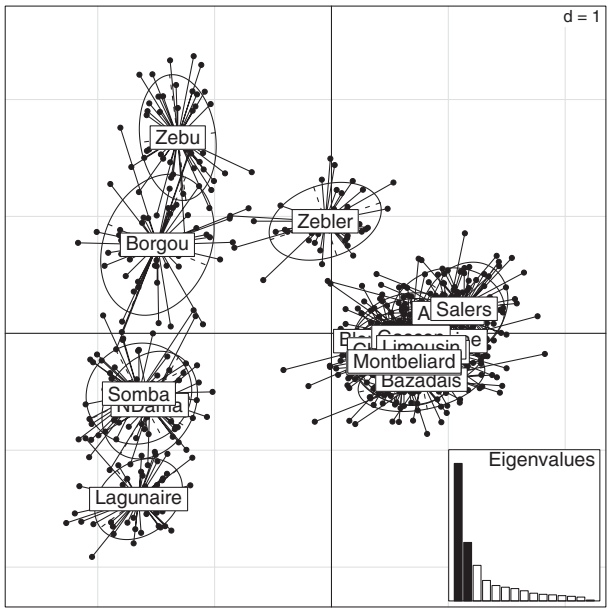

Fig. 1. Typology of cattle breeds (object `newbov`) obtained by inter-class PCA. Eigenvalues corresponding to the represented components are filled in black. Points represent genotypes; breeds are labelled inside their 95% inertia ellipses.

ACKNOWLEDGEMENTS

The author is grateful to R-Forge for hosting *adegenet*, to P. Sólymos for his contribution and to A.-B. Dufour, S. Devillard, D. Laloë and D. Pontier for their constructive comments.

Conflict of Interest: none declared.

REFERENCES

Belkhir,K. *et al*. (1996–2004) GENETIX 4.05, logiciel sous Windows TM pour la génétique des populations. Laboratoire Genome, Populations, Interactions, CNRS UMR 5000, Université de Montpellier II, Montpellier, France. URL: http://www.genetix.univmontp2.fr/genetix/intro.htm.

Chessel,D. *et al*. (2004) The ade4 package-I-one-table methods. *R News*, **4**, 5–10.

Dolédec,S. and Chessel (1987) Rythmes saisonniers et composantes stationnelles en milieu aquatique. *Acta Oecologica, Oecologia Generalis*, **8**, 403–426.

Dray,S. *et al*. (2007) The ade4 package – II: two-table and *K*-table methods. *R News*, **7**, 47–54.

Excoffier,L. and Heckel,G. (2006) Computer programs for population genetics data analysis: a survival guide. *Nat. Rev. Genet*., **7**, 745–758.

Goudet,J. (2002) Fstat 2.9.3.2. URL: http://www2.unil.ch/popgen/softwares/fstat.htm.

Goudet,J. (2005) Hierfstat, a package for R to compute and test hierarchical F-statistics. *Mol. Ecol. Notes*, **5**, 184–186.

Goudet,J. *et al*. (1996) Testing differentiation in diploid populations. *Genetics*, **144**, 1933–1940.

Jombart, T. *et al*. (in press) Revealing cryptic spatial patterns in genetic variability by a new multivariate method. *Heredity*.

Laloë,D. *et al*. (2007) Consensus genetic structuring and typological value of markers using multiple co-inertia analysis. *Genet. Sel. Evol*., **39**, 545–567.

Manel,S. *et al*. (2003) Landscape genetics: combining landscape ecology and population genetics. *Trends in Ecol. Evol*., **18**, 189–197.

Manni,F. *et al*. (2004) Geographic patterns of (genetic, morphologic, linguistic) variation: how barriers can be detected by ''Monmonier's algorithm''. *Hum. Biol*., **76**, 173–190.

Monmonier,M. (1973) Maximum-difference barriers: an alternative numerical regionalization method. *Geogr. Anal*., **3**, 245–261.

Pritchard,J. *et al*. (2000) Inference of population structure using multilocus genotype data. *Genetics*, **155**, 945–959.

R Development Core Team (2008) *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing, Vienna, Austria.

Raymond,M. and Rousset,F. (1995) Genepop (version 1.2): population genetics software for exact tests and ecumenicism. Journal of *Heredity*, **86**, 248–249.

Warnes, G. (2003) The genetics package. *R News*, **3**, 9–13.

Bioinformatics, 31(17), 2015, 2912–2914
doi: 10.1093/bioinformatics/btv300
Advance Access Publication Date: 11 May 2015
Applications Note

OXFORD

Data and text mining

GOplot: an R package for visually combining expression data with functional analysis

Wencke Walter[1], Fátima Sánchez-Cabo[2,*,†] and Mercedes Ricote[1,*,†]

[1]Department of Cardiovascular Development and Repair and [2]Bioinformatics Unit, Centro Nacional de Investigaciones Cardiovasculares (CNIC), Madrid, Spain

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the last two authors should be regarded as Joint Senior Authors.
Associate Editor: Jonathan Wren

Received on February 6, 2015; revised on April 22, 2015; accepted on May 5, 2015

Abstract

Summary: Despite the plethora of methods available for the functional analysis of omics data, obtaining comprehensive-yet detailed understanding of the results remains challenging. This is mainly due to the lack of publicly available tools for the visualization of this type of information. Here we present an R package called GOplot, based on ggplot2, for enhanced graphical representation. Our package takes the output of any general enrichment analysis and generates plots at different levels of detail: from a general overview to identify the most enriched categories (bar plot, bubble plot) to a more detailed view displaying different types of information for molecules in a given set of categories (circle plot, chord plot, cluster plot). The package provides a deeper insight into omics data and allows scientists to generate insightful plots with only a few lines of code to easily communicate the findings.
Availability and Implementation: The R package GOplot is available via CRAN-The Comprehensive R Archive Network: http://cran.r-project.org/web/packages/GOplot. The shiny web application of the Venn diagram can be found at: https://wwalter.shinyapps.io/Venn/. A detailed manual of the package with sample figures can be found at https://wencke.github.io/
Contact: fscabo@cnic.es or mricote@cnic.es

1 Introduction

Omics technologies have become standard tools in biological research for identifying and unraveling transcriptional networks, building predictive models and discovering candidate biomarkers. Gene/protein/metabolomic expression data is especially challenging for investigators due to its high-dimensional nature. Exploratory data analysis techniques are used to get a first impression of the important characteristics of the dataset and to reveal its underlying structure. Statistical analyses are then performed to detect subsets of elements owing the ability to provide valuable insight into the underlying patterns of the investigated biological process. One common approach is to map the molecules to their associated biological annotations, e.g. gene ontology (GO) terms, and to further perform an enrichment analysis. Although many visualization methods, tools (Supek *et al.*, 2011) and packages (Yin *et al.*, 2012; Zhang *et al.*, 2013) have been developed, none of them enables the user to combine expression data with the results of functional analysis in a way that guarantees the preservation of the power of both analyses.

R is commonly used by the *omics* community to analyze high-dimensional expression data. We therefore developed the R package `GOplot` based on the implementation of the grammar of graphics (Wickham, 2009). `GOPlot` follows the path of deductive reasoning to allow the user to go from the most general to the most specific details of the functional analysis results. The package implements novel, original, high-quality plots which also allow the integration of other quantitative information about molecules, i.e. the expression levels. `GOplot` improves understanding of *omics* data and aids the communication of biologically relevant findings in publications.

© The Author 2015. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

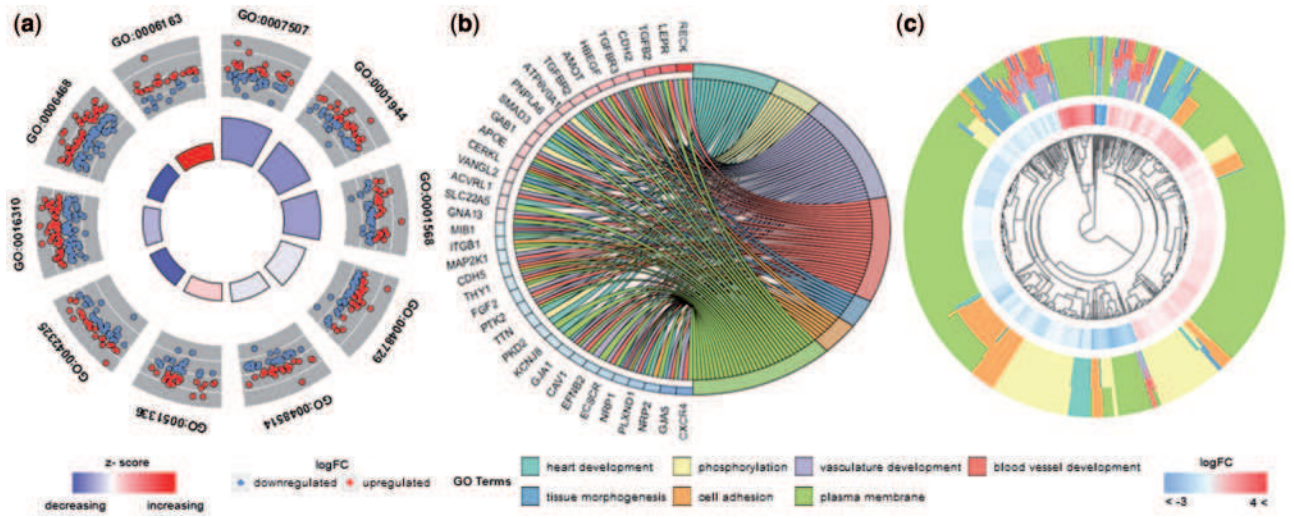

Fig. 1. (**a**) GOCircle plot; the inner ring is a bar plot where the height of the bar indicates the significance of the term ($-\log_{10}$ adjusted *P*-value), and color corresponds to the *z*-score. The outer ring displays scatterplots of the expression levels (logFC) for the genes in each term. (**b**) GOChord plot; the genes are linked via ribbons to their assigned terms. Blue-to-red coding next to the selected genes indicates logFC. (**c**) GOCluster plot displaying a circular dendrogram of the clustering of the expression profiles. The inner ring shows the color-coded logFC (up to three conditions), the outer ring the assigned functional terms

2 Package description

`GOplot` imports various R packages and was built on ggplot2, one of the three graphic systems in R. The package makes use of the complexity of `ggplot2` to provide the user with a collection of prespecified and multilayered charts. Each layer adds valuable information to the displayed context to convey the intended message. Two kinds of datasets are required as input: a list of selected molecules with their expression levels and the results of a functional analysis.

We implemented two kinds of function: preprocessing and plotting.

Preprocessing functions: Although using the preprocessing functions is not mandatory, it is highly recommended to ensure easy workflow. `circle_dat()` allows the user to easily combine expression and functional enrichment data and generates the appropriate input for most of the plotting functions. `chord_dat()` generates a binary matrix that assigns the molecules to each predefined functional term. This kind of binary matrix is used as input for `GOChord()`. The input format for the preprocessing functions can be checked with the help function in R.

Plotting functions: The package provides the user with a guide to explore the data and to select lists of elements and terms of interest. The exploratory part of the data analysis starts at a very general level with the `GOBubble()` and `GOBar()` plotting functions for comparative charts. Both charts display information about the significance of the enrichment ($-\log_{10}$ of the adjusted *P*-value) and the z-score of the term. `GOBar()` allows processes to be sorted by *z*-score or *P*-value to provide a better overview. The circles of the bubble plot are area-proportional to the number of molecules in the given category. Based on these charts a list of relevant terms can be selected. With `GOCircle()`, `GOChord()` and `GOCluster()`, the user can add quantitative molecular information to the terms of interest (Fig. 1).

With `GOVenn()` we have implemented a Venn diagram, that displays the number of overlapping elements as well as their expression patterns (commonly upregulated, commonly downregulated or contraregulated). In addition we have used shiny (http://CRAN.R-project.org/package=shiny), the web application framework for R, to create an interactive web application of `GOVenn()`.

3 Example

This section briefly exemplifies some of the functionalities of the `GOplot` package. Further details of the available functions and their usage can be found in the `GOplot` vignette.

`GOplot` comes with a manually compiled sample dataset (EC). Selected samples were downloaded from the Gene Expression Omnibus (accession number: GSE47067). The data were normalized and a statistical analysis was performed to determine differentially expressed genes. The DAVID functional annotation tool (Huang *et al.*, 2009) was used to perform a gene-annotation enrichment analysis of the set of differentially expressed genes (adjusted *P*-value < 0.05). Figure 1 shows three sample plots from the package created by the example code below.

```
> library(GOplot)
# Load the dataset
> data(EC)
# Generate the plotting object
> circ <- circle_dat(EC$david, EC$genelist)
# Generate the binary matrix
> chord <- chord_dat(circ, EC$genes, EC$process)
# Create the plots
> GOCircle(circ)
> GOChord(chord, ribbon.col = brewer.pal (7, 'Set3'))
> GOCluster(circ, EC$process)
```

Acknowledgements

We thank Manuel Gómez (Bioinformatics Unit, CNIC) for his valuable contributions in the initial steps of this work and Carlos Torroja (Bioinformatics Unit, CNIC) for his comments that helped to improve and refine the implementation. We thank S. Bartlett (CNIC) for editorial assistance.

Funding

This work was supported by grants from the Spanish Ministry of Economy and Competitiveness (SAF2012-31483), Fundación Marató TV3 and the

European Commision FP7 (CardioNext-ITN-608027) to M.R. The CNIC is supported by the Spanish Ministry of Economy and Competitiveness and the Pro-CNIC Foundation.

Conflict of Interest: none declared.

References

Huang,D.W. *et al*. (2009) Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources. *Nat. Protoc.*, **4**, 44–57.

Supek,F. *et al*. (2011) REVIGO summarizes and visualizes long lists of gene ontology terms. *PLoS one*, **6**, e21800.

Wickham,H. (2009) ggplot2: elegant graphics for data analysis. Springer-Verlag, New York.

Yin,T. *et al*. (2012) ggbio: an R package for extending the grammar of graphics for genomic data. *Genome Biol.*, **13**, R77.

Zhang,H. *et al*. (2013) RCircos: an R package for Circos 2D track plots. *BMC Bioinformatics*, **14**, 244.

MOLECULAR ECOLOGY
RESOURCES
Molecular Ecology Resources (2017) **17**, 27–32 doi: 10.1111/1755-0998.12509

SPECIAL ISSUE: POPULATION GENOMICS WITH R

POPHELPER: an R package and web app to analyse and visualize population structure

R. M. FRANCIS
Evolutionary Biology Centre, Uppsala University, Norbyvägen 18D, 75236 Uppsala, Sweden

Abstract

The POPHELPER R package and web app are software tools to aid in population structure analyses. They can be used for the analyses and visualization of output generated from population assignment programs such as ADMIXTURE, STRUCTURE and TESS. Some of the functions include parsing output run files to tabulate data, estimating *K* using the Evanno method, generating files for CLUMPP and functionality to create barplots. These functions can be streamlined into standard R analysis workflows. The latest version of the package is available on GITHUB (https://github.com/royfrancis/pophelper). An interactive web version of the POPHELPER package is available which covers the same functionalities as the R package version with features such as interactive plots, cluster alignment during plotting, sorting individuals and ordering of population groups. The interactive version is available at http://pophelper.com/.

Keywords: ADMIXTURE, CLUMPP, DISTRUCT, population structure, STRUCTURE, TESS

Received 19 October 2015; revision received 12 January 2016; accepted 16 January 2016

Introduction

The use of multilocus molecular markers to differentiate populations and to infer their genetic structure and composition is a powerful tool in ecology, conservation and landscape genetics (Sunnucks 2000; Falush *et al.* 2003). One of the most successful approaches employed is the assignment of individuals (Manel *et al.* 2005) which includes distance-based methods (Piry *et al.* 2004), frequency-based methods (Paetkau *et al.* 1995), Bayesian methods (Rannala & Mountain 1997; Baudouin & Lebrun 2000) and those involving Markov chain Monte Carlo methods (Cornuet *et al.* 1999; Pritchard *et al.* 2000; Paetkau *et al.* 2004). Several programs have been published in the recent years with various features and differences to investigate the population structure. Some of the softwares discussed in this article includes ADMIXTURE (Alexander *et al.* 2009), FASTSTRUCTURE (Raj *et al.* 2014), STRUCTURE (Pritchard *et al.* 2000) and TESS (François & Durand 2010). Along with assignment of individuals, it is often useful to cluster individuals into *K* populations. Estimation of the number of populations (*K*) is often a tricky problem, and no standard method exists that can be applied in all situations. The approach by Evanno *et al.* (2005) is one method to estimate the value of *K* and has been widely cited (Morgan *et al.* 2007; Blackburn & Maddison 2014; Diez *et al.* 2015).

Correspondence: Roy Mathew Francis, E-mail: roy.m.francis@outlook.com

With STRUCTURE, the input formats have been made convenient for users using data conversion tools such as GENALEX (Peakall & Smouse 2012), but the handling of output files have not been implemented in a convenient manner. Both of these programs also lack functionalities to generate high-quality graphics. Two common downstream approaches are to align assignment clusters across replicate runs using CLUMPP (Jakobsson & Rosenberg 2007) and to visualize the output using DISTRUCT (Rosenberg 2004). The use of these programs is often difficult and cumbersome. Therefore, a program to easily and quickly accept run files and generate outputs that can be directly used in downstream programs was a necessity.

The most popular of these 'helper' programs was STRUCTURE HARVESTER (Earl 2012). STRUCTURE HARVESTER is a web-based utility with a graphical user interface that can accept STRUCTURE runs to generate Evanno plots and input files for CLUMPP, but does not produce plots or allow to work with CLUMPP output. A more recent development was STRUCTURE PLOT (Ramasamy *et al.* 2014), which uses R shiny web framework (Chang *et al.* 2015) to interactively work in the browser. Some of the limitations of STRUCTURE PLOT include the lack of parsing run files, data tabulation, cluster alignment, CLUMPP export or Evanno method calculation. These programs also do not handle input from multiple softwares.

The statistical programming language R (R Development Core Team 2013) is widely used in data analysis

© 2016 John Wiley & Sons Ltd

and its use in ecology and population analysis has risen over the years. The program POPHELPER presented in this article is provided as an R package for command-line use and as a web app for interactive use. Many of the limitations of previously discussed applications have been rectified in this package which include parsing run files from various softwares, tabulation of runs, plotting barplots with population labels, sorting of individuals and ordering of populations, alignment of clusters and merging of repeats using CLUMPP, etc.

Features

The latest version of POPHELPER R package can be installed from the Github repository (https://github.com/royfrancis/pophelper) with installation instructions provided. A brief introduction on functions, usage and workflow are provided on the github page. A detailed demonstration is provided as an html vignette along with the package accessible from the github page. The POPHELPER package employs the ggplot (Wickham 2009) graphing package to create high-quality graphics. The POPHELPER package currently accepts ADMIXTURE, FASTSTRUCTURE, STRUCTURE and TESS runs as input. Files from other sources or modified data can be used as input as long as they are all numeric, tabular data delimited by white space, tab or comma. The files must be free of headers and decimal must be defined by dot.

The common scenario with most users is that they have a directory full of run files. POPHELPER provides

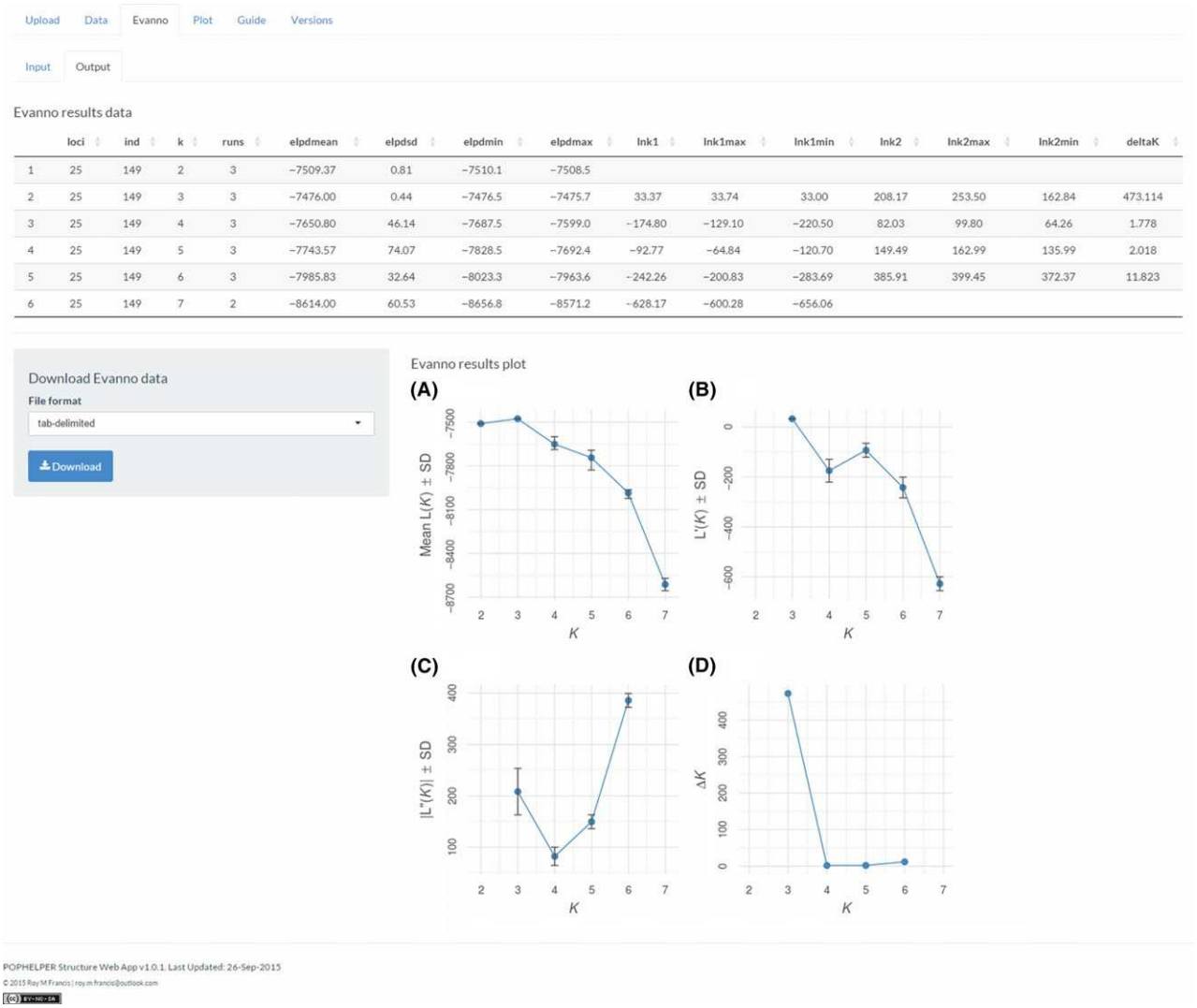

Fig. 1 A screenshot of the POPHELPER GUI for Evanno method using STRUCTURE runs. The table output is shown on the top, and the plot is shown below. The plot shows a typical Evanno analysis with (A) estimated log probability of data of runs over increasing values of *K*, (B) first derivative, (C) second derivative and (D) Δ*K* over values of *K*. The same plot is also produced in the R package version.

© 2016 John Wiley & Sons Ltd

functions to tabulate all runs listing the files and various parameters such as filename, *K* value, number of individuals and to summarize the runs by number of repeats. TESS files generated in separate directories can be easily collated into a single directory for further processing. ADMIXTURE, FASTSTRUCTURE, STRUCTURE, TESS or any tabular run files can be converted to R data frames for further analysis. The Evanno method to detect *K* has been implemented for STRUCTURE runs (Fig. 1). This method requires at least three sequential values of *K* with minimum of two repeats. Informative feedback is provided in cases where the data does not fulfil the requirements. The results are available as plots as well as text files. Input files for CLUMPP (combined file with repeats and the paramfile) are easily generated into separate directories. The CLUMPP executable can be used to generate the aligned and merged files in each *K* directory. The aligned and merged files can be collated into a single directory using POPHELPER. CLUMPP files can be generated from any of the input run files.

