

# Mixed stock analysis in R: getting started with the `mixstock` package

Ben Bolker

April 11, 2011

```
> Rver <- paste(R.version$major, R.version$minor, sep = ".")
```

## 1 Introduction

The `mixstock` package is a set of routines written in the R language [7] for doing mixed stock analysis using data on markers gathered from source populations and from one or more mixed populations. The package was developed for analyzing mitochondrial DNA (mtDNA) markers from sea turtle populations, but should be applicable to any case with discrete sources, discrete mixed populations, and discrete markers. (However, I do refer to sources as “rookeries” and markers as “haplotypes” throughout this document, and you will see other echoes of its origins, e.g. the number of markers is internally stored as variable `H` and the number of sources is stored as `R`.) The package is intended to be self-contained, but some familiarity with R or S-PLUS will definitely be helpful. (Some familiarity with your computer’s operating system, which is probably Microsoft Windows, is also assumed.) The statistical methods implemented in the package are described in [1] and [6].

**This package is in the public domain (GNU General Public License), is ©2008 Ben Bolker and Toshinori Okuyama, and comes with NO WARRANTY. Please suggest improvements to me (Ben Bolker) at [bolker@zoo.ufl.edu](mailto:bolker@zoo.ufl.edu).**

If you are feeling impatient and confident, turn to “Quick Start” (section 6).

## 22 2 Installation

23 You can skip this section if you are reading this file via the `vignette()` com-  
24 mand in R— that means you’ve already successfully installed the package.

25 To get started, you will have to download and install the R package,  
26 a general-purpose statistics and graphics package, from CRAN (the “Com-  
27 prehensive R Archive Network”); go to <http://www.r-project.org> and  
28 navigate from there<sup>1</sup>

29 The following installation instructions assume you are using a “modern”  
30 Microsoft Windows system (tested on 2000 and XP); it is possible to use R,  
31 and the `mixmap` package, on other operating systems — please contact the  
32 authors for more information. (The package has been developed under Linux  
33 and runs under Windows; most of it should run under MacOS as well, but it  
34 is not as well supported and you will have to build the package from sources.  
35 To run hierarchical models using WinBUGS, you need to have WINE set  
36 up on Linux; I’m not sure about MacOS.) The setup file is about 17M,  
37 and R takes up about 40M of disk space. If you are running an antivirus  
38 package that is configured to check the signatures of executable files before  
39 they run, make sure you turn it off or register the new files installed by R  
40 before proceeding. You may also have some difficulty downloading packages  
41 if you have a firewall running on your computer — if you have trouble, you  
42 may want to (temporarily, at your own risk!) disable it.

43 Once you have downloaded and installed R, start the R program. The  
44 setup program should have asked whether you want to add a shortcut to  
45 the desktop or the Start menu: if you didn’t, you will have to search for  
46 a file called `Rgui.exe`, which probably lives somewhere (on Windows) like  
47 `Program Files\R\R-2.13.0\bin` depending on what version of R you are  
48 using and where you decided to install it. R will open up a window for you  
49 with a command prompt (`>`), at which you can type R commands. (Don’t  
50 panic.)

51 You can exit R by selecting `File/Exit` from the menus, or by typing  
52 `q()` at the command prompt. In general, if you want help on a particular  
53 command (e.g. `uml`) you can type a question mark followed by the command

---

<sup>1</sup>if you are in the US and using Windows, you can go directly to <http://cran.us.r-project.org/bin/windows/base/>: you will need to download a file called `R-x.y.z-win32.exe` which will install R for you, when executed; `x.y.z` stands for the current version of R (2.13.0 as of April 11, 2011). Otherwise, see <http://www.r-project.org/mirrors.html> for a list of alternative “mirror sites” closer to you and navigate through the web pages to find a version to install (if you are not using Unix and/or an expert, you will want to look for a *binary* version of R).

54 name (e.g. ?uml)

55 You will next need to install the `mixstock` package and two other aux-  
56 iliary packages, over the WWW, from within R (you will need to maintain  
57 a connection to the internet for this piece, although it is also possible to do  
58 this step off-line). Within R, at the command prompt, type the following  
59 commands:

```
> install.packages("mixstock")  
> install.packages("plotrix")  
> install.packages("coda")  
> install.packages("abind")  
> install.packages("R2WinBUGS")
```

60 In each case, answer `y` to whether you want to delete the source files;  
61 you shouldn't need them again.

62 (If you don't have a convenient internet connection, you can also down-  
63 load the .zip files corresponding to the different packages and install them by  
64 going to the **Packages** menu within R and choosing **Install from local**  
65 **zip file**.)

### 66 3 Loading the `mixstock` package and reading in 67 data

68 Start every session with the `mixstock` package by typing

```
> library(mixstock)
```

69 at the command prompt; this loads the `mixstock