Barplots (Fig. 2) can be created from any of the input run files, combined files, aligned files or

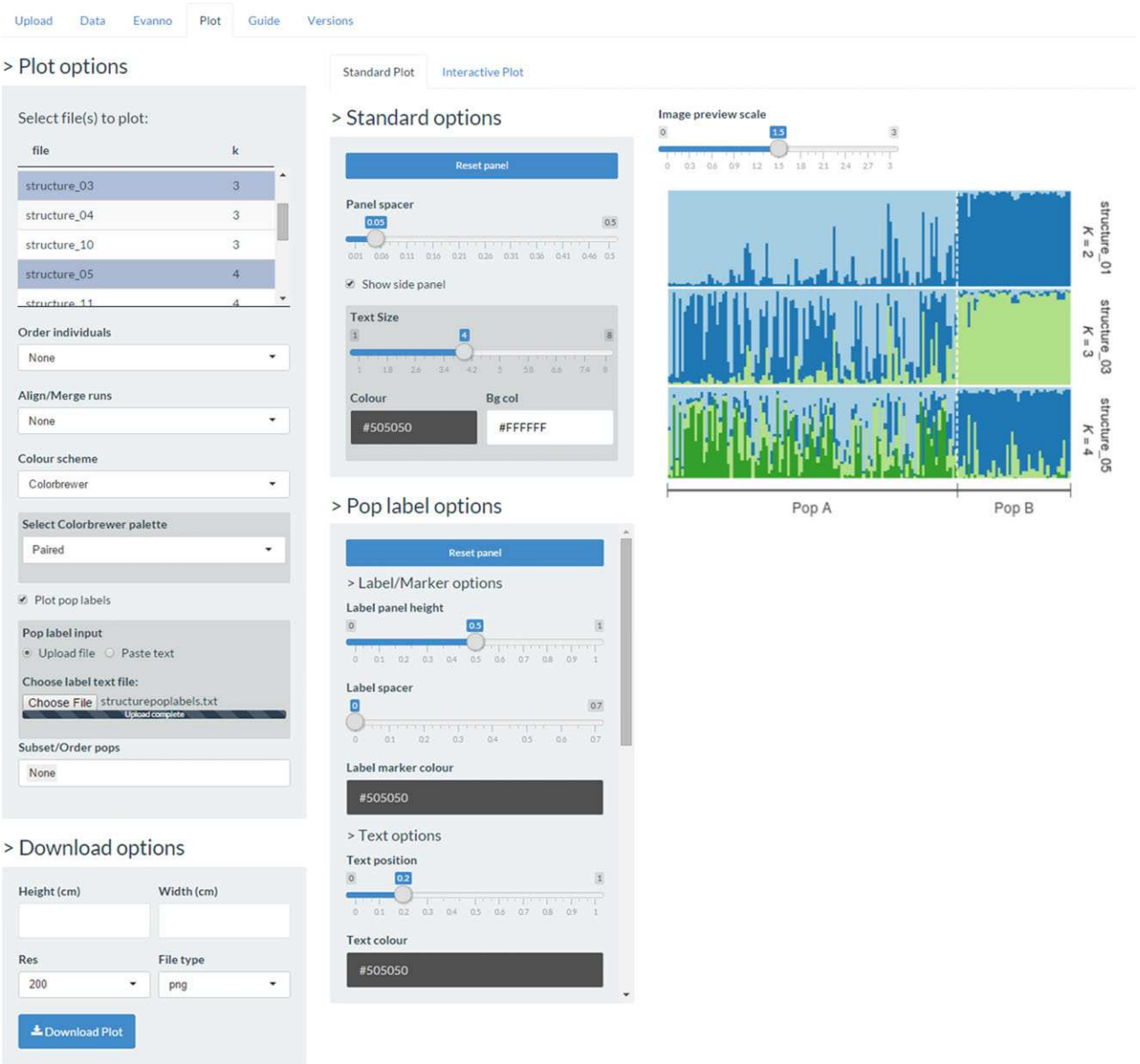

Fig. 2 A screenshot of the POPHELPER GUI for analysing and visualizing assignment run files. A typical barplot produced in POPHELPER is seen at top right. Three runs ($K = 2$, $K = 3$ and $K = 4$) are joined together as a single plot with common grouped labels. The filenames and *K* values are shown on the right-side panel. Function arguments or GUI options are provided to tweak and customize almost every aspect of the plot.

© 2016 John Wiley & Sons Ltd

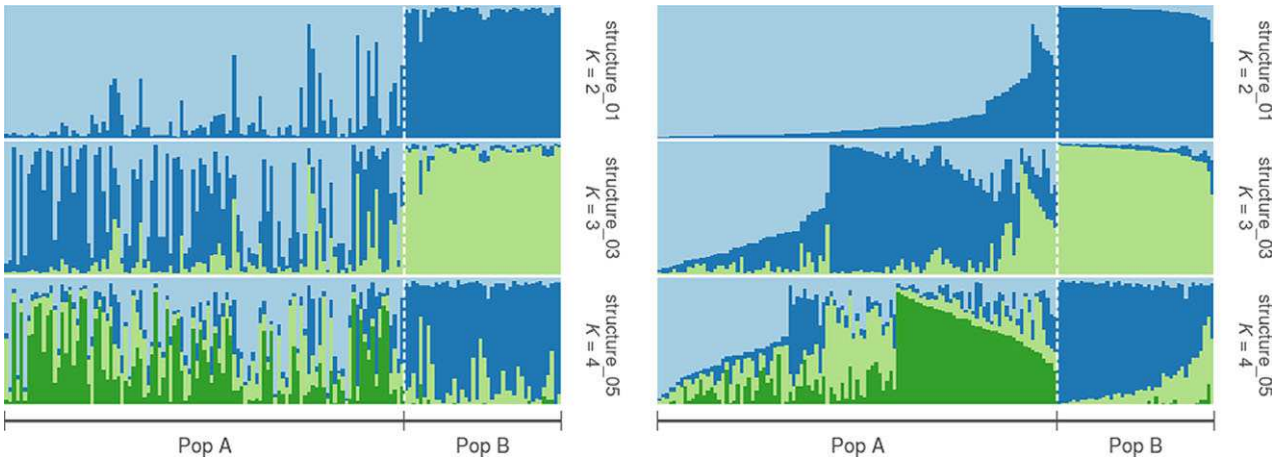

Fig. 3 Barplots created from STRUCTURE runs. Three runs ($K = 2$, $K = 3$ and $K = 4$) are joined together as a single plot with common grouped labels. The filenames and K values are shown on the right-side panel. Left: shows the input order of individuals. Right: shows individuals sorted by all clusters (similar to the 'Sort by Q' option in STRUCTURE software).

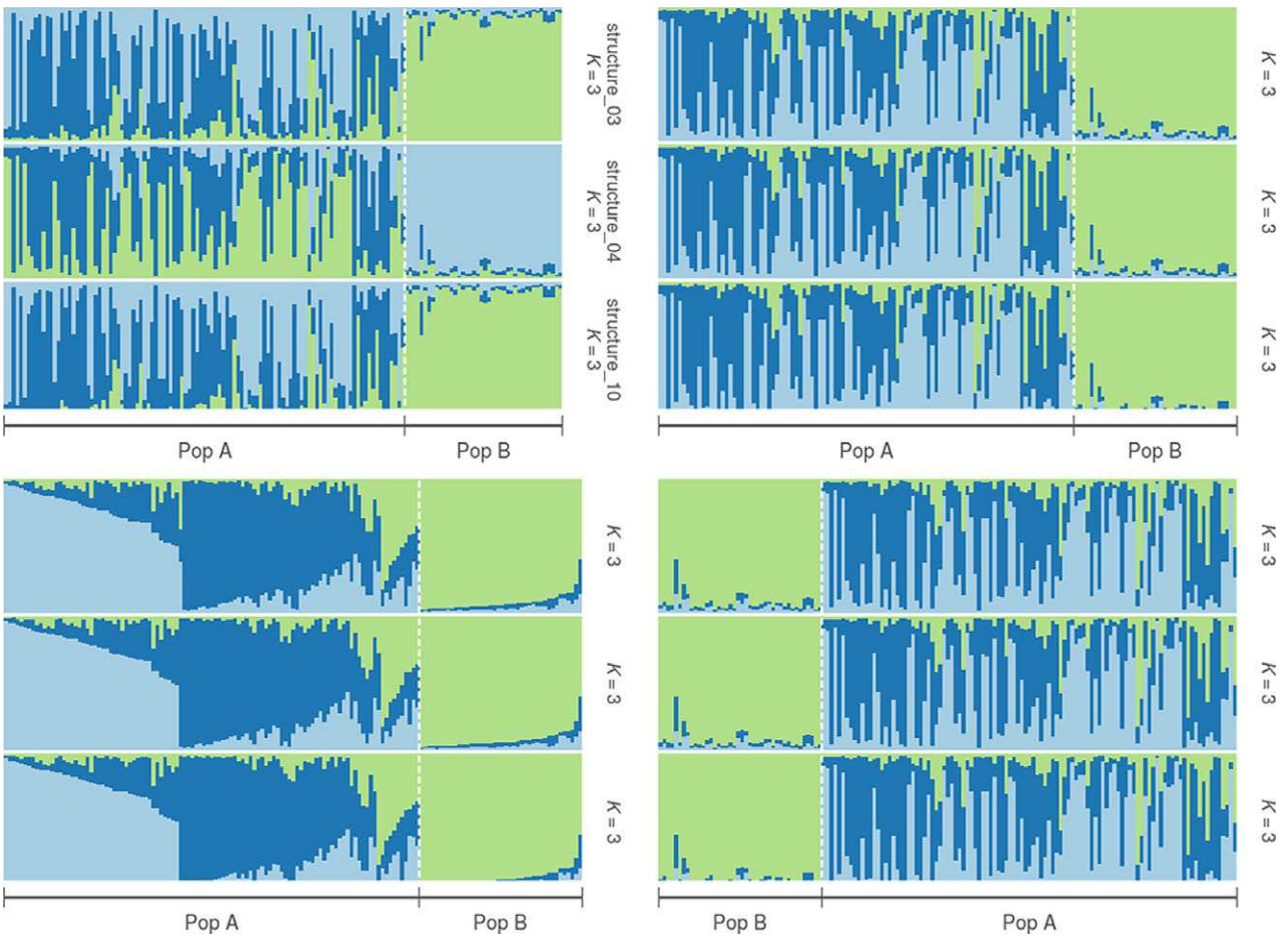

Fig. 4 Barplots created from STRUCTURE runs. Three repeats for $K = 3$ are joined together as a single plot with common grouped labels. Top left: shows the input in the original order. Top right: shows clusters aligned using CLUMPP. Bottom left: shows individuals sorted by 'all' clusters (similar to the 'Sort by Q' option in STRUCTURE software) within population groups after aligning using CLUMPP. Bottom right: shows reordered population groups (Pop B before Pop A) after aligning using CLUMPP.

© 2016 John Wiley & Sons Ltd

merged files. The plots can be created separately or as joined plots, with or without population labels (Fig. 3). The combined, aligned and merged files from CLUMPP can also be plotted (Fig. 4). Individuals in the barplot can be sorted by any one of clusters or a clustering that takes all clusters into account (similar to the 'Sort by Q' option in STRUCTURE software). When using population labels, population groups can be subsetted or reordered (Fig. 4). When using population labels along with sorting, individuals are sorted within the population groups. Multiline plots (similar to that implemented in STRUCTURE software) can be created from any of the previously mentioned file types to identify individuals. Multiline plots (Fig. 5) produce an A4 size image with a certain number of rows and certain number of individuals per row depending on the number of individuals. A reasonable default is calculated, but this can be adjusted. Multiline plots can be sorted by one or all clusters. The plot functions have several arguments to tweak and customize the figure as required. Export formats include JPEG (lossy), PNG (lossless) or PDF (vector). The package uses simple commands and basic R functions enabling access to beginner R users.

The web implementation of POPHELPER is available at https://pophelper.com. The web app was created in R using shiny (Chang *et al.* 2015) web framework and is suitable for users who wish to work interactively. The app makes use of several R packages such as fields (Nychka *et al.* 2015), RESHAPE2 (Wickham 2007), MARKDOWN (Allaire *et al.* 2015), GTABLE (Wickham 2012), GRIDEXTRA (Auguie 2015), PLYR (Wickham 2011), DT (Xie 2015), SHINYACE (LLC. TTa 2013), RCHARTS (Vaidyanathan 2013) and SHINYJS (Attali 2015). The web app has similar features to the R package. Interactive graphics are available for data exploration, and standard graphics (Fig. 2) are available for print and publication. The web app has an implementation of the Evanno method (Fig. 1) for determining K as well as cluster alignment during plotting using CLUMPP. Population labels can be uploaded as a text file or copy-pasted to mark populations under the barplot. Individuals can be sorted by one or more clusters. Population groups can be subsetted or reordered. As the cluster alignment is performed during plotting, CLUMPP and related files such as combined and aligned files are not required to be handled by the user. Population clusters can be coloured using a wide range of colour palettes including colour-blind friendly palettes. Options are provided to adjust the size, colour and position of most chart elements as well as population labels.

Detailed description of functionality and tutorials are available with the application (R package as well as the web app). The POPHELPER package is expected to be frequently updated, and new functionalities will be

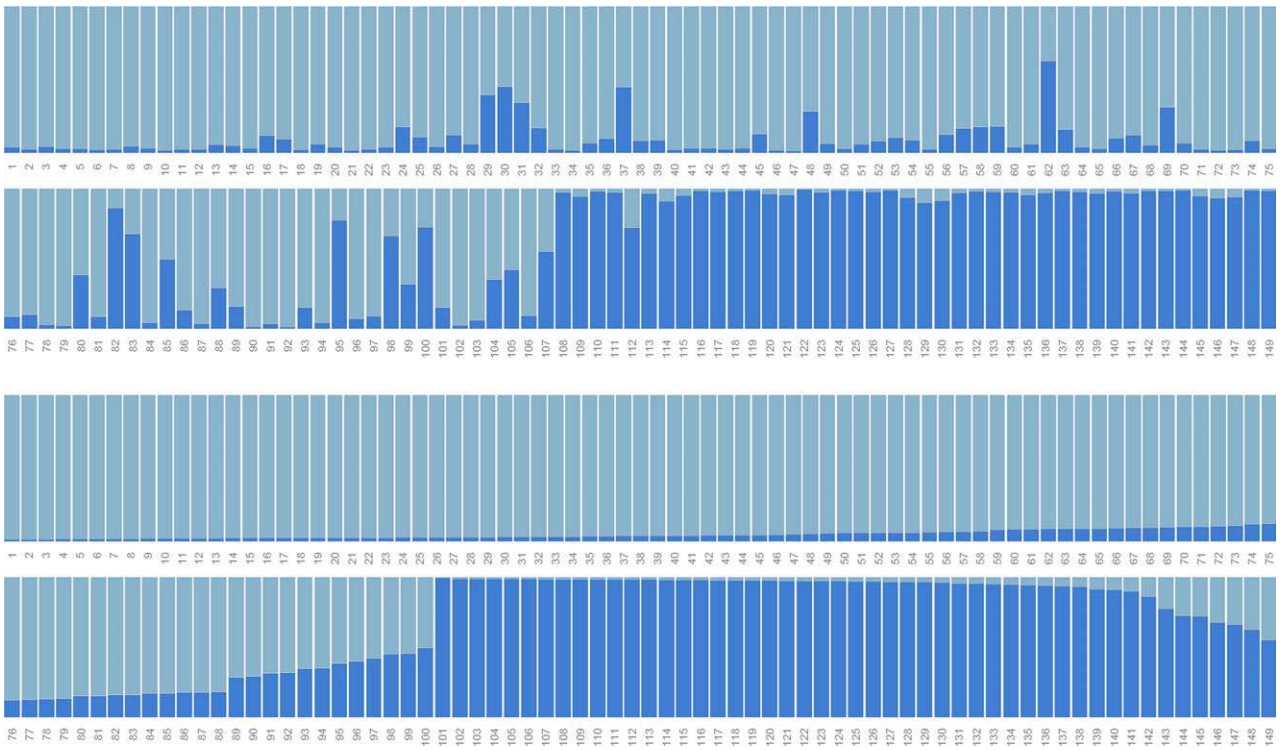

Fig. 5 Multiline barplots created from a STRUCTURE run file containing 149 individuals and $K = 2$. Top: shows individuals in the original order. Bottom: shows individuals sorted by 'all' clusters.

© 2016 John Wiley & Sons Ltd

added as and when possible. It is hoped that the R package will be useful for implementation into analysis workflows and the web app for interactive use.

Conclusion

The POPHELPER R package and web app are available to assist users working with molecular markers to investigate population structure. The R package is easy to use and possible to fit into workflows and data analysis pipelines. The interactive web app is available for applied users who wish to use a graphical user interface.

Acknowledgement

I wish to thank Per Kryger for the inspiration to embark on this project. I wish to thank the enthusiastic R community for the numerous tools and packages made available.

References

Alexander DH, Novembre J, Lange K (2009) Fast model-based estimation of ancestry in unrelated individuals. *Genome Research*, **19**, 1655–1664.

Allaire J, Horner J, Marti V, Porte N (2015) *markdown: 'Markdown' Rendering for R. R package version 0.7.7*. http://CRAN.R-project.org/package=markdown

Attali D (2015) *shinyjs: Perform Common JavaScript Operations in Shiny Apps using Plain R Code. R package version 0.2.0*. http://CRAN.R-project.org/package=shinyjs

Auguie B (2015) *gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.0.0*. http://CRAN.R-project.org/package=gridExtra

Baudouin L, Lebrun P (2000) An operational Bayesian approach for the identification of sexually reproduced cross-fertilized populations using molecular markers, 81–93.

Blackburn GS, Maddison WP (2014) Stark sexual display divergence among jumping spider populations in the face of gene flow. *Molecular Ecology*, **23**, 5208–5223.

Chang W, Cheng J, Allaire JJ, Xie Y, McPherson J (2015) *shiny: Web Application Framework for R*. http://CRAN.R-project.org/package=shiny

Cornuet JM, Piry S, Luikart G, Estoup A, Solignac M (1999) New methods employing multilocus genotypes to select or exclude populations as origins of individuals. *Genetics*, **153**, 1989–2000.

Diez CM, Trujillo I, Martinez-Urdiroz N *et al*. (2015) Olive domestication and diversification in the Mediterranean Basin. *New Phytologist*, **206**, 436–447.

Earl DA (2012) STRUCTURE HARVESTER: a website and program for visualizing STRUCTURE output and implementing the Evanno method. *Conservation Genetics Resources*, **4**, 359–361.

Evanno G, Regnaut S, Goudet J (2005) Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. *Molecular Ecology*, **14**, 2611–2620.

Falush D, Stephens M, Pritchard JK (2003) Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies. *Genetics*, **164**, 1567–1587.

François O, Durand E (2010) Spatially explicit Bayesian clustering models in population genetics. *Molecular Ecology Resources*, **10**, 773–784.

Jakobsson M, Rosenberg NA (2007) CLUMPP: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. *Bioinformatics*, **23**, 1801–1806.

LLC. TTa (2013) *shinyAce: Ace editor bindings for Shiny. R package version 0.1.0*. http://CRAN.R-project.org/package=shinyAce

Manel S, Gaggiotti OE, Waples RS (2005) Assignment methods: matching biological questions with appropriate techniques. *Trends in Ecology & Evolution*, **20**, 136–142.

Morgan JA, Vredenburg VT, Rachowicz LJ *et al*. (2007) Population genetics of the frog-killing fungus Batrachochytrium dendrobatidis. *Proceedings of the National Academy of Sciences of the United States of America*, **104**, 13845–13850.

Nychka D, Furrer R, Sain S (2015) *fields: Tools for Spatial Data. R package version 8.2-1*. http://CRAN.R-project.org/package=fields

Paetkau D, Calvert W, Stirling I, Strobeck C (1995) Microsatellite analysis of population structure in Canadian polar bears. *Molecular Ecology*, **4**, 347–354.

Paetkau D, Slade R, Burden M, Estoup A (2004) Genetic assignment methods for the direct, real-time estimation of migration rate: a simulation-based exploration of accuracy and power. *Molecular Ecology*, **13**, 55–65.

Peakall R, Smouse PE (2012) GenAlEx 6.5: genetic analysis in Excel. Population genetic software for teaching and research–an update. *Bioinformatics*, **28**, 2537–2539.

Piry S, Alapetite A, Cornuet JM *et al*. (2004) GENECLASS2: a software for genetic assignment and first-generation migrant detection. *Journal of Heredity*, **95**, 536–539.

Pritchard JK, Stephens M, Donnelly P (2000) Inference of population structure using multilocus genotype data. *Genetics*, **155**, 945–959.

R Development Core Team (2013) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna. http://www.R-project.org/

Raj A, Stephens M, Pritchard JK (2014) fastSTRUCTURE: variational inference of population structure in large SNP data sets. *Genetics*, **197**, 573–589.

Ramasamy RK, Ramasamy S, Bindroo BB, Naik VG (2014) STRUCTURE PLOT: a program for drawing elegant STRUCTURE bar plots in user friendly interface. *SpringerPlus*, **3**, 431.

Rannala B, Mountain JL (1997) Detecting immigration by using multilocus genotypes. *Proceedings of the National Academy of Sciences of the United States of America*, **94**, 9197–9201.

Rosenberg NA (2004) DISTRUCT: a program for the graphical display of population structure. *Molecular Ecology Notes*, **4**, 137–138.

Sunnucks P (2000) Efficient genetic markers for population biology. *Trends in Ecology & Evolution*, **15**, 199–203.

Vaidyanathan R (2013) *rCharts: Interactive Charts using Javascript Visualization Libraries. R package version 0.4.5*.

Wickham H (2007) Reshaping Data with the reshape Package. *Journal of Statistical Software*, **21**, 1–20.

Wickham H (2009) *ggplot2: Elegant Graphics for Data Analysis*. Springer, New York, New York.

Wickham H (2011) The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, **40**, 1–29.

Wickham H (2012) *gtable: Arrange grobs in tables. R package version 0.1.2*. http://CRAN.R-project.org/package=gtable

Xie Y (2015) *DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.1*. http://CRAN.R-project.org/package=DT

R.M.F. wrote the code for the program and the manuscript.

Data accessibility

The latest version of the package is available on Github (https://github.com/royfrancis/pophelper).

© 2016 John Wiley & Sons Ltd

Methods in Ecology and Evolution

British Ecological Society

Methods in Ecology and Evolution 2013, **4,** 782–788 doi: 10.1111/2041-210X.12067

APPLICATION

diveRsity: An R package for the estimation and exploration of population genetics parameters and their associated errors

Kevin Keenan[1], Philip McGinnity[2], Tom F. Cross[2], Walter W. Crozier[3] and Paulo A. Prodöhl[1]*

[1]*Institute for Global Food Security, School of Biological Science, Medical Biology Centre, Queen's University, Belfast, BT9 7BL, Northern Ireland;* [2]*Aquaculture & Fisheries Development Centre, School of Biological, Earth & Environmental Sciences, University College Cork, Cork, Ireland; and* [3]*Agri-Food and Biosciences Institute, Newforge Lane, Belfast, Northern Ireland*

Summary

1. We present a new R package, diveRsity, for the calculation of various diversity statistics, including common diversity partitioning statistics (θ, G_{ST}) and population differentiation statistics (D_{Jost}, G'_{ST}, χ^2 test for population heterogeneity), among others. The package calculates these estimators along with their respective bootstrapped confidence intervals for loci, sample population pairwise and global levels. Various plotting tools are also provided for a visual evaluation of estimated values, allowing users to critically assess the validity and significance of statistical tests from a biological perspective.

2. diveRsity has a set of unique features, which facilitate the use of an informed framework for assessing the validity of the use of traditional *F*-statistics for the inference of demography, with reference to specific marker types, particularly focusing on highly polymorphic microsatellite loci. However, the package can be readily used for other co-dominant marker types (e.g. allozymes, SNPs).

3. Detailed examples of usage and descriptions of package capabilities are provided. The examples demonstrate useful strategies for the exploration of data and interpretation of results generated by diveRsity. Additional online resources for the package are also described, including a GUI web app version intended for those with more limited experience using R for statistical analysis.

Introduction

As a consequence of the growing suite of statistical genetics tools, which are often tailored to particular marker types, the analyses of population genetic data are becoming an increasingly complex task (Excoffier & Heckel 2006). For instance, *F*-statistics is a commonly used framework for the description of genetic diversity partitioning within and among populations. *F*-statistics estimators (e.g. θ, G_{ST}) suffer from an incompatibility when applied to highly polymorphic microsatellite markers (Hedrick 1999; Jost 2008), as a result of their negative dependence on within subpopulation heterozygosity (Jost 2008). Thus, for loci with many alleles (e.g. >10), within subpopulation, heterozygosity will invariably be high, and as a consequence, 'traditional' *F*-statistics will have a theoretical maximum well below the expected $F_{ST} = 1$. Attempts have been made to overcome this issue, most notably by Hedrick (2005), with the development of G'_{ST} and more recently, Jost (2008) with the development of D_{Jost}. However, much confusion still exists about what these 'new' statistics should actually be used for (Gerlach *et al.* 2010). It is not the purpose of this study to elaborate on such issues; however, interested readers are encouraged to see Jost (2008), Meirmans & Hedrick (2011) and Whitlock (2011) for useful reviews.

To add to the complexity, recent advances in molecular screening methodologies have greatly facilitated the ease with which genetic data can be generated. As a consequence, an increasing number of researchers, often with a limited background in statistical genetics analyses (Karl *et al.* 2012), face the difficult task of analysing and interpreting such data. Thus, software tools that facilitate this task, by providing suitable frameworks to allow for informed analysis pipelines, are essential. To this end, we present the software diveRsity. This R package allows the estimation of various population genetic summary statistics including the two 'traditional' *F*-statistics analogues; θ (Weir & Cockerham 1984) and G_{ST} (Nei & Chesser 1983), and the two 'new' differentiation statistics; G'_{ST} (Hedrick 2005) and D_{Jost} (Jost 2008), as well as their unbiased/nearly unbiased estimators. Each statistic can be estimated for locus, global and sample pairwise comparisons. The package also provides functionality for the estimation of 95% confidence intervals at all relevant levels, through an integrated bootstrapping procedure. Uniquely to diveRsity, various plotting functions,

*Correspondence author. E-mail: p.prodohl@qub.ac.uk

© 2013 The Authors. Methods in Ecology and Evolution © 2013 British Ecological Society

designed to allow researchers to assess the validity of using their particular data set (or suite of marker loci) for the inference of geneflow using the *F*-statistics framework, are also provided, as well as visualisation tools for large pairwise matrices of genetic differentiation and parameter confidence intervals. Furthermore, `diveRsity` also provides a range of other statistical tools, which are commonly used in population genetic analyses pipelines, but are rarely integrated into a single software package.

Another major advantage of using `diveRsity` is that it produces summary data structures, which are very close to publication-ready formats (e.g. Fig. 1). Given that the compilation of such summary data is time consuming and often involves the use of several software packages, `diveRsity` offers a valuable addition to the molecular ecologist's statistical toolkit. Its implementation as an R package also makes `diveRsity` ideal for easy incorporation into analysis pipelines where batch processing of files/data is required, as is often the case in simulation-based studies.

This package is intended to promote a more considered and simplified approach to frequentist population genetic structure analyses. Through the inclusion of diversity partitioning statistics (e.g. θ & G_{ST}), differentiation statistics (e.g. G'_{ST} & D_{Jost}), as well as functionality to assess the behaviour of these statistics across loci and population samples, we hope to give researchers the necessary tools to make educated decisions about the statistical and biological validity of their analyses with relative ease. Following this rationale, we have also opted to omit the option for users to carry out *P*-value null hypothesis testing in relation to *F*-statistics and population sample differentiation estimators. This decision was taken given the lack of meaningful information conveyed through the use of *P*-values in this context, as well as the many misconceptions that exist regarding the biological interpretation of *P*-values in relation to these statistics (Wagenmakers 2007). We have instead provided functions to allow users to estimate 95% confidence intervals (calculated as the 2·5% and 97·5% quantiles of a bootstrap distribution), for a range of statistical estimators

pop1	Locus1	Locus2	Locus3	Locus4	Locus5	Overall
N	46	47	47	47	45	46
A	4	3	11	6	19	43
%	57·14	100·00	61·11	66·67	50·00	66·98
Ar	3·60	2·94	10·30	5·43	17·12	7·88
Ho	0·67	0·57	0·87	0·64	0·76	0·70
He	0·66	0·53	0·83	0·67	0·92	0·72
HWE	0·63	0·79	0·73	0·87	0·01	0·00
pop2	Locus1	Locus2	Locus3	Locus4	Locus5	Overall
N	40	42	42	42	42	42
A	5	2	13	7	20	47
%	71·43	66·67	72·22	77·78	52·63	68·15
Ar	4·85	2·00	10·78	5·87	17·17	8·13
Ho	0·65	0·48	0·74	0·52	0·90	0·66
He	0·66	0·50	0·79	0·71	0·92	0·72
HWE	0·53	0·76	1·00	0·53	0·92	0·00
pop3	Locus1	Locus2	Locus3	Locus4	Locus5	Overall
N	41	41	41	40	39	40
A	5	2	10	4	14	35
%	71·43	66·67	55·56	44·44	36·84	54·99
Ar	4·62	2·00	8·80	4·00	12·38	6·36
Ho	0·73	0·39	0·71	0·70	0·87	0·68
He	0·71	0·50	0·80	0·70	0·87	0·72
HWE	0·00	0·16	0·99	0·98	0·99	0·00

Fig. 1. A screenshot of the results output format from the function `divBasic`. This table format is commonly seen in journal articles when presenting basic population genetic parameters. However, the parameters often have to be calculated in separate software packages and tabulated by authors. `diveRsity` aims to reduce this requirement for authors. The parameter calculated in this table are; N = Number of individuals per population sample genotyped per locus, A = Total number of alleles observed per population sample per locus, % = Percentage of total alleles observed across population samples per population sample per locus, A_r = Allelic richness per locus, H_o = observed heterozygosity per locus, H_e = expected heterozygosity per locus, HWE = Hardy–Weinberg Equilibrium *P*-value from the χ^2 goodness-of-fit tests per locus.

calculated by the package, thus, leading to more reliable conclusions about the biological significance of trends in the data, (see Fig. 2 in du Prel *et al.* 2009), leaving less room for erroneous interpretation.

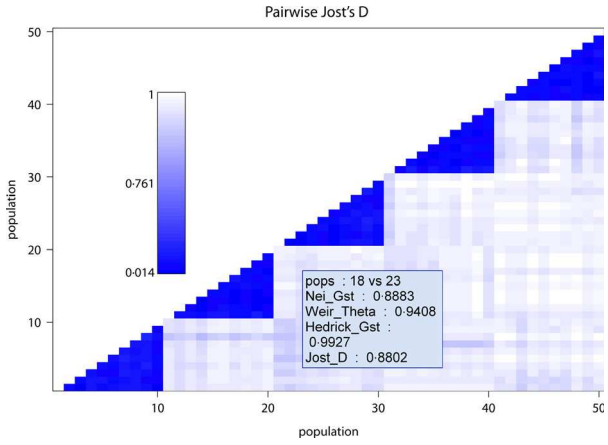

Fig. 2. Visualisation of pairwise D_{Jost} (estimator), for $N = 50$ populations. Total pairwise comparisons = 1225. This figure is returned from the `difPlot` function, which will plot diversity partitioning and differentiation estimators returned by `divPart`. Regions of dark blue represent low genetic differentiation, while light blue/white represents high differentiation. The text box caption is an example of the tooltip information associated with each pairwise population comparison.

Description

`diveRsity` is a package written for use in R (R Development Core Team 2012). It is primarily designed for the estimation, exploration and validation of genetic differentiation/structure indices. The package aims to consolidate under the same work environment, many of the most popular population genetic statistics such as those mentioned above, in order to provide researchers with a simplified way in which to calculate and compare these statistics. This strategy is particularly useful for the identification of polymorphism-based biases mentioned previously. This information can be subsequently used, along with additional exploration tools implemented in the package, to make informed decisions about which statistical measures or molecular markers can be appropriately applied to address a particular question.

`diveRsity` also calculates a plethora of other statistics and has various other population genetics applications. Table 1 provides a list of functions along with brief descriptions of their specific purposes. The package accepts raw genotype data for any group of co-dominant molecular markers in the *genepop* file format (Raymond & Rousset 1995). There is no limit to the size of the accepted input file other than the amount of random access memory (RAM) available to users. In addition to providing users with the ability to efficiently estimate an array of population genetic statistics, `diveRsity` is also particularly flexible in terms of return result formats (e.g. text files, excel

Table 1. Functions of the diveRsity package

Function	Returned objects	Description
`chiCalc`	R character matrix, optional.*txt* file	Test for genetic heterogeneity between population samples using the chi-square distribution. The function provides the unique option to disregard alleles of very low frequencies using the argument minFreq
`corPlot`	R graphics plot (not automatically written to file)	Correlation plotting of diversity statistics against the number of alleles per locus. The function is intended to aid in the assessment of marker suitability for the estimation of geneflow
`divPart`	*.html*,*.png*,*.txt*,*.xlsx*, R data object	A function for the calculation of diversity partition statistics and their associated variance through bootstrapping. Global, locus and pairwise levels are addressed
`divOnline`	NA	This function launches the web app version of divPart. Local resources are used when running analyses. The system default web browser is used to host the application
`difPlot`	*.html*,*.png*	Provides visualisation and exploration of pairwise genetic differentiation. The function is particularly useful for data sets containing a large number of population samples.
`inCalc`	*.png*,*.txt*,*.xlsx*, R data object	A function for the calculation of allele and locus informativeness for the inference of ancestry. Bootstrap confidence intervals are also calculated.
`readGenepop`	R data object	A general purpose function designed to calculate basic descriptive parameters from raw genetic data. This function is intended as a tool for developers of population genetics software in R.
`divRatio`	R data object,*.txt*, or.*xlsx*	This function calculates the diversity ratio statistics presented in (Skrbinšek *et al.* 2012)
`bigDivPart`	R data object,*.txt*, or.*xlsx*	This function is identical to divPart except for its lack of bootstrapping functionality. It is coded in a specific way to allow the sequential analysis of large number of markers (e.g. <100 000)
`fstOnly`	R data object,*.txt*, or.*xlsx*	This function calculates only Weir & Cockerham's 1984 *F*-statistics. The function is slightly faster than `divPart`, which also calculates these statistics
`divBasic`	R data object,*.txt*, or.*xlsx*	This function calculates basic population bases statistics such as Allelic richness, Hardy–Weinberg equilibrium and locus expected and observed heterozygosis

workbooks and native R objects such as matrices and data frames). This flexibility facilitates subsequent downstream analysis (e.g. incorporation into simulation or approximate bayesian computation (ABC) pipelines as the summary statistic calculation software). A list of specific output formats is also summarised in Table 1.

DEPENDENCIES AND SUGGESTED PACKAGES

In general, diveRsity can be used with a standard R installation and two additional extension packages (plotrix and shiny). The functions divPart, inCalc, chiCalc and readGenepop, divBasic, bigDivPart and divRatio, (i.e. the major analytical functions), can all operate independently of nonstandard packages. The only disadvantages of this approach are slower execution times (i.e. parallel computation is not available) and a limited number of formats available for returned results. To fully capitalise on the additional features of diveRsity (listed in Table 1), the installation of all suggested packages is recommended. Details of these packages are given in Table 2.

COMPARISONS WITH OTHER SOFTWARE

The main motivation behind the development of diveRsity was to provide a cross-platform software, which allows comprehensive and fast frequentist analysis of co-dominant molecular data, while maintaining usability and convenient result formats. On each of these aims, diveRsity performs comparatively better in relation to other similar software.

Comprehensiveness

When compared with other software which estimates similar statistics, diveRsity generally provides a more comprehensive range of parameter calculation options. In terms of the total number of available population genetics statistics, with the possible exception of the Mac OS X only program, GenoDive (Meirmans & Van Tienderen 2004), diveRsity estimates many more than DEMEtics (Gerlach *et al.* 2010), SMOGD (Crawford 2010), mmod (Winter 2012), hierfstat (Goudet 2004) or SPADE (Chao & Shen 2003).

Focusing only on diversity partitioning/differentiation statistics, diveRsity overlaps in its calculation of D_{Jost} with all of the above-mentioned software. However, diveRsity is the only package that allows the estimation of 95% confidence intervals, globally (i.e. for all samples and loci), per locus (i.e. over all samples) and for all pairwise sample comparisons (i.e. over all loci per population pair). SMOGD, for example, which is perhaps the most popular of these applications (with over 212 citations according to Google scholar), calculates bootstrapped confidence intervals for D_{Jost} at the locus level across all population samples, but does not provide this estimation for either the global or pairwise levels.

Despite the focus of this study on diversity partition/differentiation statistics, diveRsity also estimates many other useful population genetics statistics. These include, χ^2 tests of Hardy–Weinberg equilibrium (HWE), Allelic richness (A_r), Chi-square tests for sample homogeneity, 'Yardstick' diversity standardised ratios (Skrbinšek *et al.* 2012) and locus informativeness for the inference of ancestry (Rosenberg *et al.* 2003). Contrary to other similar programs, diveRsity also provides various exploratory plotting tools, which can be very useful for the identification of meaningful trends within results with minimal effort (e.g. Example 1). Typically, this task would involve the compilation of output results from various programs and subsequent visualisation in an independent software package (e.g. Microsoft Excel). A full description of diveRsity's functionality can be found by typing either of the following commands into the R console:

```
# diveRsity must be installed
# 1) package help pages
help(package ="diveRsity")
# 2) package user manual
vignette("diveRsity")
```

Speed

Given the different analytical focuses of distinct softwares, performance comparisons in terms of speed are not straightforward. For example, while in one software, a given test statistic might be estimated using a maximum likelihood procedure, in another, a more computational intensive procedure (e.g. bootstrapping) may be used. For the purposes of this study, com-

Table 2. Additional packages used by the diveRsity package, along with their implementations.

Package	Implementation	Status	Citation
Xlsx	Used in divPart and inCalc to return multisheet.*xlsx* workbooks	Suggested	Dragulescu (2012)
sendplot	Used in divPart, divPlot and inCalc to produce tooltips for data visualisation	Suggested	Gaile *et al.* (2012)
doParallel	Used in divPart and inCalc for parallel computation	Suggested	Revolution Analytics (2012a)
parallel	Used in divPart and inCalc for parallel computation	Suggested	R Development Core Team (2012)
foreach	Used in divPart and inCalc for parallel computation	Suggested	Revolution Analytics (2012b)
iterators	Used in divPart and inCalc for parallel computation	Suggested	Revolution Analytics (2012c)
plotrix	Used in difPlot for additional plotting features	Dependency	Lemon (2006)
shiny	Used to build and run the web app version of the divPart function	Dependency	RStudio & Inc (2012)

parisons were restricted to instances were distinct softwares implemented similar computational processes to calculate a similar suit of statistical parameters. Based on these criteria, only two truly comparable speed comparisons were possible between diveRsity and any of the above listed software.

The first is a comparison of locus confidence interval estimation using bootstrapping with SMOGD. The reproducible code used to run diveRsity is as follows:

```
system.time({
# load diveRsity
library("diveRsity")
# load Test_data
data(Test_data)
# run the analysis
x<-divPart(infile=Test_data, outfile=NULL, gp=3,
  pairwise = TRUE, WC_Fst = FALSE, bs_locus =
TRUE,
  bs_pairwise=FALSE, bootstraps=1000, plot=
FALSE,
  parallel=TRUE)
  })
```

When running SMOGD on the example data set Test_data (see Keenan *et al*. in press for details on these data), with bootstraps set to 1000, the time taken to return results to the web browser is 2 min 34·1 s, while diveRsity takes only 1 min 17·3 s to carry out the same calculations on a laptop with an Intel Core i5-2435 CPU @ 2·49GHz. It is also relevant to note that diveRsity's performance can be significantly increased with the use of additional CPUs.

The second comparison involves the calculation of diversity partitioning statistics per locus for large data sets (e.g. RAD-seq derived SNP genotypes). This comparison was carried out between the diveRsity function bigDivPart and the hierfstat function basic.stats. For this test, a simulated data set of 268 individuals across four population samples genotyped for 55 200 bi-allelic SNP loci was used. To complete the entire analysis, diveRsity took 3 min 20·1 s, while hierfstat took 6 min 44·8 sec, using the same laptop as described above. Such speed differences become even more important with the increasing rate at which large arrays of loci can be genotyped for large numbers of individuals.

Usability & convenience

Similar to other R packages, to fully benefit from all features built into diveRsity, a reasonable level of expertise in R is required. However, diveRsity has been designed so that even R beginners or those with very limited expertise can easily carry out comprehensive analysis of their data, including results being written to file, in many cases with a single command line. This is in contrast to other packages such as mmod and hierfstat, which invariably require users to export their own result from the R environment, as well as execute more functions to calculate fewer parameters than diveRsity. An example of the convenient results formats returned by diveRsity is shown in Fig. 1.

In keeping with the focus on ease of use, diveRsity also includes a web application, which provides a browser based user interface for the estimation of the most popular statistics implemented in the command line version of the package. This application was built using the framework provided by the R package, shiny (RStudio & Inc 2012), and provides users with a range of benefits including an easy to use interface and downloadable result files. The browser user interface also allows users to run their analyses on a remote server; thus, local system resources are not consumed. The application can be accessed at: http://glimmer.rstudio.com/kkeenan/diveRsity-online/.

Users can also run this application locally by executing the following command in the R console:

```
# after loading diveRsity
divOnline()
```

Despite an emphasis on simplicity, diveRsity still retains all of the functionality and flexibility provided by the R environment (i.e. all results are returned to the current session workspace). Thus, users with more experience can easily pipe results from their analyses into downstream custom analyses (e.g. ABC).

ACCESSING THE PACKAGE

The diveRsity package is hosted on the Comprehensive R Archive Network (CRAN), and can be downloaded using the install.packages function in R. Simply type the following command into the R console:

```
install.packages("diveRsity", dependencies =
TRUE)
```

Providing the user has a working internet connection, and following the selection of a suitable CRAN repository mirror, the package will download and install automatically.

Ongoing development of diveRsity can also be tracked at: http://diversityinlife.weebly.com/software.html

This web page contains the latest developmental versions of the package as well as an update log.

Examples

As a demonstration of some of the envisaged applications of diveRsity, two reproducible examples are provided below. These examples assume that the diveRsity, shiny, doParallel, sendplot and plotrix packages have been installed as well as their dependencies. For additional examples, users are encouraged to read the package manual.

EXAMPLE 1. USING VISUALISATION TOOLS TO INVESTIGATE LARGE GENETIC DIFFERENTIATION MATRICES

Pairwise genetic differentiation is an important parameter in the assessment of relationships among populations within a geographical context. To date, the true potential of pairwise genetic differentiation statistics has not been fully realised, owing mainly to difficulties in identifying

meaningful trends in often very large numbers of population comparisons.

However, using both the `divPart` and `difPlot` functions, `diveRsity` allows users to visualise large pairwise matrices of genetic differentiation, making the identification of particularly differentiated population samples relatively straightforward. This procedure is demonstrated below.

Load `diveRsity` into the current R session:

```
# Load the diveRsity package
require("diveRsity")
```

In this example, the `Big_data` data set (distributed with `diveRsity`) will be used. The data were simulated under a hierarchical island model (i.e. five island groups with 10 subpopulations each allowing high geneflow within island groups and low geneflow among island groups), using the software EASYPOP v1.7 (Balloux 2001). Population samples within the `Big_data` data file were arranged in order of geographical proximity for the purpose of demonstrating how `diveRsity` can be used to identify broad-scale geographical trends from genetic data.

```
# Load' Big_data'
data(Big_data, package ="diveRsity")
```

The `divPart` function is first used to calculate the required pairwise statistics matrices. In this example, the argument `parallel` will be set to `TRUE` as a large number of comparisons have to be computed (i.e. $[\frac{1}{2}N] \times [N-1] = 1225$ for $N = 50$).

```
# Assign the results to the variable' pwStats'
# (i.e. pw = pairwise)
pwStats <- divPart(infile = Big_data, outfile
="Big_results",
gp = 2, WC_Fst = TRUE, bs_locus = FALSE,
bs_pairwise = FALSE, bootstraps = 0,
Plot = FALSE, parallel = TRUE)
```

The resulting R object, `pwStats` contains the required pairwise statistics, which can be passed to the function `difPlot` for visualisation.

```
difPlot(x = pwStats, outfile ="Big_results",
  interactive = TRUE)
```

This command will write four.*png* files (one for each estimated statistic) and four.*html* files to the folder `Big_results` under the current R working directory. An example of the functionality of the.*html* tooltips is given in Fig. 2. From this figure, it is clear that the data are represented by five distinct genetic groups, which correlates with the simulation conditions described above. There are clearly high levels of differentiation among island groups (light blue/white) and low levels of differentiation within island groups (dark blue). This graphical representation perfectly relays what is known to be genetically/evolutionarily true (though natural population systems will rarely be so ideal).

Figure 2 also illustrates the ability to rapidly identify population pairs of interest by simply positioning the mouse pointer over a particular comparison square/pixel. In this example, the pairwise comparison between populations 18 vs. 23, (G_{ST} = 0·8883, θ = 0·9408, G'_{ST} = 0·9927 and D_{Jost} = 0·8802),

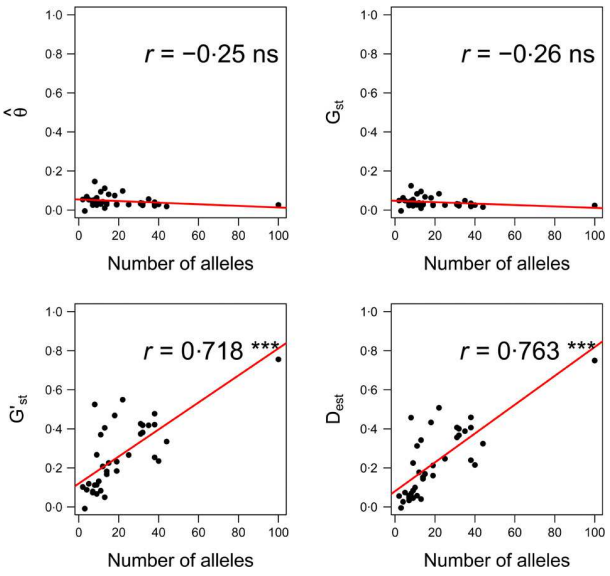

Fig. 3. Correlation assessment of locus estimators θ, G_{ST}, G'_{ST} and D_{est} (D_{Jost} unbiased estimator), with locus polymorphism (total number of alleles), returned from the `corPlot` function. Red lines represent the line of best fit and r values are Pearson product moment correlation coefficients.

indicates that these two populations are highly differentiated from one another.

EXAMPLE 2. ASSESSING POLYMORPHISM BIAS IN DIVERSITY PARTITIONING ESTIMATORS

As discussed above, diversity partitioning statistics such as G_{ST} and θ are negatively dependent on within subpopulation heterozygosity. Where this negative dependence is present (e.g. when using highly polymorphic microsatellites), it is important to ensure that inferences made from calculated values do not violate important assumptions. Using the functions `divPart`, `readGenepop` and `corPlot`, it is possible to carry out an *ad hoc* assessment of polymorphism bias in diversity statistics, thus allowing users to make informed decisions about whether to proceed with inference of demographic processes for example. A reproducible example is given below:

```
# Load the diveRsity package
require("diveRsity")
```

Next, an example data set (`Test_data`) provided with `diveRsity` should be loaded into the R session.

```
# Load' Test_data'
data(Test_data, package ="diveRsity")
```

Initially, `Test_data` is analysed by the function `divPart` to calculate locus θ, G_{ST}, G'_{ST} and D_{Jost} estimators.

```
  # Assign the results to the variable 'difStats'
difStats <- divPart(infile = Test_data, outfile
="Test",
  gp = 3, WC_Fst = TRUE, bs_locus = TRUE,
  bs_pairwise = FALSE, bootstraps = 1000,
  plot = TRUE, parallel = TRUE)
```

Next, `Test_data` is analysed by `readGenepop` to count the total number of alleles per locus.

```
# Assign the result to the variable 'numAlleles'
numAlleles <- readGenepop(infile = Test_data, gp = 3,
  bootstrap = FALSE)
```

The package has now generated two results objects in the R environment: `difStats` and `numAlleles`. These objects can be passed to the function `corPlot`.

```
corPlot(x = numAlleles, y = difStats)
```

Figure 3 provides an example of the output from this analysis. As can be seen in this example, both θ and G_{ST} are negatively correlated with the number of alleles per locus, while G'_{ST} and D_{Jost} are strongly positively correlated. This discordance is indicative of a case where the mutation rate is likely to obscure past demographic processes (e.g. geneflow); thus, such a data set is unsuitable for addressing such questions.

Users executing the above code will also see a range of other graphical outputs in a folder named '`Test`' within their working directory. These plots allow users to assess the variability of parameter estimation for individual loci, which can in turn be incorporated into decisions about 'misbehaving' loci for example.

Acknowledgements

The authors would like to thank J.J. Magee, M.S.P Ravinet, J. Coughlan and C. Johnston for testing the `diveRsity` package and R. Hynes for proofreading the manuscript. We would also like to express our gratitude to MEE executive editor Dr. Robert B. O'Hara and two anonymous reviewers, whose comments greatly improved the manuscript and the `diveRsity` package. K.K. was supported by a PhD studentship from the Beaufort Marine Research Award in Fish Population Genetics funded by the Irish Government under the Sea Change programme. P.A.P, T.F.C, W.W.C and P.McG were also supported by this award.

References

Balloux, F. (2001) EASYPOP (version 1.7): a computer program for population genetics simulations. *Journal of Heredity*, **92**, 301–302.

Chao, A. & Shen, T.J. (2003) Program SPADE (species prediction and diversity estimation). published at http://chao.stat.nthu.edu.tw. [accessed 27 March 2013]

Crawford, N.G. (2010) SMOGD: software for the measurement of genetic diversity. *Molecular Ecology Resources*, **10**, 556–557.

Dragulescu, A.A. (2012) *xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files*. http://cran.r-project.org/web/packages/xlsx/ [accessed 27 March 2013]

Excoffier, L. & Heckel, G. (2006) Computer programs for population genetics data analysis: a survival guide. *Nature Reviews Genetics*, **7**, 745–758.

Gaile, D.P., Shepherd, L.A., Sucheston, L., Bruno, A. & Manly, K.F. (2012) *sendplot: Tool for sending interactive plots with tool-tip content*. http://cran.r-project.org/web/packages/sendplot/ [accessed 27 March 2013]

Gerlach, G., Jueterbock, A., Kraemer, P., Deppermann, J. & Harmand, P. (2010) Calculations of population differentiation based on G_{ST} and D: forget G_{ST} but not all of statistics!. *Molecular Ecology*, **19**, 3845–3852.

Goudet, J. (2004) hierfstat, a package for R to compute and test hierarchical F-statistics. *Molecular Ecology Notes*, **5**, 184–186.

Hedrick, P.W. (1999) Highly variable loci and their interpretation in evolution and conservation. *Evolution*, **53**, 313–318.

Hedrick, P.W. (2005) A standardized genetic differentiation measure. *Evolution*, **59**, 1633–1638.

Jost, L. (2008) G_{ST} and its relatives do not measure differentiation. *Molecular Ecology*, **17**, 4015–4026.

Karl, S.A., Toonen, R.J., Grant, W.S. & Bowen, B.W. (2012) Common misconceptions in molecular ecology: echoes of the modern synthesis. *Molecular Ecology*, **21**, 4171–4189.

Keenan, K., Bradley, C.R., Magee, J.J., Hynes, R.A., Kennedy, R.J., Crozier, W.W., Poole, R., Cross, T.F., McGinnity, P. & Prodöhl, P.A. (2013) Beaufort Trout MicroPlex: a high throughput multiplex platform comprising 38 informative microsatellite loci for use in brown trout and sea trout (*Salmo trutta* L.) genetics studies. *Journal of Fish Biology*, **82**, 1789–1804.

Lemon, J. (2006) Plotrix: a package in the red light district of R. *R News*, **6**, 8–12.

Meirmans, P.G. & Hedrick, P.W. (2011) Assessing population structure: F_{ST} and related measures. *Molecular Ecology Resources*, **11**, 5–18.

Meirmans, P.G. & Van Tienderen, P.H. (2004) GENOTYPE and GENODIVE: two programs for the analysis of genetic diversity of asexual organisms. *Molecular Ecology Notes*, **4**, 792–794.

Nei, M. & Chesser, R.K. (1983) Estimation of fixation indices and gene diversities. *Annals of Human Genetics*, **47**, 253–259.

du Prel, J.-B., Hommel, G., Röhrig, B. & Blettner, M. (2009) Confidence interval or p-value? *Deutsches Ärzteblatt International*, **106**, 335–339.

R Development Core Team (2012) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna. http://www.R-project.org [accessed 27 March 2013]

Raymond, M. & Rousset, F. (1995) GENEPOP (version 1.2): population genetics software for exact tests and ecumenicism. *Journal of Heredity*, **86**, 248.

Revolution Analytics (2012a) *doParallel: Foreach parallel adaptor for the parallel package*. http://cran.r-project.org/web/packages/doParallel/ [accessed 27 March 2013]

Revolution Analytics (2012b) *foreach: Foreach looping construct for R*. http://cran.r-project.org/web/packages/foreach/ [accessed 27 March 2013]

Revolution Analytics (2012c) *iterators: Iterator construct for R*. http://cran.r-project.org/web/packages/iterators/ [accessed 27 March 2013]

Rosenberg, N.A., Li, L.M., Ward, R. & Pritchard, J.K. (2003) Informativeness of genetic markers for inference of ancestry. *American Journal of Human Genetics*, **73**, 1402–1422.

RStudio and Inc. (2012) *shiny: Web Application Framework for R*. http://cran.r-project.org/web/packages/shiny/ [accessed 27 March 2013]

Skrbinšek, T., Jelenčič, M., Waits, L.P., Potočnik, H., Kos, I. & Trontelj, P. (2012) Using a reference population yardstick to calibrate and compare genetic diversity reported in different studies: an example from the brown bear. *Heredity*, **109**, 299–305.

Wagenmakers, E.J. (2007) A practical solution to the pervasive problems of *p*-values. *Psychonomic Bulletin & Review*, **14**, 779–804.

Weir, B.S. & Cockerham, C.C. (1984) Estimating F-statistics for the analysis of population structure. *Evolution*, **38**, 1358–1370.

Whitlock, M.C. (2011) G'_{ST} and D do not replace F_{ST}. *Molecular Ecology*, **20**, 1083–1091.

Winter, D.J. (2012) mmod: an R library for the calculation of population differentiation statistics. *Molecular Ecology Resources*, **12**, 1158–1160.

Received 17 January 2013; accepted 1 May 2013
Handling Editor: Robert B. O'Hara

Methods in Ecology and Evolution

British Ecological Society

Methods in Ecology and Evolution 2013, **4,** 906–913 doi: 10.1111/2041-210X.12069

APPLICATION

pavo: an R package for the analysis, visualization and organization of spectral data

Rafael Maia[1]*, Chad M. Eliason[1], Pierre-Paul Bitton[2], Stéphanie M. Doucet[2] and Matthew D. Shawkey[1]

[1]*Department of Biology, Integrated Bioscience Program, University of Akron, Akron, OH, 44325–3908, USA; and* [2]*Department of Biological Sciences, University of Windsor, 401 Sunset Avenue, Biology Building, Windsor, Ontario, N9B 3P4, Canada*

Summary

1. Recent technical and methodological advances have led to a dramatic increase in the use of spectrometry to quantify reflectance properties of biological materials, as well as models to determine how these colours are perceived by animals, providing important insights into ecological and evolutionary aspects of animal visual communication.

2. Despite this growing interest, a unified cross-platform framework for analysing and visualizing spectral data has not been available. We introduce `pavo`, an R package that facilitates the organization, visualization and analysis of spectral data in a cohesive framework. `pavo` is highly flexible, allowing users to (a) organize and manipulate data from a variety of sources, (b) visualize data using R's state-of-the-art graphics capabilities and (c) analyse data using spectral curve shape properties and visual system modelling for a broad range of taxa.

3. In this paper, we present a summary of the functions implemented in `pavo` and how they integrate in a workflow to explore and analyse spectral data. We also present an exact solution for the calculation of colour volume overlap in colourspace, thus expanding previously published methodologies.

4. As an example of `pavo`'s capabilities, we compare the colour patterns of three African glossy starling species, two of which have diverged very recently. We demonstrate how both colour vision models and direct spectral measurement analysis can be used to describe colour attributes and differences between these species. Different approaches to visual models and several plotting capabilities exemplify the package's versatility and streamlined workflow.

5. `pavo` provides a cohesive environment for handling spectral data and addressing complex sensory ecology questions, while integrating with R's modular core for a broader and comprehensive analytical framework, automated management of spectral data and reproducible workflows for colour analysis.

Key-words: animal communication, colour, colourspace, receptor noise, reflectance, sensory ecology, spectrometry, visual model

Introduction

The role of colouration and colour vision in animal communication has been a fundamental question in evolutionary biology for many decades (Darwin 1859, 1896; Poulton 1890; Bennett & Théry 2007). Studies on visual communication have shed light on various aspects of natural (Chittka & Menzel 1992) and sexual selection (Hill 2002), and how these interact (Kemp *et al.* 2009). It is also an ideal system for truly integrative biological research, spanning from the optical processes generating colour (Shawkey *et al.* 2009), hormonal and genetic mechanisms regulating phenotype (Muller & Eens 2009), physiological processes involved in perceiving the signal (Hart 2001), and its adaptive and evolutionary patterns (Badyaev & Hill 2003; Darst *et al.* 2006).

*Correspondence author. E-mail: rm72@zips.uakron.edu

However, 'colour' refers to a sensory experience, not an objective quantity, and the realization that animals can vary quite considerably in their visual system and how they process this information prompted two important methodological advances. First, it highlighted the need for an objective quantification of the energy reflected at different wavelengths, as a first approximation of a 'receiver-independent' measure of an organism or object's colour (Endler 1993; Eaton & Lanyon 2003; Bennett & Théry 2007). Over the last 20 years, the rising popularity of portable spectrometers has made objective quantification of the spectral properties of animal and plant integuments commonplace (Endler 1990; Eaton & Lanyon 2003; Andersson & Prager 2006). Second, advances in the understanding of perception and processing of colour have allowed analysis of reflectance data using visual models that estimate how animals see and differentiate these colours (Goldsmith 1990; Tovee 1995; Vorobyev & Osorio 1998; Vorobyev *et al.* 1998).

© 2013 The Authors. Methods in Ecology and Evolution © 2013 British Ecological Society

Because of these advances, a cohesive framework for working with and analysing colour from reflectance data is needed. Output file types from spectrometer manufacturers are not standardized, and though existing software programs have provided helpful implementations of several methodologies (Gomez 2006; Stoddard & Prum 2008; Montgomerie 2006), they are often limited in the number or types of methodologies implemented, the types of data they can import and process, or the platforms in which they are available. Moreover, many are proprietary and/or closed, hindering customization (for a review of available software, see Montgomerie 2006). Furthermore, once the relevant colour data are extracted, they often require additional conversion and export into statistical software for analyses, which precludes protocol standardization across laboratories, batch processing, and automation of workflows.

Here, we introduce `pavo` – a package for R (R Development Core Team 2013) that addresses these problems by providing a flexible, yet cohesive, environment in which researchers can organize, analyse and visualize colour data generated by spectrometry. R is open source and multi-platform and is rapidly becoming the working language for scientific programming and data analysis, particularly in ecology and evolution (e.g. Paradis *et al.* 2004; Bolker 2008). `pavo` incorporates R's flexibility using object classes that can seamlessly interpret each other, providing functions that can be used to import, explore, process, and analyse spectral colour data under a variety of user-defined models. We propose that combining these procedures under a coherent framework not only streamlines workflow, but also allows data to be explored and manipulated in ways that can be used to visualize patterns, obtain information, and develop and test hypotheses (Fig. 1).

The `pavo` package

The stable release of `pavo` is available from CRAN (http://CRAN.R-project.org/package=pavo) for direct installation from R, and the development version is available from github (https://github.com/rmaia/pavo). `pavo` was developed with three main workflow stages in mind (Fig. 1): organization of spectral data by inputting raw files and processing their spectral content; visualization of the output, including exploratory capacities to identify further required manipulations and previously unconsidered patterns; and analysis of data from the spectral shape of reflectance curves or by incorporating receiver psychophysiology in visual models. As noted by Bennett & Théry (2007) and others (Andersson & Prager 2006; Montgomerie 2006), though spectral data have become commonplace in studies of animal colouration, it is easy to obtain poor-quality or inaccurate data. Therefore, a workflow for spectral

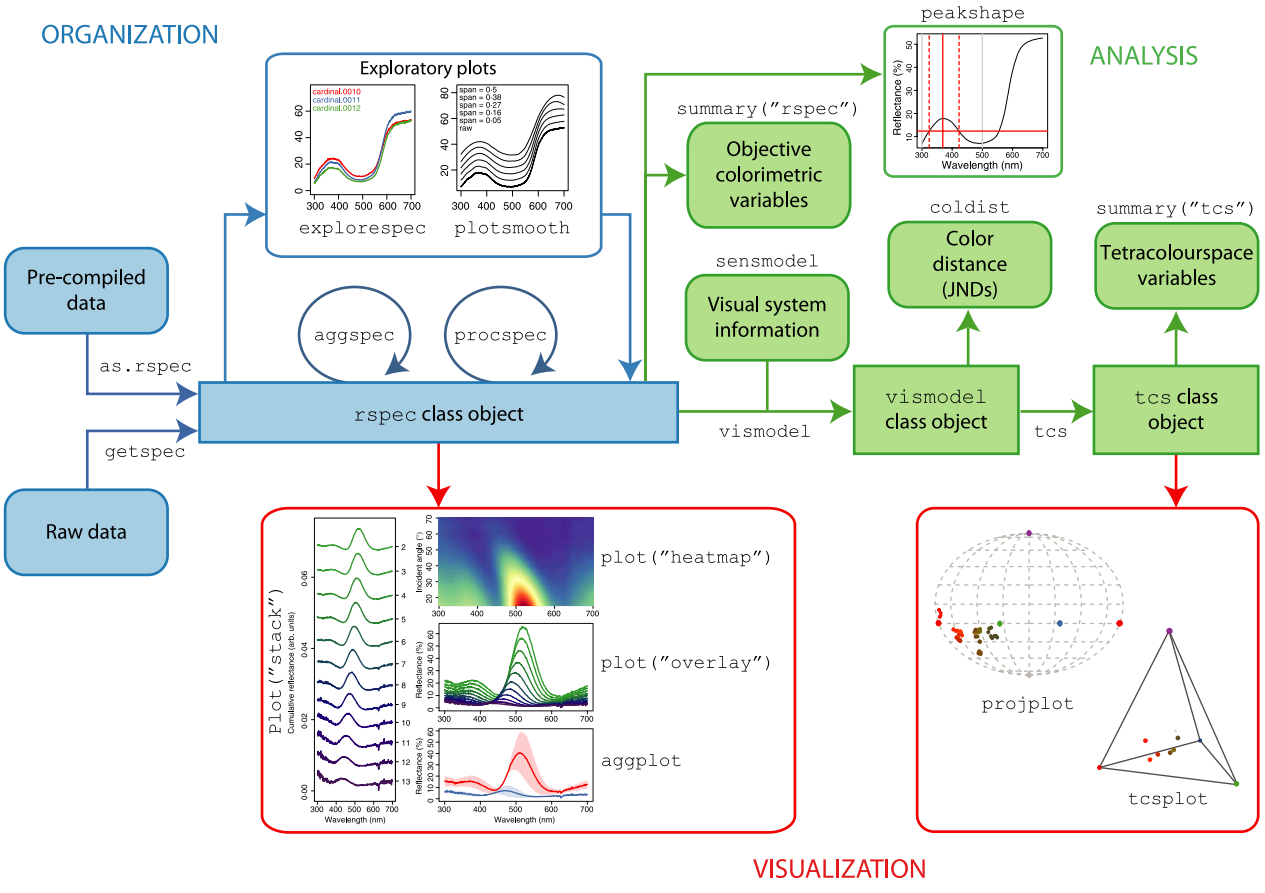

Fig. 1. Example `pavo` workflow, highlighting its main functions and plotting capabilities.

colour data analysis has to go beyond a 'plug and chug' implementation, requiring thorough exploratory investigation. With this in mind, `pavo` takes advantage of R's object-oriented programming environment to implement modular functions that allow each of these steps to be explored and to work cohesively. Below we outline `pavo`'s organization, visualization and analysis capabilities, followed by a detailed worked example using African glossy starlings to demonstrate some of its functionalities.

ORGANIZATION

Spectral data are stored in `pavo` and recognized for its functions by use of a new object class, `'rspec'`, which inherits methods from `data.frame`. Objects of class `rspec` are characterized by having individual reflectance spectra as columns of the data frame, with a first column containing the associated wavelength values. Raw spectral data can be imported using the function `getspec`, which currently supports data from a variety of software (including Ocean Optics OOIBase and SpectraSuite files, Avantes AvaSpec and CRAIC). In addition, spectral data that have been previously compiled (into a spreadsheet, for example) can be imported into R and converted to `rspec` objects using the `as.rspec` function.

The use of dedicated R object classes allows generic functions such as `plot` and `summary` to identify the object as a particular type of data frame and interpret it accordingly (see below). The class `'vismodel'` is used to interpret spectral data that have been processed through one of the visual models implemented and also stores information on how it was generated (e.g. the visual phenotype, background and illuminant used, and any transformations applied; see below). Additionally, the `'tcs'` class refers to tetrahedral colourspace models (Endler & Mielke 2005; Stoddard & Prum 2008), and the `summary` function can be used to extract summary variables, such as the colour volume or hue span (Stoddard & Prum 2008) for groups and subsets of points.

It is common when collecting spectral data to take multiple measurements from the same sample, averaging these to avoid sampling error (Quesada & Senar 2006). `pavo` provides the `aggspec` function for this purpose, as well as the `procspec` function for noise removal via smoothing, and transformations to standardize and clean spectral data.

VISUALIZATION

With `pavo` installed and loaded, the `plot` function recognizes `rspec` objects and plots them accordingly – interpreting the first column as wavelengths (usually using it as the x axis) and the remaining columns as reflectance values (y axis) for individual spectra. Several plotting options for multiple spectra are implemented (Fig. 1). In addition, the `aggplot` function provides plotting capabilities for among-spectra summary statistics plotting (Fig. 1), with a similar syntax to `aggspec`. `pavo` also offers exploratory plotting capabilities that can be combined with data processing and formatting, such as `explorespec` (for visualizing groups of spectra) and `smoothplot` (for choosing smoothening parameters; Fig. 1).

Finally, `pavo` offers plotting capabilities for the avian tetrachromatic colourspace model (Stoddard & Prum 2008; Endler & Mielke 2005) through the `tcsplot` and `projplot` functions (Fig. 1, see below).

ANALYSIS

The `summary` function can be applied to `rspec` objects to extract several objective ('receiver-independent') reflectance shape variables, relevant to specific or universal mechanisms of colour signalling (e.g. pigments, structures) or colour perception (spectral intensity, location and purity). Description and discussion of these variables can be found in Andersson & Prager (2006) and Montgomerie (2006), as well as in the package vignette. Additionally, the function `peakshape` provides descriptors of spectral peaks, such as the wavelength of maximum reflectance and the full width at half-maximum, and can be fine-tuned to extract information from specific areas of the curve. This implementation can be useful when the spectral curve has multiple peaks or a complex shape (e.g. the UV peak of carotenoid curves, Fig. 1).

`pavo` also allows the easy production of models that incorporate the visual system of the receiver through the `vismodel` function. Models can be calculated incorporating the visual phenotype (cone absorbance), background colour, and ambient illuminant (Vorobyev & Osorio 1998). Several avian receptor phenotypes (Hart 2001; Endler & Mielke 2005) are implemented as options, but user-defined receptor data from any taxon can be used as model input. Further, the `sensmodel` function implements the calculation of cone absorbance curves based on peak sensitivity information (available from the literature, for example Hart 2001) and can also include oil droplet and ocular transmission information in the calculations (Govardovskii *et al.* 2000; Hart & Vorobyev 2005).

Visual models can be calculated in terms of absolute photon catches, in which case the receptor noise model can be used to infer contrast between colours (implemented in the function `coldist` Vorobyev & Osorio 1998), or in relative cone stimulation, in which case the model reduces to a colourspace model represented in $n-1$ dimensions (where n is the number of different receptors involved in colour vision; Goldsmith 1990; Endler & Mielke 2005; Stoddard & Prum 2008). Absolute or relative cone stimulation can be selected by the logical argument `relative` from the `vismodel` function. In the case of the avian tetrahedral colourspace, several additional variables can be calculated based on spherical coordinates which represent the hue angles and saturation (the distance from the achromatic centre; see Stoddard & Prum 2008) by calling the `tcs` function. This function generates an object of class `tcs`; a `summary` call from a `tcs` object will return summary statistics described in Stoddard & Prum (2008) for sets of points (see below).

`pavo` also builds upon previously described visual model methods. For example, Stoddard & Stevens (2011) presented the useful technique of calculating the overlap between the

volumes defined by two sets of points in colourspace. They used this metric to quantify mimicry (Stoddard & Stevens 2011; Stoddard 2012), such that a greater volume overlap would indicate greater overall colour similarity. Given the complexity of calculating the intersection of three-dimensional convex hulls, Stoddard & Stevens (2011) used a Monte Carlo approach to estimate the degree of volume overlap. `pavo`, instead, provides the exact solution for the calculation of the intersection of colour volumes using a method originally implemented to calculate the overlap between multidimensional niches (Villéger *et al*. 2011) through the computational geometry capabilities available from the `rcdd` package (Geyer *et al*. 2012) (For performance and precision comparison, see Supporting Information).

Worked example: colour divergence in glossy starlings (Sturnidae)

Glossy starlings from the African clade have bright and diverse iridescent colours likely used in courtship displays and social competition (Rubenstein & Lovette 2009). Here, we demonstrate some of `pavo`'s capabilities through a worked example to compare the colours of a monophyletic clade of three glossy starling species, two of which (*Lamprotornis chloropterus* and *L. elisabeth*) have recently diverged and whose status as full species is debated (Gill & Donsker 2012; Craig *et al*. 1998; Lovette & Rubenstein 2007). The data used for this example have been deposited online and is available from the Dryad Database (Maia *et al*. 2013). Further information and a 'how-to' of `pavo`'s main functions, including the formulae used for the calculations, can be found in detail in the package vignette – available online from CRAN or directly from `R` using the command `vignette('pavo')`.

STEP 1: ORGANIZATION AND PROCESSING

The data consist of reflectance spectra in Avantes '`.ttt`' output format, which in this example are located in the folder '`/Desktop/glossystarlings`' (by default, the current working directory and the '`.txt`' extension are used). The files are named to indicate the species, bird ID, plumage patch and measurement ID (for example, '`LAAC.14423.belly.001.ttt`' would be the first measurement taken from the belly of individual 14423, a *Lamprotornis acuticaudus* specimen). This is important because `pavo` uses the file name (minus the file extension) to label the columns of the `rspec` object with the imported reflectance spectra. We recommend that users carefully consider and adhere to a rigorous file naming scheme to make subsetting and data manipulation easier (see Andersson & Prager 2006 for a useful discussion and suggestions in this regard). Additionally, we suggest that sample identity labels should have the same number of characters, which simplifies character string manipulation and subsetting based on partial matching.

We measured reflectance spectra from 11 plumage patches (see Fig. 2 for list) of four males from museum specimens of each of the three species. Three measurements from different locations within each patch were collected. We used the `getspec` function to load these 396 raw spectral data files, then used the `aggspec` function to average the spectra within patches (as determined by the `by` argument) and the `procspec` function to remove electrical noise arising from the spectrometer (using local polynomial regression fitting, or loess) with the following annotated lines of code:

```
#get raw data
>specs <- getspec(where ='~/Desktop/
glossystarlings', ext='ttt', lim=c(300,700))
396 files found; importing spectra
========================================
#average by groups of 3 spectral curves
>specs <- aggspec(specs, by=3, FUN="mean")
#remove electrical noise using
#Gaussian smoothing
>specs <- procspec(specs, opt="smooth")
processing options applied:
 smoothing spectra with a span of 0.25
```

STEP 2: VISUALIZATION

Next, we plotted spectra contained in the resulting `rspec` object. We used `aggplot` to visualize the mean reflectance curves for each body part from each species, as can be seen in the example below for the 'belly' body patch (Fig. 2b):

```
#subset wavelength column and the
#12 spectra from the belly patch
>specs.belly <- subset(specs, "Belly")
#extract first 4 characters from column names
#(species labels)
>spp <- substr(names(specs.belly), 1, 4)
#average and plot spectral data by species
>aggplot(specs.belly, by=spp)
```

STEP 3: ANALYSIS

Colour distances

To explore how these colours may be perceived by birds, we first used the `vismodel` function, which takes into account avian visual sensitivities (sensory phenotype), to calculate the quantum catches for each photoreceptor. `pavo` allows for user-defined cone sensitivities, but also incorporates available data from several taxa, including the European starling (*Sturnus vulgaris*, `visual="star"`, Hart *et al*. 1998), which we used for our example. We used the `relative = FALSE` option from the `vismodel` function to obtain raw photon catch values for the four avian photoreceptor classes ({*usml*}), suitable for calculating chromatic distances ΔS, Vorobyev *et al*., 1998).

`vismodel` can return either the calculated quantum catches (Q_i, the default) or values transformed according to Fechner's law (f_i; the signal being proportional to the logarithm of the quantum catch), as determined by the argument `qcatch` (not used in the code below, hence the default, Q_i, is returned). It can also apply the von Kries transformation (normalizing by

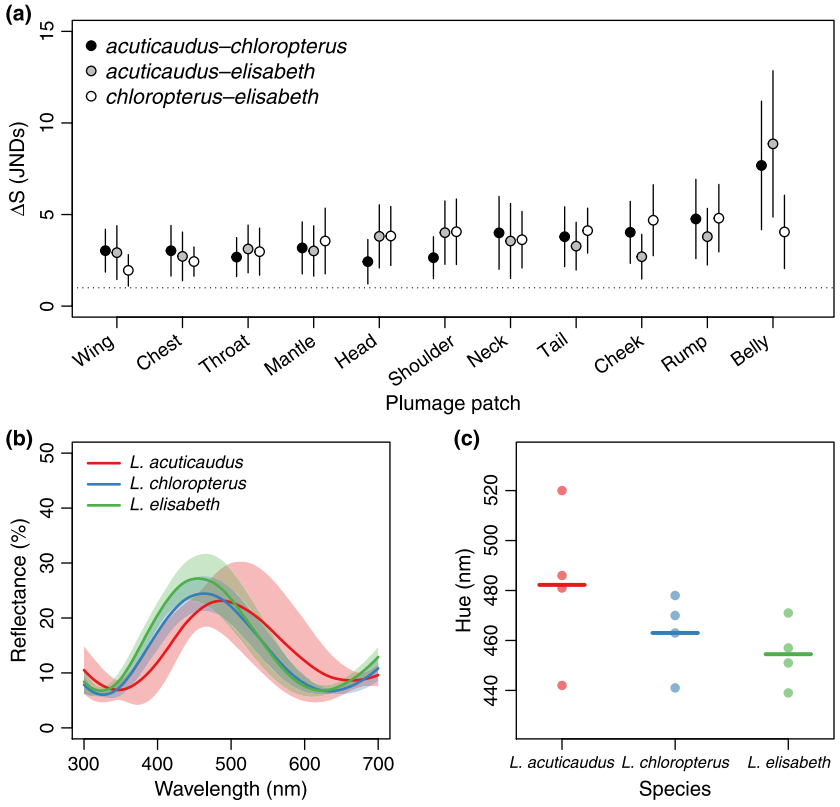

Fig. 2. (a) Plot showing colour distances (in units of just noticeable differences, JNDs) by patch (*y*-axis) for three pairs of African starling species (Sturnidae). The dotted horizontal line indicates JND = 1, above which the pair of colour patches is considered to be distinguishable by birds. Points and error bars indicate mean ± standard error chromatic distances between different pairs of species: *Lamprotornis chloropterus* and *L. elisabeth* (open circles); *L. acuticaudus* and *L. elisabeth* (grey circles); and *L. acuticaudus* and *L. chloropterus* (black circles). (b) Plot of mean smoothed spectra for the belly body patch. Line colours indicate species (green: *L. acuticaudus*, red: *L. chloropterus*, blue: *L. elisabeth*), and shaded areas indicate the standard deviation of the spectral data. (c) Hue (wavelength of maximum reflectance) values for each specimen measured (points) and mean per species (bars) for the belly patch ($N = 4$ individuals each; colours as in b).

the stimulus' background to account for receptor adaptation) through the `vonkries` argument (which defaults to `FALSE`). These settings are stored as arguments in the `vismodel` object, are referenced by functions downstream in the workflow (e.g. `coldist`, below), and can be called using the `summary` function:

```
>vm.star <- vismodel(specs, visual="star",
relative = FALSE)
>summary(vm.star)
visual model options:
* Quantal catch: Qi
* Visual system: star bt.dc
* Illuminant: ideal, scale = 1 (von Kries color
correction not applied)
* Background: ideal
* Relative: FALSE
```

Second, we used the `coldist` function to calculate colour distances with receptor noise based on the relative photoreceptor densities (Vorobyev & Osorio 1998). We used relative cone abundances (arguments `n1`, `n2`, `n3` and `n4`) for the European starling (Hart *et al.* 1998), and the `v` argument was set to 0·1 to give a {'*l*'}-cone Weber fraction (calculated as $v/\sqrt{n_i}$) of approximately 0·05 (Vorobyev & Osorio 1998; Vorobyev *et al.* 1998). The results of `coldist` give colour distances (chromatic, Δ*S*, shown in Fig. 2; and achromatic, Δ*L*) between all possible combinations of plumage patches in a `vismodel` object and can be further subsetted (using the `subset` argument) to focus on comparisons of interest (e.g. colour difference between homologous patches of two species, or between colour patches and a given background). This argument employs regular expressions functionality and thus allows for partial string matching of row names to the rule specified in the argument. In this case, if we want comparisons between a single type of patch, the `subset` argument will have two values to match both `patch1` and `patch2` columns:

```
>deltas <- coldist(vm.star, n1=1, n2=1.38,
n3=3.34, n4=3.46, v=0.1)
#subset only comparisons between wing spectra
>deltas.wing <- coldist(vm.star, n1=1, n2=1.38,
n3=3.34, n4=3.46, v=0.1, subset=c("Wing", "Wing"))
>head(deltas.wing, 3)
          patch1          patch2       dS       dL
1 LAAC.260046.Wing LAAC.264607.Wing 2.661180 8.714508
2 LAAC.260046.Wing LAAC.347959.Wing 2.790763 3.033686
3 LAAC.260046.Wing LAAC.347960.Wing 5.158210 5.629931
```

We can see from Fig. 2a that the recently divergent *L. chloropterus* and *L. elisabeth* have accumulated similar levels of colour disparity as they have to their sister species, *L. acuticaudus*. Considering a value of 1 as a threshold for Just Noticeable Differences (JNDs, Fig. 2a dashed line), nearly all plumage patch comparisons yield discernible colours, both within the *L. chloropterus* – *L. elisabeth* subclade as well as compared with the *L. acuticaudus* outgroup.

Spectral analysis

Another strength of `pavo` is its ability to compare different colour metrics (Butler *et al.* 2011), such as those extracted directly from the spectral curves (Fig. 2b) and thus do not consider

receiver visual system phenotype (see Andersson & Prager (2006) for discussion). As many as 23 objective colourimetric variables are calculated by the `summary` of an `rspec` object (for a complete description and summary of formulae, see the package vignette and Montgomerie 2006), but the `subset` argument can be used to filter these variables. If `subset = TRUE`, the variables described in Andersson & Prager (2006) are returned, but a string of variable names can also be used as a filter. For our example, we extracted and plotted an objective metric of hue (wavelength of peak reflectance, Fig. 2c) for the most contrasting body patch ('belly", Fig. 2a) in the visual model analysis:

```
# extract colour variables
>colr <- summary(specs.belly, subset=TRUE)
> head(colr, 3)
                          B2         S8    H1
LAAC.260046.Belly   18.92453   1.518254   520
LAAC.264607.Belly   12.31255   1.257920   481
LAAC.347959.Belly   11.20595   1.233001   486
# extract hue (H1)
> hue <- colr$H1
# extract species names
#(first 4 letters from row names)
>spp <- substr(rownames(colr), 1, 4)
# make box plots of hue (H1) by species
> boxplot(hue ~ spp)
```

Avian tetrahedral colourspace

To analyse these spectra in avian tetrahedral colourspace, we used `vismodel` again, but now specifying `relative=TRUE` to indicate that {*usml*} values should be scaled to sum of unity for each observation. We then used the `voloverlap` function to calculate the volumes occupied by each species' plumage patches, as well as their overlap. The `plot=TRUE` option provides a useful graphical representation of the overlap in colourspace (Fig. 3).

```
>vm.star.rel <- vismodel(specs, visual="star",
relative=TRUE)
> tcs.star <- tcs(vm.star.rel)
#subset points from the two species
> tcs.lael <- subset(tcs.star, "LAEL")
> tcs.lacl <- subset(tcs.star, "LACL")
> voloverlap(tcs.lael, tcs.lacl, plot=TRUE)
```

Conclusions and future directions

`pavo` implements colour analyses for reflectance data based on both spectral properties and visual system receptor stimulation, as well as a modular framework that allows the assumptions and parameters of such models to be tuned as necessary. As such, it is particularly suited to compare and validate the use of different colourimetrics (Butler *et al.* 2011), determine the influence of various assumptions in visual models, or help choose parameter values that would introduce the least bias in visual models (Hanley 2013). `pavo` can also be used to investigate the information content of colour signals using

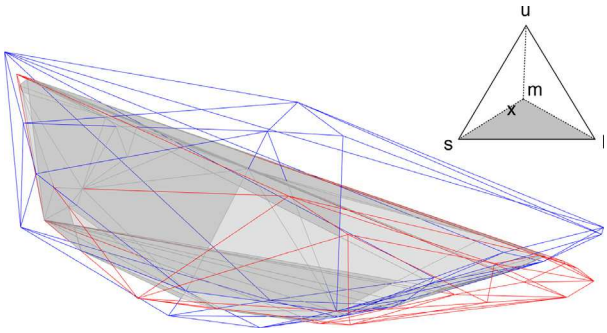

Fig. 3. Colourspace occupied by the recently diverged *Lamprotornis chloropterus* (red) and *L. elisabeth* (blue). The colour volume overlap between the two species is shown in grey. For reference, the avian tetrahedral colourspace (inset) shows the {*usml*} colour vertices, with the location of the volumes in the colourspace indicated by an 'x'.

receiver-specific visual systems (Pike *et al.* 2011) and answer questions pertaining to signal production mechanisms (e.g. comparing the relative contribution of pigments and structural colours in various integuments; D'Alba *et al.* 2012). Currently, the visual modelling functions implemented in `pavo` already allow for any number of photoreceptors to be considered, but colour distances can so far only be calculated for di-, tri- and tetrachromats. Implementing extensions of the Vorobyev & Osorio (1998) model will allow `pavo` to calculate colour distances considering the visual systems of organisms such as butterflies (e.g. six photoreceptors in Pierids and *Papilio* butterflies; Morehouse & Rutowski 2010, Arikawa 2003) and mantis shrimps (Order Stomapoda; 16 photoreceptors; Cronin *et al.* 1993).

Furthermore, as an R package, `pavo` provides a direct integration with the growing number of statistical procedures that the language incorporates, facilitating a streamlined workflow from initial data input to final analysis. It therefore allows the inclusion of colour analysis from spectral data in open and reproducible workflows, which can be directly automated and adjusted for consistency. Finally, `pavo` will continue to incorporate reference data for illuminants, backgrounds, photoreceptors, and other components of visual systems (e.g. ocular media; Siebeck & Marshall 2001) as they become available, providing a repository of spectral and colour data for physiological models across a broad range of taxa.

Citation of methods implemented in `pavo`

Most of the methods implemented in `pavo` have been thoroughly described in their original publications, to which users should refer for details and interpretation. For reflectance shape variables ('objective colourimetrics') and their particular relation to signal production and perception, see Andersson & Prager (2006) and Montgomerie (2006). Visual models based on photon catches and receptor noise are detailed in the study by Vorobyev & Osorio (1998) and Vorobyev *et al.* (1998), and photoreceptor sensitivity curve estimation in the study by Govardovskii *et al.* (2000) and Hart & Vorobyev (2005). For tetra-

hedral colourspace model implementations and variable calculations, see Endler & Mielke (2005) and Stoddard & Prum (2008), and for colour volume overlap, see Stoddard & Stevens (2011) and Stoddard (2012). Users of the functions that apply these methods should cite the original sources as appropriate, along with `pavo`.

Data accessibility

Data for the examples have been deposited in the Dryad repository: http://dx.doi.org/10.5061/dryad.298b1

Acknowledgements

We would like to thank two anonymous reviewers for comments to a previous version of this manuscript, and Jarrod D. Hadfield and Mary Caswell Stoddard for sharing code that helped us develop some of `pavo`'s capabilities. This work was supported by the NSF grant DEB-1210630, AMNH Chapman research grant and Sigma XI GIAR (R.M.), NSF grant EAR-1251895, HFSP grant RGY0083, AFOSR grant FA9550-09-1-0159 (M.D.S.), the University of Akron (M.D.S., R.M., C.M.E.), NSERC Discovery grant and Equipment grant (S.M.D), and NSERC Graduate Scholarship (P.-P.B.).

References

Andersson, S. & Prager, M. (2006) Quantifying colors. *Bird Coloration* Vol. **I** (eds K.J. Mcgraw & G.E. Hill), pp. 41–89. Harvard University Press, Cambridge, MA.

Arikawa, K. (2003) Spectral organization of the eye of a butterfly, *Papilio*. *Journal of Comparative Physiology A. Neuroethology, Sensory, Neural, and Behavioral Physiology*, **189**, 791–800.

Badyaev, A.V. & Hill, G.E. (2003) Avian sexual dichromatism in relation to phylogeny and ecology. *Annual Review of Ecology Evolution and Systematics*, **34**, 27–49.

Bennett, A.T.D. & Théry, M. (2007) Avian color vision and coloration: multidisciplinary evolutionary biology. *The American Naturalist*, **169**(S1), S1–S6.

Bolker, B. (2008) *Ecological Models and Data in R*. Princeton University Press, Princeton, NJ.

Butler, M.W., Toomey, M.B. & McGraw, K.J. (2011) How many color metrics do we need? evaluating how different color-scoring procedures explain carotenoid pigment content in avian bare-part and plumage ornaments. *Behavioral Ecology and Sociobiology*, **65**, 401–413.

Chittka, L. & Menzel, R. (1992) The evolutionary adaptation of flower colours and the insect pollinators' colour vision. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **171**, 171–181.

Craig, A., Feare, C., Croucher, B. & Shields, C. (1998) *Starlings and Mynas*. Helm Identification Guides, A&C Black, London.

Cronin, T., Marshall, N. & Caldwell, R. (1993) Photoreceptor spectral diversity in the retinas of squilloid and lysiosquilloid stomatopod crustaceans. *Journal of Comparative Physiology A. Neuroethology, Sensory, Neural, and Behavioral Physiology*, **172**, 339–350.

D'Alba, L., Kieffer, L. & Shawkey, M.D. (2012) Relative contributions of pigments and biophotonic nanostructures to natural color production: a case study in budgerigar (*Melopsittacus undulatus*) feathers. *Journal of Experimental Biology*, **215**, 1272–1277.

Darst, C.R., Cummings, M.E. & Cannatella, D.C. (2006) A mechanism for diversity in warning signals: conspicuousness versus toxicity in poison frogs. *Proceedings of the National Academy of Sciences of the United States of America*, **103**, 5852–5857.

Darwin, C. (1859) *On the Origin of the Species by Means of Natural Selection: Or, The Preservation of Favoured Races in the Struggle for Life*. John Murray, London.

Darwin, C. (1896) *The Descent of Man, and Selection in Relation to Sex*. John Murray, London.

Eaton, M.D. & Lanyon, S. (2003) The ubiquity of avian ultraviolet plumage reflectance. *Proceedings of the Royal Society B-Biological Sciences*, **270**, 1721–1726.

Endler, J.A. (1990) On the measurement and classification of colour in studies of animal colour patterns. *Biological Journal of the Linnean Society*, **41**, 315–352.

Endler, J.A. (1993) The color of light in forests and its implications. *Ecological Monographs*, **63**, 1–27.

Endler, J.A. & Mielke, P. (2005) Comparing entire colour patterns as birds see them. *Biological Journal of the Linnean Society*, **86**, 405–431.

Geyer, C.J., Meeden, G.D. & Incorporates code from cddlib written by Komei Fukuda (2012) rcdd: rcdd (Computational Geometry). URL http://CRAN.R-project.org/package = rcdd. R package version 1.1-7.

Gill, F. & Donsker, D. (2012) Ioc world bird names (version 3.2). http://www.worldbirdnames.org. [Online; accessed 12 December 2012].

Goldsmith, T.H. (1990) Optimization, constraint, and history in the evolution of eyes. *Quarterly Review of Biology*, **65**, 281–322.

Gomez, D. (2006) AVICOL, a program to analyse spectrometric data. http://sites.google.com/site/avicolprogram/. Last update January 2012.

Govardovskii, V.I., Fyhrquist, N., Reuter, T., Kuzmin, D.G. & Donner, K. (2000) In search of the visual pigment template. *Visual Neuroscience*, **17**, 509–528.

Hanley, D. (2013) Eggshell conspicuousness is related to paternal brood patch vascularisation in the american thrashers. *Avian Biology Research*, **6**, 163–177.

Hart, N.S. (2001) The visual ecology of avian photoreceptors. *Progress in Retinal and Eye Research*, **20**675–703.

Hart, N.S., Partridge, J.C. & Cuthill, I.C. (1998) Visual pigments, oil droplets and cone photoreceptor distribution in the european starling (*Sturnus vulgaris*). *Journal of Experimental Biology*, **201**, 1433–1446.

Hart, N. & Vorobyev, M. (2005) Modelling oil droplet absorption spectra and spectral sensitivities of bird cone photoreceptors. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **191**, 381–392.

Hill, G.E. (2002) *A Red Bird in a Brown Bag: The Function and Evolution of Colorful Plumage in the House Finch*. Oxford University Press, USA.

Kemp, D.J., Reznick, D.N., Grether, G.F. & Endler, J.A. (2009) Predicting the direction of ornament evolution in Trinidadian guppies (*Poecilia reticulata*). *Proceedings of the Royal Society B-Biological Sciences*, **276**, 4335–4343.

Lovette, I.J. & Rubenstein, D.R. (2007) A comprehensive molecular phylogeny of the starlings (aves: Sturnidae) and mockingbirds (aves: Mimidae): congruent mtdna and nuclear trees for a cosmopolitan avian radiation. *Molecular Phylogenetics and Evolution*, **44**, 1031–1056.

Maia, R., Eliason, C.M., Bitton, P.-P., Doucet, S.M. & Shawkey, M.D. (2013) Data from: pavo: an R package for the analysis, visualization and organization of spectral data. Dryad Digital Repository. doi: 10.5061/dryad.298b1

Montgomerie, R. (2006) Analyzing colors. *Bird Coloration* Vol. **I** (eds K.J. Mcgraw & G.E. Hill), pp. 90–147. Harvard University Press, Cambridge, MA.

Morehouse, N.I. & Rutowski, R.L. (2010) In the eyes of the beholders: female choice and avian predation risk associated with an exaggerated male butterfly color. *The American Naturalist*, **176**, 768–784.

Muller, W. & Eens, M. (2009) Elevated yolk androgen levels and the expression of multiple sexually selected male characters. *Hormones and Behavior*, **55**, 175–181.

Paradis, E., Claude, J. & Strimmer, K. (2004) Ape: analyses of phylogenetics and evolution in R language. *Bioinformatics*, **20**, 289–290.

Pike, T.W., Bjerkeng, B., Blount, J.D., Lindstroem, J. & Metcalfe, N.B. (2011) How integument colour reflects its carotenoid content: a stickleback's perspective. *Functional Ecology*, **25**, 297–304.

Poulton, E. (1890) *The Colours of Animals: Their Meaning and Use, Especially Considered in the Case of Insects*. Kegan Paul, Trench & Trübner, London.

Quesada, J. & Senar, J.C. (2006) Comparing plumage colour measurements obtained directly from live birds and from collected feathers: the case of the great tit Parus major. *Journal of Avian Biology*, **37**, 609–616.

R Development Core Team (2013) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rubenstein, D.R. & Lovette, I.J. (2009) Reproductive skew and selection on female ornamentation in social species. *Nature*, **462**, 786–789.

Shawkey, M.D., Morehouse, N.I. & Vukusic, P. (2009) A protean palette: colour materials and mixing in birds and butterflies. *Journal of the Royal Society Interface*, **6**, S221–S231.

Siebeck, U. & Marshall, N. (2001) Ocular media transmission of coral reef fish-can coral reef fish see ultraviolet light? *Vision Research*, **41**, 133–149.

Stoddard, M.C. (2012) Mimicry and masquerade from the avian visual perspective. *Current Zoology*, **58**, 630–648.

Stoddard, M.C. & Prum, R.O. (2008) Evolution of avian plumage color in a tetrahedral color space: a phylogenetic analysis of new world buntings. *The American Naturalist*, **171**, 755–776.

Stoddard, M.C. & Stevens, M. (2011) Avian vision and the evolution of egg color mimicry in the common cuckoo. *Evolution*, **65**, 2004–2013.
Tovee, M.J. (1995) Ultra-violet photoreceptors in the animal kingdom: their distribution and function. *Trends in Ecology & Evolution*, **10**, 455–460.
Villéger, S., Novack-Gottshall, P.M. & Mouillot, D. (2011) The multidimensionality of the niche reveals functional diversity changes in benthic marine biotas across geological time. *Ecology Letters*, **14**, 561–568.
Vorobyev, M. & Osorio, D. (1998) Receptor noise as a determinant of colour thresholds. *Proceedings of the Royal Society of London Series B-Biological Sciences*, **265**, 351–358.
Vorobyev, M., Osorio, D., Bennett, A.T.D., Marshall, N. & Cuthill, I. (1998) Tetrachromacy, oil droplets and bird plumage colours. *Journal of Comparative Physiology A-Neuroethology Sensory Neural and Behavioral Physiology*, **183**, 621–633.

Received 3 January 2013; accepted 13 May 2013
Handling Editor: Andrew Tatem

Supporting Information

Additional Supporting Information may be found in the online version of this article.

Appendix S1. Exact determination of volume overlap.

Figure S1. Results of Monte Carlo simulations of volume overlaps for various 3-dimensional shapes (a: cube, b: regular tetrahedron, c: cuboid) compared to the exact result.

Figure S2. Results of Monte Carlo simulations of proportional volume overlaps for arbitrary polyhedra with varying overlap volumes (a: 40%, b: 16%, c: 3%) compared to the exact calculations.

HAL
archives-ouvertes.fr

mixtools: An R Package for Analyzing Finite Mixture Models

Tatiana Benaglia, Didier Chauveau, David Hunter, Derek Young

▶ **To cite this version:**

Tatiana Benaglia, Didier Chauveau, David Hunter, Derek Young. mixtools: An R Package for Analyzing Finite Mixture Models. Journal of Statistical Software, University of California, Los Angeles, 2009, 32 (6), pp.1-29.

HAL Id: hal-00384896

https://hal.archives-ouvertes.fr/hal-00384896

Submitted on 16 May 2009

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.

mixtools: An R Package for Analyzing Finite Mixture Models

Tatiana Benaglia[1] Didier Chauveau[2]

David R. Hunter[1] Derek S. Young[1]

May 2009

[1] Department of Statistics – Pennsylvania State University

[2]MAPMO – UMR 6628 – Université d'Orléans

Abstract

The mixtools package for the R statistical software (R Development Core Team, 2008) provides a set of functions for analyzing a variety of finite mixture models. These functions include both traditional methods, such as EM algorithms for univariate and multivariate normal mixtures, and newer methods that reflect some recent research in finite mixture models. In the latter category, mixtools provides algorithms for estimating parameters in a wide range of different mixture-of-regression contexts, in multinomial mixtures such as those arising from discretizing continuous multivariate data, in nonparametric situations where the multivariate component densities are completely unspecified, and in semiparametric situations such as a univariate location mixture of symmetric but otherwise unspecified densities. Many of the algorithms of the mixtools package are EM algorithms or are based on EM-like ideas, so this article includes an overview of EM algorithms for finite mixture models.

Keywords: cutpoint, EM algorithm, mixture of regressions, model-based clustering, nonparametric mixture, semiparametric mixture, unsupervised clustering

1 Introduction to finite mixtures and mixtools

Populations of individuals may often be divided into subgroups. Yet even when we observe characteristics of these individuals that provide information about their subgroup memberships, we may not actually observe these memberships *per se*. The basic goal of the tools in the mixtools package (version 0.4.0, as of this writing) is to examine a sample of measurements to discern and describe subgroups of individuals, even when there is no observable variable that readily indexes into which subgroup an individual properly belongs. This task is sometimes referred to as "unsupervised clustering" in the literature, and in fact mixture models may be generally thought of as comprising the subset of clustering methods known as "model-based clustering".

Finite mixture models may also be used in situations beyond those for which clustering of individuals is of interest. For one thing, finite mixture models give descriptions of entire subgroups, rather than assignments of individuals to those subgroups (though the latter may be accomplished using mixture models). Indeed, even the subgroups may not necessarily be of interest; sometimes finite mixture models merely provide a means for adequately describing a particular distribution, such as the distribution of residuals in a linear regression model where outliers are present.

Whatever the goal of the modeler when employing mixture models, much of the theory of these models involves the assumption that the subgroups are distributed according to a particular parametric form — and quite often this form is univariate or multivariate normal. While mixtools does provide tools for traditional fitting of finite mixtures of univariate and multivariate normal distributions, it goes well beyond this well-studied realm. Arising from recent research whose goal is to relax or modify the assumption of multivariate normality, mixtools provides computational techniques for finite mixture model analysis in which components are regressions, multinomial vectors arising from discretization of multivariate data, or even distributions that are almost completely unspecified.

To make the mixture model framework more concrete, suppose the possibly vector-valued random variables $\mathbf{X}_1, \ldots, \mathbf{X}_n$ are a simple random sample from a finite mixture of $m > 1$ arbitrary distributions, which we will call *components* throughout this article. The density of each $\mathbf{X}_i$ may be written

$$g_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \phi_j(\mathbf{x}_i), \quad \mathbf{x}_i \in \mathbb{R}^r, \tag{1}$$

where $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \boldsymbol{\phi}) = (\lambda_1, \ldots, \lambda_m, \phi_1, \ldots, \phi_m)$ denotes the parameter and the λ_m are positive and sum to unity. We assume that the ϕ_j are drawn from some family $\mathcal{F}$ of multivariate density functions absolutely continuous with respect to, say, Lebesgue measure. The representation (1) is not identifiable if no restrictions are placed on $\mathcal{F}$, where by "identifiable" we mean that $g_{\boldsymbol{\theta}}$ has a *unique* representation of the form (1) and we do not consider that "label-switching" — i.e., reordering the m pairs $(\lambda_1, \phi_1), \ldots, (\lambda_m, \phi_m)$ — produces a distinct representation.

In the next sections we will sometimes have to distinguish between *parametric* and more general *nonparametric* situations. This distinction is related to the structure of the family $\mathcal{F}$ of distributions to which the component densities ϕ_j in model (1) belong. We say that the mixture is *parametric* if $\mathcal{F}$ is a parametric family, $\mathcal{F} = \{\phi(\cdot|\boldsymbol{\xi}), \boldsymbol{\xi} \in \mathbb{R}^d\}$, indexed by a ($d$-dimensional) Euclidean parameter $\boldsymbol{\xi}$. A parametric family often used is the univariate gaussian family $\mathcal{F} = \{\phi(\cdot|\mu, \sigma^2) = \text{density of } \mathcal{N}(\mu, \sigma^2), (\mu, \sigma^2) \in \mathbb{R} \times \mathbb{R}_*^+\}$, in which case the model parameter reduces to $\boldsymbol{\theta} = (\boldsymbol{\lambda}, (\mu_1, \sigma_1^2), \ldots, (\mu_m, \sigma_m^2))$. For the multivariate case, a possible parametric model is the *conditionally i.i.d. normal model*, for which $\mathcal{F} = \{\phi(\mathbf{x}_i) = \prod_{k=1}^{r} f(x_{ik}), f(t) \text{ density of } \mathcal{N}(\mu, \sigma^2)\}$ (this model is included in mixtools; see section 6.1). An example of a (multivariate) nonparametric situation is $\mathcal{F} = \{\phi(\mathbf{x}) = \prod_{k=1}^{r} f(x_i), f(t) \text{ a univariate density on } \mathbb{R}\}$, in which case $\boldsymbol{\theta}$ consists in a Euclidean part ($\boldsymbol{\lambda}$) and a nonparametric part ($f_1, \ldots, f_m$).

As a simple example of a dataset to which mixture models may be applied, consider the sample depicted in Figure 1. In the Old Faithful dataset, measurements give time in minutes between eruptions of the Old Faithful geyser in Yellowstone National Park, USA.

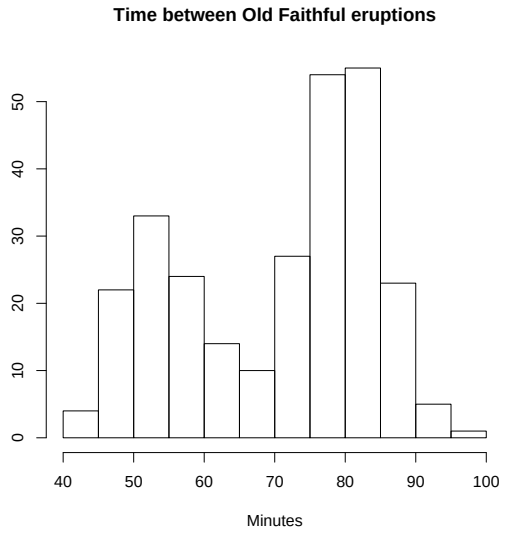

Figure 1: *The Old Faithful dataset is clearly suggestive of a two-component mixture of symmetric components.*

These data are included as part of the datasets package in R; type `help("faithful")` in R for more details. For the Old Faithful eruption data, a two-component mixture model is clearly a reasonable model based on the bimodality evident in the histogram. This example is analyzed by Hunter et al. (2007), who compare a standard normal-mixture method for fitting it with a novel semiparametric approach. Both approaches are included in mixtools; see Sections 2.3 and 4.2 of this article.

In Section 2 of the current article we review the well-known class of EM algorithms for finite mixture models, a common thread that runs throughout much of the rest of the article. The remaining sections discuss various categories of functions found in the mixtools package, from cutpoint methods that relax distributional assumptions for multivariate data by discretizing the data (Section 3), to semi- and non-parametric methods that eliminate distributional assumptions almost entirely depending on what the identifiability of the model allows (Section 4), to methods that handle various mixtures of regressions (Section 5). Finally, Section 6 describes several miscellaneous features of the mixtools package.

2 EM algorithms for finite mixtures

2.1 Missing data setup

Much of the general methodology used in mixtools involves the representation of the mixture problem as a particular case of maximum likelihood estimation (MLE) when the observations can be viewed as incomplete data. This setup implies consideration of two sample spaces, the sample space of the (incomplete) observations, and a sample space of some "complete" observations, the characterization of which being that the estimation can be performed explicitly at this level. For instance, in parametric situations, the MLE based on the complete data may exist in closed form. Among the numerous reference

papers and monographs on this subject are, e.g., the original EM algorithm paper by Dempster et al. (1977) and the finite mixture model book by McLachlan and Peel (2000) and references therein. We now give a brief description of this setup as it applies to finite mixture models in general.

The (observed) data consist of n i.i.d. observations $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ from a density $g_{\boldsymbol{\theta}}$ given by (1). It is common to denote the density of the sample by $\mathbf{g}_{\boldsymbol{\theta}}$, the n-fold product of $g_{\boldsymbol{\theta}}$, so that we write simply $\mathbf{x} \sim \mathbf{g}_{\boldsymbol{\theta}}$. In the missing data setup, $\mathbf{g}_{\boldsymbol{\theta}}$ is called the incomplete-data density, and the associated log-likelihood is $L_{\mathbf{x}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log g_{\boldsymbol{\theta}}(\mathbf{x}_i)$. The (parametric) ML estimation problem consists in finding $\hat{\boldsymbol{\theta}}_{\mathbf{x}} = \operatorname{argmax}_{\boldsymbol{\theta}\in\Phi} L_{\mathbf{x}}(\boldsymbol{\theta})$, or at least finding a local maximum — there are certain well-known cases in which a finite mixture model likelihood is unbounded (McLachlan and Peel, 2000), but we ignore these technical details for now. Calculating $\hat{\boldsymbol{\theta}}_{\mathbf{x}}$ even for a parametric finite mixture model is known to be a difficult problem, and considering $\mathbf{x}$ as incomplete data resulting from non-observed complete data helps.

The associated complete data is denoted by $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$, with density $\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{c}) = \prod_{i=1}^{n} h_{\boldsymbol{\theta}}(\mathbf{c}_i)$ (there exists a many-to-one mapping from $\mathbf{c}$ to $\mathbf{x}$, representing the loss of information). In the model for complete data associated with model (1), each random vector $\mathbf{C}_i = (\mathbf{X}_i, \mathbf{Z}_i)$, where $\mathbf{Z}_i = (Z_{ij}, j = 1, \ldots m)$, and $Z_{ij} \in \{0, 1\}$ is a Bernoulli random variable indicating that individual i comes from component j. Since each individual comes from exactly one component, this implies $\sum_{j=1}^{m} Z_{ij} = 1$, and

$$\mathbb{P}(Z_{ij} = 1) = \lambda_j, \quad (\mathbf{X}_i | Z_{ij} = 1) \sim \phi_j, \quad j = 1, \ldots, m.$$

The complete-data density for one observation is thus

$$h_{\boldsymbol{\theta}}(\mathbf{c}_i) = h_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{z}_i) = \sum_{j=1}^{m} \mathbb{I}_{z_{ij}} \lambda_j \phi_j(\mathbf{x}_i),$$

In the parametric situation, i.e. when $\mathcal{F}$ is a parametric family, it is easy to check that the complete-data MLE $\hat{\boldsymbol{\theta}}_{\mathbf{c}}$ based on maximizing $\log \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{c})$ is easy to find, provided that this is the case for the family $\mathcal{F}$.

2.2 EM algorithms

An EM algorithm iteratively maximizes, instead of the observed log-likelihood $L_{\mathbf{x}}(\boldsymbol{\theta})$, the operator

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathbb{E}\left[\log \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{C})|\mathbf{x}, \boldsymbol{\theta}^{(t)}\right],$$

where $\boldsymbol{\theta}^{(t)}$ is the current value at iteration t, and the expectation is with respect to the distribution $\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{c}|\mathbf{x})$ of $\mathbf{c}$ given $\mathbf{x}$, for the value $\boldsymbol{\theta}^{(t)}$ of the parameter. The iteration $\boldsymbol{\theta}^{(t)} \rightarrow \boldsymbol{\theta}^{(t+1)}$ is defined in the above general setup by

1. E-step: compute $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$
2. M-step: set $\boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}\in\Phi} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$

For finite mixture models, the E-step does not depend on the structure of $\mathcal{F}$, since the missing data part is only related to the $\mathbf{z}$'s:

$$\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{c}|\mathbf{x}) = \prod_{i=1}^{n} k_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i).$$

The $\mathbf{z}$ are discrete, and their distribution is given via Bayes' theorem. The M-step itself can be split in two parts, the maximization related to $\boldsymbol{\lambda}$, which does not depend on $\mathcal{F}$, and the maximization related to $\boldsymbol{\phi}$, which has to be handled specifically (say, parametrically, semi- or non-parametrically) for each model. Hence the EM algorithms for the models handled by the mixtools package share the following common features:

1. **E-step:** Calculate the "posterior" probabilities (conditional on the data and $\boldsymbol{\theta}^{(t)}$) of component inclusion,

$$p_{ij}^{(t)} \stackrel{\text{def}}{=} \mathbb{P}_{\boldsymbol{\theta}^{(t)}}(Z_{ij} = 1|\mathbf{x}_i) = \frac{\lambda_j^{(t)}\phi_j^{(t)}(\mathbf{x}_i)}{\sum_{j'=1}^{m}\lambda_{j'}^{(t)}\phi_{j'}^{(t)}(\mathbf{x}_i)} \tag{2}$$

   for all $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Numerically, it can be dangerous to implement equation (2) exactly as written due to the possibility of the indeterminant form $0/0$ in cases where $\mathbf{x}_i$ is so far from any of the components that all $\phi_{j'}^{(t)}(\mathbf{x}_i)$ values result in a numerical underflow to zero. Thus, many of the routines in mixtools actually use the equivalent expression

$$p_{ij}^{(t)} = \left[1 + \sum_{j' \neq j} \frac{\lambda_{j'}^{(t)}\phi_{j'}^{(t)}(\mathbf{x}_i)}{\lambda_j^{(t)}\phi_j^{(t)}(\mathbf{x}_i)}\right]^{-1} \tag{3}$$

   or some variant thereof.

2. **M-step for $\boldsymbol{\lambda}$:** Set

$$\lambda_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} p_{ij}^{(t)}, \quad \text{for } j = 1, \ldots, m. \tag{4}$$

2.3 An EM algorithm example

As an example, we consider the univariate normal mixture analysis of the Old Faithful waiting data depicted in Figure 1. This fully parametric situation corresponds to a mixture from the univariate gaussian family described in Section 1, where the jth component density $\phi_j(x)$ in (1) is normal with mean μ_j and variance σ_j^2. The M-step for the parameters (μ_j, σ_j^2), $j = 1, \ldots, m$ of this EM algorithm for such mixtures of univariate normals is straightforward, and can be found, e.g., in McLachlan and Peel (2000). The function `normalmixEM` implements it in mixtools. A code for the Old Faithful example, using most of the default values (e.g., stopping criterion, maximum number of iterations), is simply

```
R> data("faithful")
R> attach(faithful)
R> wait1 <- normalmixEM(waiting, lambda = .5, mu = c(55, 80), sigma = 5)

number of iterations= 9
```

The code above will fit a 2-component mixture (because `mu` is a vector of length two) in which the standard deviations are assumed equal (because `sigma` is a scalar instead of a vector). See `help("normalmixEM")` for details about specifying starting values for this EM algorithm.

```
R> plot(wait1, density = TRUE, cex.axis = 1.4, cex.lab = 1.4, cex.main = 1.8,
+    main2 = "Time between Old Faithful eruptions", xlab2 = "Minutes")
```

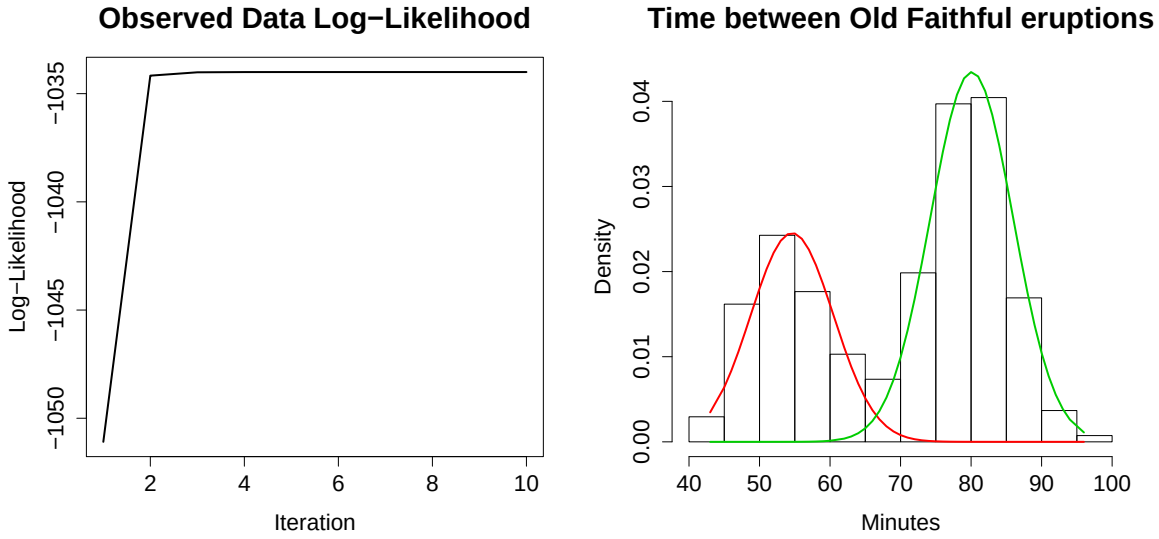

Figure 2: *The Old Faithful waiting data fitted with a parametric EM algorithm in mixtools. Left: the sequence of log-likelihood values, $L_{\mathbf{x}}(\boldsymbol{\theta}^{(t)})$; Right: the fitted gaussian components.*

The `normalmixEM` function returns an object of class `"mixEM"`, and the `plot.mixEM` function delivers the two plots given in Figure 2: the sequence $t \mapsto L_{\mathbf{x}}(\boldsymbol{\theta}^{(t)})$ of observed log-likelihood values and the histogram of the data with the m ($m = 2$ here) fitted gaussian component densities of $\mathcal{N}(\hat{\mu}_j, \hat{\sigma}_j^2)$, $j = 1, \ldots, m$, each scaled by the corresponding $\hat{\lambda}_j$, superimposed. The estimator $\hat{\boldsymbol{\theta}}$ can be displayed by typing, e.g.,

```
R> wait1[c("lambda", "mu", "sigma")]

$lambda
[1] 0.3608498 0.6391502

$mu
[1] 54.61364 80.09031

$sigma
[1] 5.869089 5.869089
```

Alternatively, the same output may be obtained using the `summary.mixEM` function:

```
R> summary(wait1)

summary of normalmixEM object:
         comp 1   comp 2
lambda  0.36085  0.63915
mu     54.61364 80.09031
sigma   5.86909  5.86909
loglik at estimate:  -1034.002
```

3 Cutpoint methods

Traditionally, most literature on finite mixture models has assumed that the density functions $\phi_j(\mathbf{x})$ of equation (1) come from a known parametric family. However, some authors have recently considered the problem in which $\phi_j(\mathbf{x})$ is unspecified except for some conditions necessary to ensure the identifiability of the parameters in the model. One such set of conditions is as follows:

Hettmansperger and Thomas (2000); Cruz-Medina et al. (2004); and Elmore et al. (2004) treat the case in which $\phi_j(\mathbf{x})$ equals the product $f_j(x_i) \cdots f_j(x_r)$ for some univariate density function f_j. Thus, conditional on knowing that $\mathbf{X}$ comes from the jth mixture component, the coordinates of $\mathbf{X}$ are independent and identically distributed. For this reason, this case is called the conditionally i.i.d. model.

The authors named above have developed an estimation method for the conditionally i.i.d. model. This method, the *cutpoint approach*, discretizes the continuous measurements by replacing each r-dimensional observation, say $\mathbf{X}_i = (x_{i1}, \ldots, x_{ir})$, by the p-dimensional multinomial vector $(n_1, \ldots, n_p)$, where $p \geq 2$ is chosen by the experimenter along with a set of cutpoints $-\infty = c_0 < c_1 < \cdots < c_p = \infty$, so that for $a = 1, \ldots, p$,

$$n_a = \sum_{k=1}^{r} I\{c_{a-1} < x_{ik} \leq c_a\}.$$

Note that the multinomial distribution is guaranteed by the conditional i.i.d. assumption, and the multinomial probability of the ath category is equal to $\theta_a \equiv P(c_{a-1} < X_{ik} \leq c_a)$.

The cutpoint approach is completely general in the sense that it can be applied to any number of components m and any number of repeated measures r, just as long as $r \geq 2m-1$, a condition that guarantees identifiability (Elmore and Wang, 2003). However, some information is lost in the discretization step, and for this reason it becomes difficult to obtain density estimates of the component densities. Furthermore, even if the assumption of conditional independence is warranted, the extra assumption of identically distributed coordinates may not be; and the cutpoint method collapses when the coordinates are not identically distributed.

As an illustration of the cutpoint approach applied to a dataset, we show here how to use mixtools to reconstruct—almost—an example from Elmore et al. (2004). The dataset is `Waterdata`, a description of which is available by typing `help("Waterdata")`. This dataset contains 8 observations on each of 405 subjects, where the observations are angle degree measurements ranging from -90 to 90 that describe the subjects' answers to a series of 8 questions related to a conceptual task about how the surface of a liquid would

be oriented if the vessel containing it were tipped to a particular angle. The correct answer is 0 degree in all cases, yet the subjects showed a remarkable variety of patterns of answers. Elmore et al. (2004) assumed the conditionally i.i.d. model (see Benaglia et al. (2009a) for an in-depth discussion of this assumption and this dataset) with both $m = 3$ and $m = 4$ mixture components. Elmore et al. (2004) summarized their results by providing plots of estimated empirical distribution functions for the component distributions, where these functions are given by

$$\tilde{F}_j(x) = \frac{1}{mn\lambda_j} \sum_{i=1}^{n} \sum_{\ell=1}^{r} p_{ij} I\{x_{i\ell} \leq x\}. \tag{5}$$

In equation (5), the values of λ_j and p_{ij} are the final maximum likelihood estimates of the mixing proportions and posterior component membership probabilities that result from fitting a mixture of m multinomials.

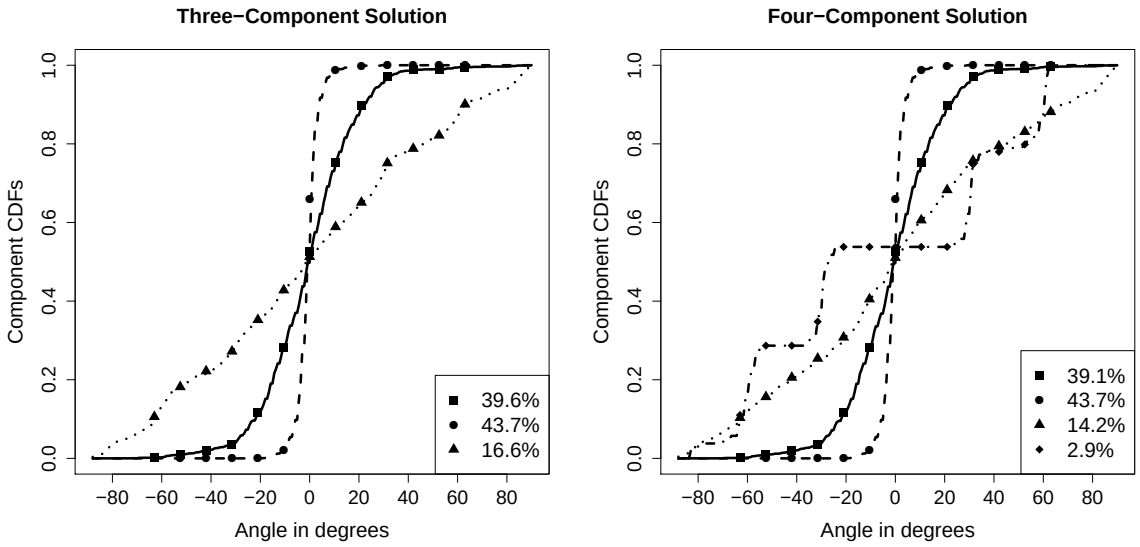

Figure 3: *Empirical cumulative distribution function (CDF) estimates for the three- and four-component multinomial cutpoint models for the water-level data; compare Figures 1 and 2 of Elmore et al. (2004). The 13 cutpoints used are indicated by the points in the plots, and the estimated mixing proportions for the various components are given by the legend.*

We cannot obtain the exact results of Elmore et al. (2004) because those authors do not state specifically which cutpoints c_a they use; they merely state that they use thirteen cutpoints. It appears from their Figures 1 and 2 that these cutpoints occur approximately at intervals of 10.5 degrees, starting at -63 and going through 63; these are the cutpoints that we adopt here. The function `makemultdata` will create a multinomial dataset from the original data, as follows:

```
R> data("Waterdata")
R> cutpts <- 10.5*(-6:6)
R> watermult <- makemultdata(Waterdata, cuts = cutpts)
```

Once the multinomial data have been created, we may apply the `multmixEM` function to estimate the multinomial parameters via an EM algorithm. Finally, `compCDF` calculates and plots the estimated distribution functions of equation (5). Figure 3 gives plots for both a 3-component and a 4-component solution; these plots are very similar to the corresponding plots in Figures 1 and 2 of Elmore et al. (2004).

```
R> set.seed(15)
R> theta4 <- matrix(runif(56), ncol = 14)
R> theta3 <- theta4[1:3,]
R> mult3 <- multmixEM(watermult, lambda=rep (1, 3) / 3, theta = theta3)

number of iterations= 378

R> cdf3 <- compCDF(Waterdata, mult3$posterior, lwd = 2, lab = c(7, 5, 7),
+    xlab = "Angle in degrees",  ylab = "Component CDFs",
+    main = "Three-Component Solution")
R> mult4 <- multmixEM (watermult, lambda = rep (1, 4) / 4, theta = theta4)

number of iterations= 197

R> cdf4 <- compCDF(Waterdata, mult4$posterior, lwd = 2, lab = c(7, 5, 7),
+    xlab = "Angle in degrees", ylab = "Component CDFs",
+    main = "Four-Component Solution")
```

As with the output of `normalmixEM` in Section 2, it is possible to summarize the output of the `multmixEM` function using the `summary.mixEM` function:

```
R> summary(mult4)

summary of multmixEM object:
            comp 1       comp 2    comp 3       comp 4
lambda  0.39129858  4.37324e-01 0.1423719  2.90055e-02
theta1  0.00158495 1.00000e-100 0.1032950  1.10049e-01
theta2  0.00702398 1.00000e-100 0.0530971  1.76659e-01
theta3  0.01046161 1.00000e-100 0.0492899 1.00000e-100
theta4  0.01496546 1.00000e-100 0.0483266  6.10181e-02
theta5  0.08306419  4.24139e-09 0.0538023  1.90177e-01
theta6  0.16368958  2.23277e-02 0.0971990  2.31815e-79
theta7  0.24531772  6.36828e-01 0.1054539 1.00000e-100
theta8  0.22535197  3.28360e-01 0.0954597 1.00000e-100
theta9  0.14693736  1.14673e-02 0.0768796 1.00000e-100
theta10 0.07284375  1.01752e-03 0.0747733  2.09779e-01
theta11 0.01809642 1.00000e-100 0.0369879  3.18723e-02
theta12 0.00114603 1.00000e-100 0.0363463  1.89517e-02
theta13 0.00533544 1.00000e-100 0.0505107  2.01493e-01
theta14 0.00418155  0.00000e+00 0.1185788  0.00000e+00
loglik at estimate:  -2881.278
```

4 Nonparametric and semiparametric methods

In this section we consider nonparametric multivariate finite mixture models. The first algorithm presented here was introduced by Benaglia et al. (2009a) as a generalization of the stochastic semiparametric EM algorithm of Bordes et al. (2007). Both algorithms are implemented in mixtools.

4.1 EM-like algorithms for mixtures of unspecified densities

Consider the mixture model described by equation (1). If we assume that the coordinates of the $\mathbf{X}_i$ vector are *conditionally independent*, i.e. they are independent conditional on the subpopulation or component (ϕ_1 through ϕ_m) from which $\mathbf{X}_i$ is drawn, the density in (1) can be rewritten as:

$$g_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \prod_{k=1}^{r} f_{jk}(x_{ik}), \tag{6}$$

where the function $f(\cdot)$, with or without subscripts, will always denote a univariate density function. Here we do not assume that $f_{jk}(\cdot)$ comes from a family of densities that may be indexed by a finite-dimensional parameter vector, and we estimate these densities using nonparametric density techniques. That is why we say that this algorithm is a fully nonparametric approach.

The density in equation (6) allows for a different distribution for each component and each coordinate of $\mathbf{X}_i$. Notice that if the density $f_{jk}(\cdot)$ does not depend on k, we have the case in which the $\mathbf{X}_i$ are not only conditionally independent but identically distributed as well. These are the two extreme cases. In order to encompass both the conditionally i.i.d. case and the more general case (6) simultaneously in one model, we allow that the coordinates of $\mathbf{X}_i$ are conditionally independent and there exist *blocks* of coordinates that are also identically distributed. If we let b_k denote the block to which the kth coordinate belongs, where $1 \leq b_k \leq B$ and B is the total number of such blocks, then equation (6) is replaced by

$$g_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \prod_{k=1}^{r} f_{jb_k}(x_{ik}). \tag{7}$$

The indices i, j, k, and ℓ will always denote a generic individual, component (subpopulation), coordinate (repeated measurement), and block, respectively. Therefore, we will always have $1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq k \leq r$, and $1 \leq \ell \leq B$.

The EM algorithm to estimate model (7) has the E-step and M-step described in Section 2.2. In equation (2), we have $\phi_j^{(t)}(\mathbf{x}_i) = \prod_{k=1}^{r} f_{jb_k}^{(t)}(x_{ik})$, where $f_{j\ell}^{(t)}(\cdot)$ is obtained by a weighted nonparametric (kernel) density estimate, given by:

3. **Nonparametric (Kernel) density estimation step:** For any real u, define for each component $j \in \{1, \ldots, m\}$ and each block $\ell \in \{1, \ldots, B\}$

$$f_{j\ell}^{t+1}(u) = \frac{1}{n h_{j\ell} C_\ell \lambda_j^{t+1}} \sum_{k=1}^{r} \sum_{i=1}^{n} p_{ij}^{(t)} I\{b_k = \ell\} K\left(\frac{u - x_{ik}}{h_{j\ell}}\right), \tag{8}$$

where $K(\cdot)$ is a kernel density function, $h_{j\ell}$ is the bandwidth for the jth component and ℓth block density estimate, and C_ℓ is the number of coordinates in the ℓth block.

The function `npEM` implements this algorithm in mixtools. This function has an argument `samebw` which, when set to `TRUE` (the default), takes $h_{j\ell} = h$, for all $1 \leq j \leq m$ and $1 \leq \ell \leq B$, that is, the same bandwidth for all components and blocks, while `samebw = FALSE` allows a different bandwidth for each component and each block, as detailed in Benaglia et al. (2009b). This function will, if called using `stochastic = TRUE`, replace the deterministic density estimation step (8) by a *stochastic* density estimation step of the type proposed by Bordes et al. (2007): First, generate $\mathbf{Z}_i^{(t)} = (Z_{i1}^{(t)}, \ldots, Z_{im}^{(t)})$ as a multivariate random vector with a single trial and success probability vector $\mathbf{p}_i^{(t)} = (p_{i1}^{(t)}, \ldots, p_{1m}^{(t)})$, then in the M-step for λ_j^{t+1} in equation (4), replace $p_{ij}^{(t)}$ by $Z_{ij}^{(t)}$ and let

$$f_{j\ell}^{t+1}(u) = \frac{1}{nh_{j\ell}C_\ell\lambda_j^{t+1}} \sum_{k=1}^{r}\sum_{i=1}^{n} Z_{ij}^{(t)} I\{b_k = \ell\} K\left(\frac{u - x_{ik}}{h_{j\ell}}\right).$$

In other words, the stochastic versions of these algorithms re-assign each observation randomly at each iteration, according to the $p_{ij}^{(t)}$ values at that iteration, to one of the m components, then the density estimate for each component is based only on those observations that have been assigned to it. Because the stochastic algorithms do not converge the way a deterministic algorithm often does, the output of `npEM` is slightly different when `stochastic = TRUE` than when `stochastic = FALSE`, the default. See the corresponding help file for details.

Benaglia et al. (2009a) also discuss specific cases of model (7) in which some of the $f_{jb_k}(\cdot)$ densities are assumed to be the same except for a location and scale change. They refer to such cases as semiparametric since estimating each $f_{jb_k}(\cdot)$ involves estimating an unknown density as well as multiple location and scale parameters. For instance, equation (17) of Benaglia et al. (2009a) sets

$$f_{j\ell}(x) = \frac{1}{\sigma_{j\ell}} f\left(\frac{x - \mu_{j\ell}}{\sigma_{j\ell}}\right), \tag{9}$$

where $\ell = b_k$ for a generic k.

The mixtools package implements an algorithm for fitting model (9) in a function called `spEM`. Details on the use of this function may be obtained by typing `help("spEM")`. Implementation of this algorithm and of that of the `npEM` function requires updating the values of $f_{jb_k}(x_{ik})$ for all i, j, and k for use in the E-step (2). To do this, the `spEM` algorithm keeps track of an $n \times m$ matrix, called Φ here, where

$$\Phi_{ij} \equiv \phi_j(\mathbf{x}_i) = \prod_{k=1}^{r} f_{jb_k}(x_{ik}).$$

The density estimation step of equation (8) updates the Φ matrix for the $(t+1)$th iteration based on the most recent values of all of the parameters. For instance, in the case of model

(9), we obtain

$$\begin{aligned}\Phi_{ij}^{t+1} &= \prod_{\ell=1}^{B}\prod_{k:b_k=\ell}\frac{1}{\sigma_{j\ell}^{t+1}}f^{t+1}\left(\frac{x-\mu_{j\ell}^{t+1}}{\sigma_{j\ell}^{t+1}}\right)\\ &= \prod_{\ell=1}^{B}\prod_{k:b_k=\ell}\frac{1}{\sigma_{j\ell}^{t+1}}\sum_{i'=1}^{n}\frac{p_{ij}^{t+1}}{hrn\lambda_j^{t+1}}\sum_{k'=1}^{r}K\left[\frac{\left(\frac{x_{ik}-\mu_{j\ell}^{t+1}}{\sigma_{j\ell}^{t+1}}\right)-(x_{i'k'}-\mu_{j\ell}^{t+1})}{h\sigma_{j\ell}^{t+1}}\right].\end{aligned}$$

4.2 A univariate symmetric, location-shifted semiparametric example

Both Hunter et al. (2007) and Bordes et al. (2006) study a particular case of model (1) in which x is univariate and

$$g_{\boldsymbol{\theta}}(x) = \sum_{j=1}^{m}\lambda_j\phi(x-\mu_j), \tag{10}$$

where $\phi(\cdot)$ is a density that is assumed to be completely unspecified except that it is symmetric about zero. Because each component distribution has both a nonparametric part $\phi(\cdot)$ and a parametric part μ_j, we refer to this model as semiparametric.

Under the additional assumption that $\phi(\cdot)$ is absolutely continuous with respect to Lebesgue measure, Bordes et al. (2007) propose a stochastic algorithm for estimating the model parameters, namely, $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \phi)$. This algorithm is implemented by the mixtools function `spEMsymloc`. This function also implements a nonstochastic version of the algorithm, which is the default and which is a special case of the general algorithm described in Section 4.1.

As noted in Figure 1, model (10) appears to be an appropriate model for the Old Faithful waiting times dataset. Here, we provide code that applies the `spEMsymloc` function to these data. First, we display the normal mixture solution of Figure 2 with a semiparametric solution superimposed, in Figure 4(a):

```
R> plot(wait1, which = 2, cex.axis = 1.4, cex.lab = 1.4, cex.main = 1.8,
+    main2 = "Time between Old Faithful eruptions", xlab2 = "Minutes")
R> wait2 <- spEMsymloc(waiting, mu0 = c(55, 80))
R> plot(wait2, lty = 2, newplot = FALSE, addlegend = FALSE)
```

Because the semiparametric version relies on a kernel density estimation step (8), it is necessary to select a bandwidth for this step. By default, `spEMsymloc` uses a fairly simplistic approach: It applies "Silverman's rule of thumb" (Silverman, 1986) to the entire dataset using the `bw.nrd0` function in R. For the Old Faithful waiting time dataset, this bandwidth is about 4:

```
R> bw.nrd0(waiting)

[1] 3.987559
```

But the choice of bandwidth can make a big difference, as seen in Figure 4(b).

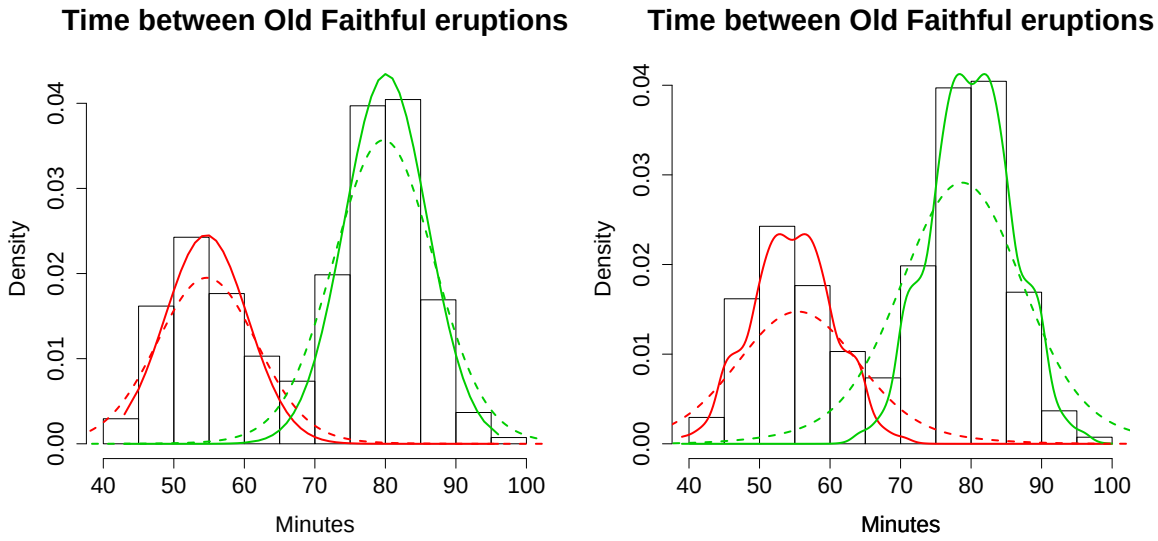

Figure 4: *The Old Faithful dataset, fit using different algorithms in* mixtools*. Left: the fitted gaussian components (solid) and a semiparametric fit assuming model (10) with the default bandwidth of* 4.0 *(dashed); Right: the same model (10) using bandwidths of* 1.0 *(solid) and* 6.0 *(dashed).*

```
R> wait2a <- spEMsymloc(waiting, mu0 = c(55, 80), bw = 1)
R> wait2b <- spEMsymloc(waiting, mu0 = c(55, 80), bw = 6)
R> plot(wait2a, lty = 1, addlegend = FALSE, cex.axis = 1.4,
+    cex.lab = 1.4, cex.main = 1.8, xlab = "Minutes",
+    title = "Time between Old Faithful eruptions")
R> plot(wait2b, lty = 2, newplot = FALSE, addlegend = FALSE)
```

We find that with a bandwidth near 2, the semiparametric solution looks quite close to the normal mixture solution of Figure 2. Reducing the bandwidth further results in the "bumpiness" exhibited by the solid line in Figure 4(b). On the other hand, with a bandwidth of 8, the semiparametric solution completely breaks down in the sense that algorithm tries to make each component look similar to the whole mixture distribution. We encourage the reader to experiment by changing the bandwidth in the above code.

4.3 A trivariate gaussian example

As a first simple, nonparametric example, we simulate a gaussian trivariate mixture with independent repeated measures and a shift of location between the two components in each coordinate, i.e., $m = 2$, $r = 3$, and $b_k = k$, $k = 1, 2, 3$. The individual densities f_{jk} are the densities of $\mathcal{N}(\mu_{jk}, 1)$, with component means $\boldsymbol{\mu}_1 = (0, 0, 0)$ and $\boldsymbol{\mu}_2 = (3, 4, 5)$. This example was introduced by Hall et al. (2005) then later reused by Benaglia et al. (2009a) for comparison purposes. Note that the parameters in this model are identifiable, since Hall and Zhou (2003) showed that for two components ($m = 2$), identifiability holds in model (1) is under mild assumptions as long as $r \geq 3$, even in the most general case in

which $b_k = k$ for all k.

A function `ise.npEM` has been included in mixtools for numerically computing the Integrated Squared Error (ISE) relative to a user-specified true density for a selected estimated density $\hat{f}_{jk}$ from `npEM` output. Each density $\hat{f}_{jk}$ is computed using equation (8) together with the posterior probabilities after convergence of the algorithm, i.e., the final values of the p_{ij}^t (when `stochastic = FALSE`). We illustrate the usage of `ise.npEM` in this example by running a Monte Carlo simulation for S replications, then computing the square root of the Mean Integrated Squared Error (MISE) for each density, where

$$\text{MISE} = \frac{1}{S}\sum_{s=1}^{S}\int \left(\hat{f}_{jk}^{(s)}(u) - f_{jk}(u)\right)^2 du, \quad j = 1, 2 \text{ and } k = 1, 2, 3.$$

For this example, we first set up the model true parameters with $S = 100$ replications of $n = 300$ observations each:

```
R> m = 2; r = 3; n = 300; S = 100
R> lambda <- c(0.4, 0.6)
R> mu <- matrix(c(0, 0, 0, 3, 4, 5), m, r, byrow = TRUE)
R> sigma <- matrix(rep(1, 6), m, r, byrow = TRUE)
```

Next, we set up "arbitrary" initial centers, a matrix for storing sums of Integrated Squared Errors, and an integer storing the number of suspected instances of label switching that may occur during the replications:

```
R> centers <- matrix(c(0, 0, 0, 4, 4, 4), 2, 3, byrow = TRUE)
R> ISE <- matrix(0, m, r, dimnames = list(Components = 1:m, Blocks = 1:r))
R> nblabsw <- 0
```

Finally, we run the Monte-Carlo simulation, using the `samebw = FALSE` option since it is more appropriate for this location-shift model:

```
R> for (mc in 1:S) {
+    x <- rmvnormmix(n, lambda, mu, sigma)
+    a <- npEM(x, centers, verb = FALSE, samebw = FALSE)
+    if (a$lambda[1] > a$lambda[2]) nblabsw <- nblabsw + 1
+    for (j in 1:m) {
+       for (k in 1:r) {
+       ISE[j, k] <- ISE[j, k] + ise.npEM(a, j, k, dnorm,
+           lower = mu[j, k] - 5, upper = mu[j, k] + 5, plots = FALSE,
+           mean = mu[j, k], sd = sigma[j, k])$value
+      }
+    }
+  }
R> MISE <- ISE/S
R> print(sqMISE <- sqrt(MISE))

          Blocks
Components         1          2          3
         1 0.06835326 0.07072128 0.06909451
         2 0.06345899 0.06256829 0.06229651
```

We can examine the `npEM` output from the last replication above using

```
R> summary(a)

300 observations, 3 coordinates, 2 components, and 3 blocks.

Means (and std. deviations) for each component:
  Block #1:  Coordinate 1
    0.0379 (0.982)   2.99 (0.973)
  Block #2:  Coordinate 2
    0.116 (1.1)   3.96 (0.894)
  Block #3:  Coordinate 3
    -0.236 (0.988)   4.99 (0.947)
```

We can also get plots of the estimated component densities for each block (recall that in this example, block ℓ consists only of coordinate ℓ) using

```
R> plot(a)
```

The resulting plots are given in Figure 5.

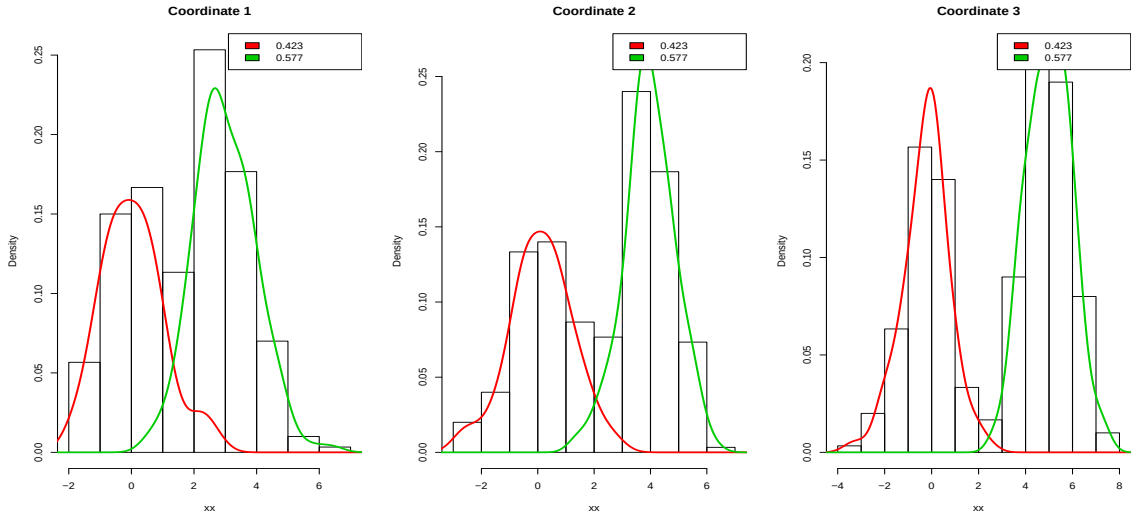

Figure 5: *An output of the* `npEM` *algorithm for the trivariate gaussian model with independent repeated measures.*

4.4 A more general multivariate nonparametric example

In this section, we fit a more difficult example, with non-multimodal mixture densities (in block #2), heavy-tailed distributions, and different scales among the coordinates. The model is multivariate with $r = 5$ repeated measures and $m = 2$ components (hence identifiability holds; cf. Hall and Zhou (2003) as cited in section 4.3). The 5 repeated measures are grouped into $B = 2$ blocks, with $b_1 = b_2 = b_3 = 1$ and $b_4 = b_5 = 2$. Block 1

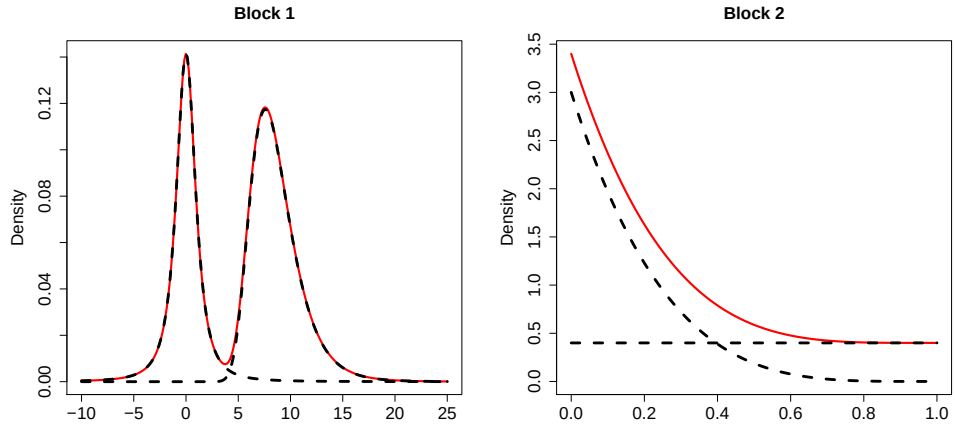

Figure 6: *True densities for the mixture of Section 4.4, with individual component densities (scaled by λ_j) in dotted lines and mixture densities in solid lines. The noncentral t mixture of coordinates 1 through 3 is on the left, the beta mixture of coordinates 4 and 5 on the right.*

corresponds to a mixture of two noncentral Student t distributions, $t'(2,0)$ and $t'(10,8)$, where the first parameter is the number of degrees of freedom, and the second is the non-centrality. Block 2 corresponds to a mixture of Beta distributions, $\mathcal{B}(1,1)$ (which is actually the uniform distribution over $[0,1]$) and $\mathcal{B}(1,5)$. The first component weight is $\lambda_1 = 0.4$. The true mixtures are depicted in Figure 6.

To fit this model in mixtools, we first set up the model parameters:

```
R> m <- 2; r <- 5
R> lambda <- c(0.4, 0.6)
R> df <- c(2, 10); ncp <- c(0, 8)
R> sh1 <- c(1, 1) ; sh2 <- c(1, 5)
```

Then we generate a pseudo-random sample of size $n = 300$ from this model:

```
R> n <- 300; z <- sample(m, n, rep = TRUE, prob = lambda)
R> r1 <- 3; z2 <- rep(z, r1)
R> x1 <- matrix(rt(n * r1, df[z2], ncp[z2]), n, r1)
R> r2 <- 2; z2 <- rep(z, r2)
R> x2 <- matrix(rbeta(n * r2, sh1[z2], sh2[z2]), n, r2)
R> x <- cbind(x1, x2)
```

For this example in which the coordinate densities are on different scales, it is obvious that the bandwidth in `npEM` should depend on the blocks and components. We set up the block structure and some initial centers, then run the algorithm with the option `samebw = FALSE`:

```
R> id <- c(rep(1, r1), rep(2, r2))
R> centers <- matrix(c(0, 0, 0, 1/2, 1/2, 4, 4, 4, 1/2, 1/2), m, r,
```

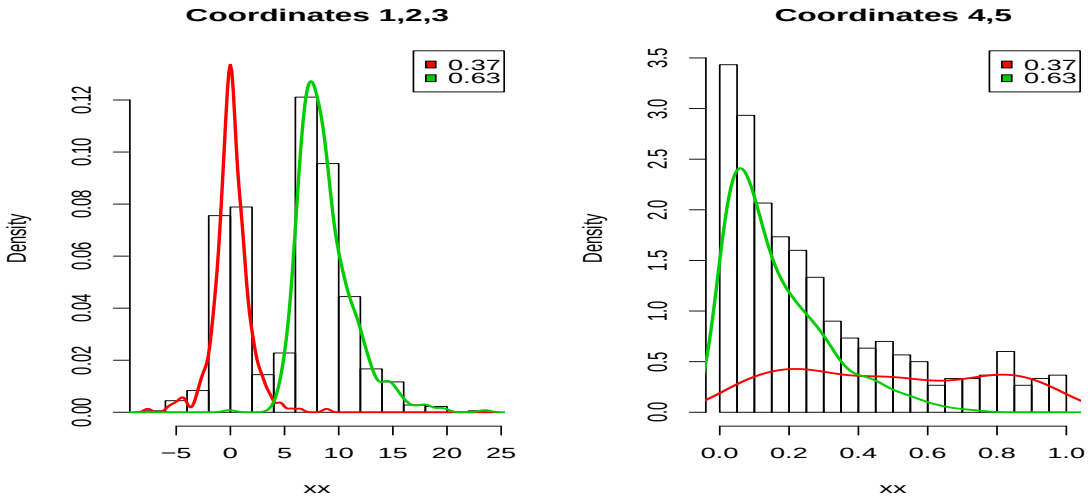

Figure 7: *Result of plotting* `npEM` *output for the example of Section 4.4. Since* $n = 300$*, the histogram on the left includes 900 observations and the one on the right includes 600.*

```
+    byrow = TRUE)
R> b <- npEM(x, centers, id, eps = 1e-8, verb = FALSE, samebw = FALSE)
```

Figure 7 shows the resulting density estimates, which may be obtained using the plotting function included in mixtools:

```
R> plot(b, breaks = 15)
```

Finally, we can compute the ISE of the estimated density relative to the truth for each block and component. The corresponding output is depicted in Figure 8.

```
R> par(mfrow = c(2, 2))
R> for (j in 1:2){
+    ise.npEM(b, j, 1, truepdf = dt, lower = ncp[j] - 10,
+      upper = ncp[j] + 10, df = df[j], ncp = ncp[j])
+    ise.npEM(b, j, 2, truepdf = dbeta, lower = -0.5,
+      upper = 1.5,  shape1 = sh1[j], shape2 = sh2[j])
+  }
```

5 Mixtures of Regressions

5.1 Mixtures of linear regressions

Consider a mixture setting where we now assume $\mathbf{X}_i$ is a vector of covariates observed with a response Y_i. The goal of mixtures of regressions is to describe the conditional distribution of $Y_i|\mathbf{X}_i$. Mixtures of regressions have been extensively studied in the econometrics literature and were first introduced by Quandt (1972) as the *switching regimes* (or *switching regressions*) problem. A switching regimes system is often compared to

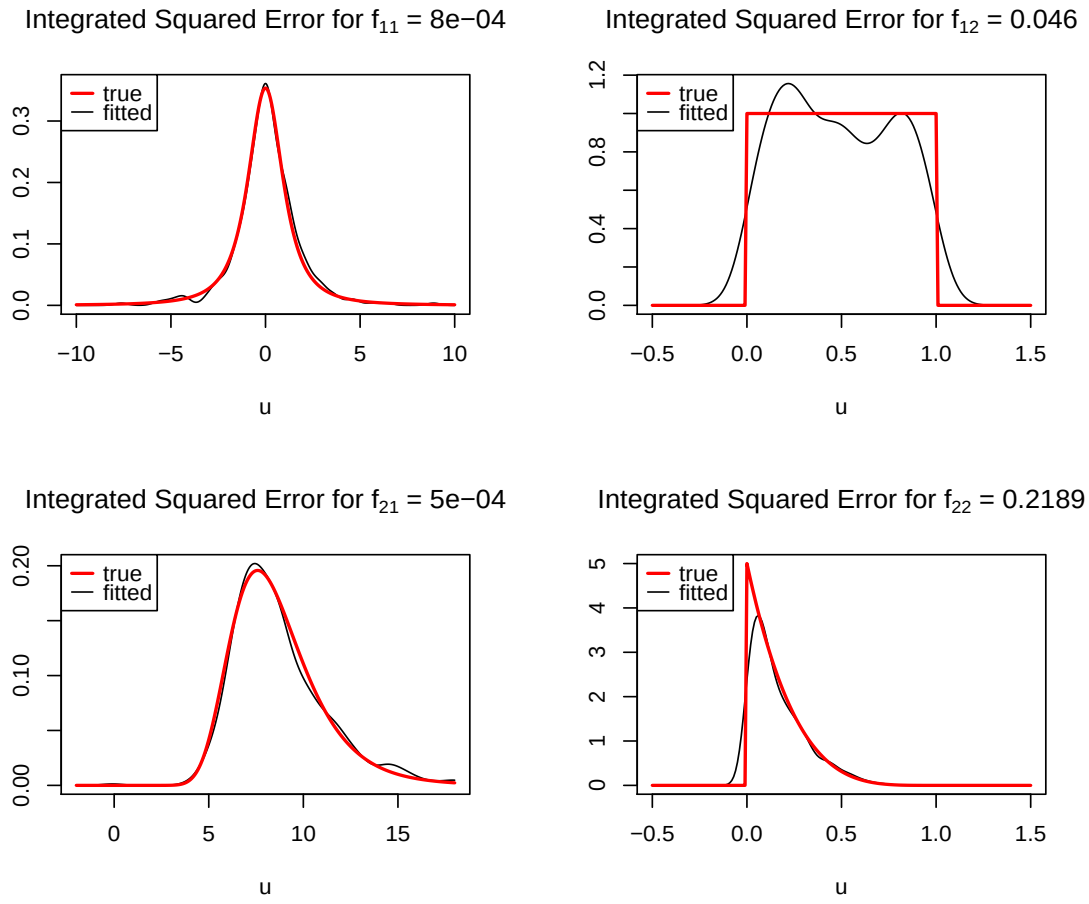

Figure 8: `ise.npEM` *output for the 5-repeated measures example; the true densities are* $f_{11} \equiv t'(2,0)$, $f_{21} \equiv t'(10,8)$, $f_{12} \equiv \mathcal{U}_{(0,1)}$, $f_{22} \equiv \mathcal{B}(1,5)$.

structural change in a system (Quandt and Ramsey, 1978). A structural change assumes the system depends deterministically on some observable variables, but switching regimes implies one is unaware of what causes the switch between regimes. In the case where it is assumed there are two heterogeneous classes, Quandt (1972) characterized the switching regimes problem "by assuming that nature chooses between regimes with probabilities λ and $1 - \lambda$".

Suppose we have n independent univariate observations, $y_1, \ldots, y_n$, each with a corresponding vector of predictors, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, with $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,p})^{\mathrm{T}}$ for $i = 1, \ldots, n$. We often set $x_{i,1} = 1$ to allow for an intercept term. Let $\mathbf{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$ and let $\underline{\mathbf{X}}$ be the $n \times p$ matrix consisting of the predictor vectors.

Suppose further that each observation $(y_i, \mathbf{x}_i)$ belongs to one of m classes. Conditional on membership in the jth component, the relationship between y_i and $\mathbf{x}_i$ is the normal regression model

$$y_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta}_j + \epsilon_i, \tag{11}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_j^2)$ and $\boldsymbol{\beta}_j$ and σ_j^2 are the p-dimensional vector of regression coefficients and the error variance for component j, respectively.

Accounting for the mixture structure, the conditional density of $y_i|\mathbf{x}_i$ is

$$g_{\boldsymbol{\theta}}(y_i|\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \phi(y_i|\mathbf{x}_i^T\boldsymbol{\beta}_j, \sigma_j^2), \tag{12}$$

where $\phi(\cdot|\mathbf{x}^T\boldsymbol{\beta}_j, \sigma_j^2)$ is the normal density with mean $\mathbf{x}^T\boldsymbol{\beta}$ and variance σ^2. Notice that the model parameter for this setting is $\boldsymbol{\theta} = (\boldsymbol{\lambda}, (\boldsymbol{\beta}_1, \sigma_1^2), \ldots, (\boldsymbol{\beta}_m, \sigma_m^2))$. The mixture of regressions model (12) differs from the well-known mixture of multivariate normals model $(Y_i, \mathbf{X}_i^{\mathrm{T}})^{\mathrm{T}} \sim \sum_{j=1}^{m} \lambda_j \mathcal{N}_{p+1}(\boldsymbol{\mu}_j, \Sigma_j)$ because model (12) makes no assertion about the marginal distribution of $\mathbf{X}_i$, whereas the mixture of multivariate normals specifies that $\mathbf{X}_i$ itself has a mixture of multivariate normals distribution.

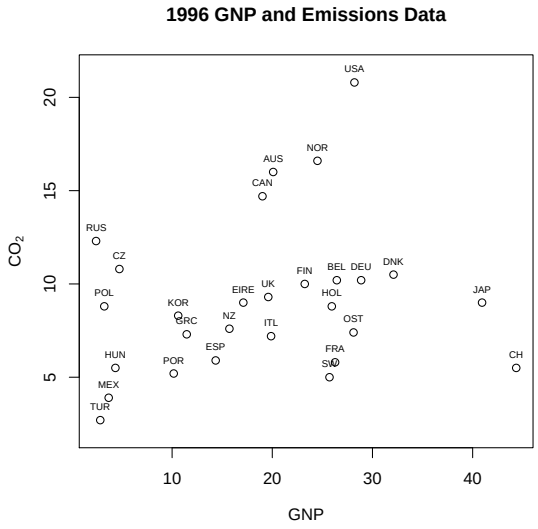

Figure 9: *1996 data on gross national product (GNP) per capita and estimated carbon dioxide (CO_2) emissions per capita. Note that "CH" stands for Switzerland, not China.*

As a simple example of a dataset to which a mixture of regressions models may be applied, consider the sample depicted in Figure 9. In this dataset, the measurements of carbon dioxide (CO_2) emissions are plotted versus the gross national product (GNP) for $n = 28$ countries. These data are included mixtools; type `help("CO2data")` in R for more details. Hurn et al. (2003) analyzed these data using a mixture of regressions from the Bayesian perspective, pointing out that "there do seem to be several groups for which a linear model would be a reasonable approximation." They further point out that identification of such groups could clarify potential development paths of lower GNP countries.

5.2 EM algorithms for mixtures of regressions

A standard EM algorithm, as described in Section 2, may be used to find a local maximum of the likelihood surface. The E-step is the same as for any finite mixture model EM algorithm; i.e., the $p_{ij}^{(t)}$ values are updated according to equation (2)—or, in reality, equation (3)—where each $\phi_j^{(t)}(\mathbf{x}_i)$ is replaced in the regression context by $\phi(y_i|\mathbf{x}_i^T\boldsymbol{\beta}_j, \sigma_j^2)$:

$$p_{ij}^{(t)} = \left[1 + \sum_{j' \neq j} \frac{\lambda_{j'}^{(t)} \phi(y_i|\mathbf{x}_i^T\boldsymbol{\beta}_{j'}, \sigma_{j'}^2)}{\lambda_j^{(t)} \phi(y_i|\mathbf{x}_i^T\boldsymbol{\beta}_j, \sigma_j^2)}\right]^{-1} \tag{13}$$

The update to the λ parameters in the M-step, equation (4), is also the same. Letting $\mathbf{W}_j^{(t)} = \text{diag}(p_{1j}^{(t)}, \ldots, p_{nj}^{(t)})$, the additional M-step updates to the $\boldsymbol{\beta}$ and σ parameters are given by

$$\boldsymbol{\beta}_j^{(t+1)} = (\underline{\mathbf{X}}^{\mathrm{T}}\mathbf{W}_j^{(t)}\underline{\mathbf{X}})^{-1}\underline{\mathbf{X}}^{\mathrm{T}}\mathbf{W}_j^{(t)}\mathbf{y} \quad \text{and} \tag{14}$$

$$\sigma_j^{2(t+1)} = \frac{\left\|\mathbf{W}_j^{1/2(t)}(\mathbf{y} - \underline{\mathbf{X}}^{\mathrm{T}}\boldsymbol{\beta}_j^{(t+1)})\right\|^2}{\text{tr}(\mathbf{W}_j^{(t)})}, \tag{15}$$

where $\|\mathbf{A}\|^2 = \mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\text{tr}(\mathbf{A})$ means the trace of the matrix $\mathbf{A}$. Notice that equation (14) is a weighted least squares (WLS) estimate of $\boldsymbol{\beta}_j$ and equation (15) resembles the variance estimate used in WLS.

Allowing each component to have its own error variance σ_j^2 results in the likelihood surface having no maximizer, since the likelihood may be driven to infinity if one component gives a regression surface passing through one or more points exactly and the variance for that component is allowed to go to zero. A similar phenomenon is well-known in the finite mixture-of-normals model where the component variances are allowed to be distinct (McLachlan and Peel, 2000). However, in practice we observe this behavior infrequently, and the mixtools functions automatically force their EM algorithms to restart at randomly chosen parameter values when it occurs. A local maximum of the likelihood function, a consistent version of which is guaranteed to exist by the asymptotic theory as long as the model is correct and all λ_j are positive, usually results without any restarts.

The function `regmixEM` implements the EM algorithm for mixtures of regressions in mixtools. This function has arguments that control options such as adding an intercept term, `addintercept = TRUE`; forcing all $\boldsymbol{\beta}_j$ estimates to be the same, `arbmean = FALSE` (for instance, to model outlying observations as having a separate error variance from the non-outliers); and forcing all σ_j^2 estimates to be the same, `arbvar = FALSE`. For additional details, type `help("regmixEM")`.

As an example, we fit a 2-component model to the GNP data shown in Figure 9. Hurn et al. (2003) and Young (2007) selected 2 components for this dataset using model selection criteria, Bayesian approaches to selecting the number of components, and a bootstrapping approach. The function `regmixEM` will be used for fitting a 2-component mixture of regressions by an EM algorithm:

```
R> data("CO2data")
```

```
R> attach(CO2data)
R> CO2reg <- regmixEM(CO2, GNP, lambda = c(1, 3) / 4,
+    beta = matrix(c(8, -1, 1, 1), 2, 2), sigma = c(2, 1))

number of iterations= 10
```

We can then pull out the final observed log-likelihood as well as estimates for the 2-component fit, which include $\hat{\lambda}$, $\hat{\boldsymbol{\beta}}_1$, $\hat{\boldsymbol{\beta}}_2$, $\hat{\sigma}_1$, and $\hat{\sigma}_2$:

```
R> summary(CO2reg)

summary of regmixEM object:
           comp 1   comp 2
lambda  0.7549214 0.245079
sigma   2.0493151 0.809389
beta1   8.6789866 1.415150
beta2  -0.0233440 0.676596
loglik at estimate:  -66.93977
```

The reader is encouraged to alter the starting values or let the internal algorithm generate random starting values. However, this fit seems appropriate and the solution is displayed in Figure 10 along with 99% Working-Hotelling Confidence Bands, which are constructed automatically by the `plot.mixEM` function in this case by assigning each point to its most probable component and then fitting two separate linear regressions:

```
plot(CO2reg, density = TRUE, alpha = 0.01)
```

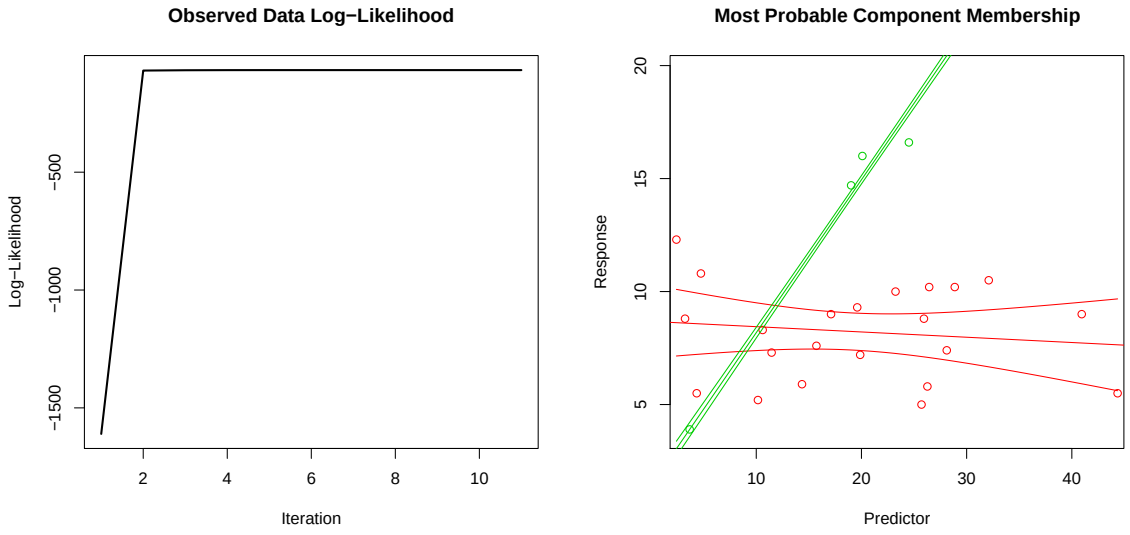

Figure 10: *The GNP data fitted with a 2-component parametric EM algorithm in* mixtools. *Left: the sequence of log-likelihood values, $L_{\mathbf{x}}(\boldsymbol{\theta}^{(t)})$; Right: the fitted regression lines with 99% Working-Hotelling Confidence Bands.*

5.3 Predictor-dependent mixing proportions

Suppose that in model (12), we replace λ_j by $\lambda_j(\mathbf{x}_i)$ and assume that the mixing proportions vary as a function of the predictors $\mathbf{x}_i$. Allowing this type of flexibility in the model might be useful for a number of reasons. For instance, sometimes it is the proportions λ_j that are of primary scientific interest, and in a regression setting it may be helpful to know whether these proportions appear to vary with the predictors. As another example, consider a `regmixEM` model using `arbmean = FALSE` in which the mixture structure only concerns the error variance: In this case, $\lambda_j(\mathbf{x})$ would give some sense of the proportion of outliers in various regions of the predictor space.

One may assume that $\lambda_j(\mathbf{x})$ has a particular parametric form, such as a logistic function, which introduces new parameters requiring estimation. This is the idea of the *hierarchical mixtures of experts* (HME) procedure (Jacobs et al., 1991), which is commonly used in neural networks. This procedure is a variant on tree-based methods — a context somewhat different from mixtures of regressions. However, a parametric form of $\lambda_j(\mathbf{x})$ may be too restrictive; in particular, the logistic function is monotone, which may not realistically capture the pattern of change of λ_j as a function of $\mathbf{x}$. As an alternative, Young and Hunter (2009) propose a nonparametric estimate of $\lambda_j(\mathbf{x}_i)$ that uses ideas from kernel density estimation.

The intuition behind the approach of Young and Hunter (2009) is as follows: The M-step estimate (4) of λ_j at each iteration of a finite mixture model EM algorithm is simply an average of the "posterior" probabilities $p_{ij} = \mathbb{E}(Z_{ij}|\text{data})$. As a substitute, The nonparametric approach uses an idea from nonparametric regression, taking a locally weighted average using a kernel function to give the weights.

Thus, considering the case of univariate x for simplicity, we take

$$\lambda_j(x) = \frac{\sum_{i=1}^{n} p_{ij} K_h(x - x_i)}{\sum_{l=1}^{n} K_h(x - x_i)}, \tag{16}$$

where $K_h(\cdot)$ is a kernel density function with scale parameter (i.e., bandwidth) h. It is straightforward to generalize equation (16) to the case of vector-valued $\mathbf{x}$ by using a multivariate kernel function.

Young and Hunter (2009) give an iterative algorithm for estimating mixture of regression parameters that replaces the standard λ_j updates (4) by the kernel-weighted version (16). The algorithm is otherwise similar to a standard EM; thus, like the algorithm in section 4.1 of this article, the resulting algorithm is an EM-like algorithm. Because only the λ_j parameters depend on $\mathbf{x}$ (and are thus "locally estimated"), whereas the other parameters (the $\boldsymbol{\beta}_j$ and σ_j) can be considered to be globally estimated, Young and Hunter (2009) call this algorithm an iterative global/local estimation (IGLE) algorithm. Naturally, it replaces the usual E-step (13) by a modified version in which each λ_j is replaced by $\lambda_j(x_i)$.

The function `regmixEM.loc` implements the IGLE algorithm in mixtools. Like the `regmixEM` function, `regmixEM.loc` has the flexibility to include an intercept term by using `addintercept = TRUE`. Moreover, this function has the argument `kern.l` to specify the kernel used in the local estimation of the $\lambda_j(\mathbf{x}_i)$. Kernels the user may specify include `"Gaussian"`, `"Beta"`, `"Triangle"`, `"Cosinus"`, and `"Optcosinus"`. Further numeric arguments relating to the chosen kernel include `kernl.g` to specify the shape parameter for

when `kern.l = "Beta"` and `kernl.h` to specify the bandwidth which controls the size of the window used in the local estimation of the mixing proportions. See the corresponding help file for additional details.

For the GNP and emissions dataset, Figure 10 indicates that the assumption of constant weights for the component regressions across all values of the covariate space may not be appropriate. The countries with higher GNP values appear to have a greater probability of belonging to the first component (i.e., the red line in Figure 10). We will therefore apply the IGLE algorithm to this dataset.

We will use the triweight kernel in equation (16), which is given by setting $\gamma = 3$ in

$$K_h(x) = \frac{1}{hB(1/2, \gamma + 1)} \left(1 - \frac{x^2}{h^2}\right)_+^{\gamma}, \tag{17}$$

where $B(x, y) = \Gamma(x)\Gamma(y)/\Gamma(x + y)$ is the beta function. For the triweight, $B(1/2, 4)$ is exactly $32/35$. This kernel may be specified in `regmixEM.loc` with `kern.l = "Beta"` and `kernl.g = 3`. The bandwidth we selected was $h = 20$, which we specify with `kernl.h = 20`.

For this implementation of the IGLE algorithm, we set the parameter estimates obtained from the mixture of regressions EM algorithm as starting values for $\hat{\boldsymbol{\beta}}_1$, $\hat{\boldsymbol{\beta}}_2$, $\hat{\sigma}_1$, and $\hat{\sigma}_2$, and set the starting values for $\lambda(x_i)$ to be 0.5 for all x_i.

```
R> CO2igle <- regmixEM.loc(CO2, GNP, beta = CO2reg$beta, sigma = CO2reg$sigma,
+    lambda = matrix(.5, 28, 2), kern.l = "Beta", kernl.h = 20, kernl.g = 3)
```

We can view the estimates for $\hat{\boldsymbol{\beta}}_1$, $\hat{\boldsymbol{\beta}}_2$, $\hat{\sigma}_1$, and $\hat{\sigma}_2$. Notice that the estimates are comparable to those obtained for the mixture of regressions EM output and the log-likelihood value is slightly higher.

```
R> summary(CO2igle)

summary of regmixEM.loc object:
          comp 1   comp 2
sigma  2.0277246 0.816510
beta1  8.8138838 1.473989
beta2 -0.0281142 0.673964
loglik at estimate:  -66.14069
```

Next, we can plot the estimates of $\lambda(x_i)$ from the IGLE algorithm.

```
R> plot(GNP, CO2igle$post[,1], xlab = "GNP",
+    ylab = "Final posterior probabilities")
R> lines(sort(GNP), CO2igle$lambda[order(GNP), 1], col=2)
R> abline(h = CO2igle$lambda[1], lty = 2)
```

This plot is given in Figure 11. Notice the curvature provided by the estimates from the IGLE fit. These fits indicate an upward trend in the posteriors. The predictor-dependent mixing proportions model provides a viable way to reveal this trend since the regular mixture of regressions fit simply provides the same estimate of λ for all x_i.

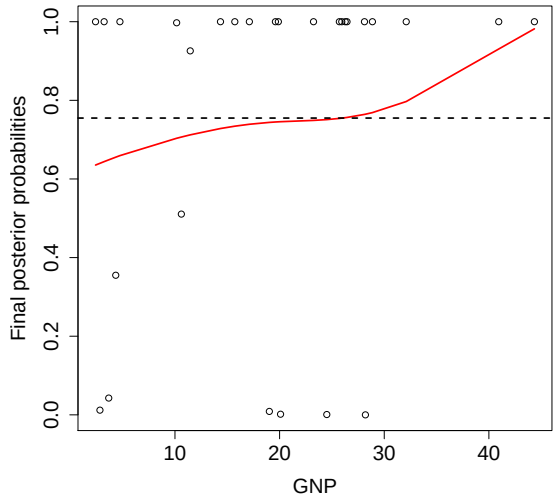

Figure 11: *Posterior membership probabilities p_{i1} for component one versus the predictor GNP along with estimates of $\lambda_1(x)$ from the IGLE algorithm (the solid red curve) and λ_1 from the mixture of linear regressions EM algorithm (the dashed black line).*

5.4 Parametric bootstrapping for standard errors

With likelihood methods for estimation in mixture models, it is possible to obtain standard error estimates by using the inverse of the observed information matrix when implementing a Newton-type method. However, this may be computationally burdensome. An alternative way to report standard errors in the likelihood setting is by implementing a parametric bootstrap. Efron and Tibshirani (1993) claim that the parametric bootstrap should provide similar standard error estimates to the traditional method involving the information matrix. In a mixture-of-regressions context, a parametric bootstrap scheme may be outlined as follows:

1. Use `regmixEM` to find a local maximizer $\hat{\boldsymbol{\theta}}$ of the likelihood.
2. For each $\mathbf{x}_i$, simulate a response value y_i^* from the mixture density $g_{\hat{\boldsymbol{\theta}}}(\cdot|\mathbf{x}_i)$.
3. Find a parameter estimate $\tilde{\boldsymbol{\theta}}$ for the bootstrap sample using `regmixEM`.
4. Use some type of check to determine whether label-switching appears to have occurred, and if so, correct it.
5. Repeat steps 2 through 4 B times to simulate the bootstrap sampling distribution of $\hat{\boldsymbol{\theta}}$.
6. Use the sample covariance matrix of the bootstrap sample as an approximation to the covariance matrix of $\hat{\boldsymbol{\theta}}$.

Note that step 3, which is not part of a standard parametric bootstrap, can be especially important in a mixture setting.

The mixtools package implements a parametric bootstrap algorithm in the `boot.se` function. We may apply it to the regression example of this section, which assumes the same estimate of λ for all x_i, as follows:

```
R> set.seed(123)
R> CO2boot <- boot.se(CO2reg, B = 100)
```

This output consists of both the standard error estimates and the parameter estimates obtained at each bootstrap replicate. An examination of the slope and intercept parameter estimates of the 500 bootstrap replicates reveals that no label-switching is likely to have occurred. For instance, the intercept terms of component one range from 4 to 11, whereas the intercept terms of component two are all tightly clumped around 0:

```
R> rbind(range(CO2boot$beta[1,]), range(CO2boot$beta[2,]))

            [,1]       [,2]
[1,]  4.4214691 11.5268409
[2,] -0.1676704  0.1007048
```

We may examine the bootstrap standard error estimates by themselves as follows:

```
R> CO2boot[c("lambda.se", "beta.se", "sigma.se")]

$lambda.se
[1] 0.0848547 0.0848547

$beta.se
            [,1]       [,2]
[1,] 1.04787448 1.20371640
[2,] 0.04464039 0.05145009

$sigma.se
[1] 0.3291421 0.2803955
```

6 Additional capabilities of mixtools

6.1 Selecting the number of components

Determining the number of components k is still a major contemporary issue in mixture modeling. Two commonly employed techniques are information criterion and parametric bootstrapping of the likelihood ratio test statistic values for testing

$$\begin{aligned} H_0 &: k = k_0 \\ H_1 &: k = k_0 + 1 \end{aligned} \tag{18}$$

for some positive integer k_0 (McLachlan, 1987).

The mixtools package has functions to employ each of these methods using EM output from various mixture models. The information criterion functions calculate An Information Criterion (AIC) of Akaike (1973), the Bayesian Information Criterion (BIC) of

Schwarz (1978), the Integrated Completed Likelihood (ICL) of Biernacki et al. (2000), and the consistent AIC (CAIC) of Bozdogan (1987). The functions for performing parametric bootstrapping of the likelihood ratio test statistics sequentially test $k = k_0$ versus $k = k_0 + 1$ for $k_0 = 1, 2, \ldots$, terminating after the bootstrapped p-value for one of these tests exceeds a specified significance level.

Currently, mixtools has functions for calculating information criteria for mixtures of multinomials (`multmixmodel.sel`), mixtures of multivariate normals under the conditionally i.i.d. assumption (`repnormmixmodel.sel`), and mixtures of regressions (`regmixmodel.sel`). Output from various mixture model fits available in mixtools can also be passed to the function `boot.comp` for the parametric bootstrapping approach. The parameter estimates from these EM fits are used to simulate data from the null distribution for the test given in (18). For example, the following application of the `multmixmodel.sel` function to the water-level multinomial data from Section 3 indicates that either 3 or 4 components seems like the best option (no more than 4 are allowed here since there are only 8 multinomial trials per observation and the mixture of multinomials requires $2m \leq r + 1$ for identifiability):

```
R> set.seed(10)
R> multmixmodel.sel(watermult, comps = 1:4, epsilon = 0.001)

number of iterations= 32
number of iterations= 393
number of iterations= 240
               1         2         3         4 Winner
AIC    -7222.967 -3109.434 -2965.748 -2936.278      4
BIC    -7248.992 -3163.487 -3047.828 -3046.385      4
CAIC   -7255.492 -3176.987 -3068.328 -3073.885      3
ICL    -7248.992 -3162.794 -3046.801 -3045.275      4
Loglik -7209.967 -3082.434 -2924.748 -2881.278      4
```

Young (2007) gives more applications of these functions to real datasets.

6.2 Bayesian methods

Currently, there are only two mixtools functions relating to Bayesian methodology and they both pertain to analyzing mixtures of regressions as described in Hurn et al. (2003). The `regmixMH` function performs a Metropolis-Hastings algorithm for fitting a mixture of regressions model where a proper prior has been assumed. The sampler output from `regmixMH` can then be passed to `regcr` in order to construct credible regions of the regression lines. Type `help("regmixMH")` and `help("regcr")` for details and an illustrative example.

Acknowledgments

This research is partially supported by NSF Award SES-0518772. D.R. Hunter received additional funding from *Le Studium*, an agency of the Centre National de la Recherche Scientifique of France.

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In Petrov, B. N. and Csaki, F., editors, *Second International Symposium on Information Theory*, pages 267–281. Akademiai Kiado.

Benaglia, T., Chauveau, D., and Hunter, D. R. (2009a). An EM-Like Algorithm for Semi- and Non-Parametric Estimation in Multivariate Mixtures. *Journal of Computational and Graphical Statistics*, to appear.

Benaglia, T., Chauveau, D., and Hunter, D. R. (2009b). Bandwidth selection in an em-like algorithm for nonparametric multivariate mixtures. Technical Report hal-00353297, version 1, HAL.

Biernacki, C., Celeux, G., and Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719–725.

Bordes, L., Chauveau, D., and Vandekerkhove, P. (2007). A Stochastic EM Algorithm for a Semiparametric Mixture Model. *Computational Statistics and Data Analysis*, 51(11):5429–5443.

Bordes, L., Mottelet, S., and Vandekerkhove, P. (2006). Semiparametric Estimation of a Two-Component Mixture Model. *Annals of Statistics*, 34(3):1204–1232.

Bozdogan, H. (1987). Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370.

Cruz-Medina, I. R., Hettmansperger, T. P., and Thomas, H. (2004). Semiparametric Mixture Models and Repeated Measures: The Multinomial Cut Point Model. *Journal of the Royal Statistical Society Series C(Applied Statistics)*, 53(3):463–474.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Chapman & Hall, London.

Elmore, R. T., Hettmansperger, T. P., and Thomas, H. (2004). Estimating Component Cumulative Distribution Functions in Finite Mixture Models. *Communications in Statistics-Theory and Methods*, 33(9):2075–2086.

Elmore, R. T. and Wang, S. (2003). Identifiability and Estimation in Finite Mixture Models with Multinomial Coefficients. Technical Report 03-04, Penn State University.

Hall, P., Neeman, A., Pakyari, R., and Elmore, R. T. (2005). Nonparametric Inference in Multivariate Mixtures. *Biometrika*, 92(3):667–678.

Hall, P. and Zhou, X. H. (2003). Nonparametric Estimation of Component Distributions in a Multivariate Mixture. *The Annals of Statistics*, 31(1):201–224.

Hettmansperger, T. P. and Thomas, H. (2000). Almost Nonparametric Inference for Repeated Measures in Mixture Models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62(4):811–825.

Hunter, D. R., Wang, S., and Hettmansperger, T. P. (2007). Inference for Mixtures of Symmetric Distributions. *Annals of Statistics*, 35:224–251.

Hurn, M., Justel, A., and Robert, C. P. (2003). Estimating mixtures of regressions. *Journal of Computational and Graphical Statistics*, 12(1):55–79.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

McLachlan, G. J. (1987). On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Applied Statistics*, 36:318–324.

McLachlan, G. J. and Peel, D. (2000). *Finite Mixture Models*. Wiley New York.

Quandt, R. E. (1972). The Estimation of the Parameters of a Linear Regression System Obeying Two Separate Regimes. *Journal of the American Statistical Association*, 67(338):306–310.

Quandt, R. E. and Ramsey, J. B. (1978). Estimating Mixtures of Normal Distributions and Switching Regressions. *Journal of the American Statistical Association*, 73(364):730–738.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC.

Young, D. S. (2007). *A Study of Mixtures of Regressions*. PhD thesis, The Pennsylvania State University. Unpublished.

Young, D. S. and Hunter, D. R. (2009). Mixtures of regressions with predictor-dependent mixing proportions. Technical Report 09-03, Penn State University.

AFFILIATION:

Didier Chauveau
Laboratoire MAPMO - UMR 6628 - Fédération Denis Poisson
Université d'Orléans
BP 6759, 45067 Orléans cedex 2, FRANCE.
E-mail: didier.chauveau@univ-orleans.fr
URL: http://www.univ-orleans.fr/mapmo/membres/chauveau/

David R. Hunter
Department of Statistics
326 Thomas Building
Pennsylvania State University
University Park, PA 16802
Telephone: +1/814-863-0979
Fax: +1/814-863-7114
E-mail: dhunter@stat.psu.edu
URL: http://www.stat.psu.edu/ dhunter/

Tatiana Benaglia
Department of Statistics, Penn State (see above)
E-mail: tab321@stat.psu.edu

Derek Young
Department of Statistics, Penn State (see above)
E-mail: dsy109@psu.edu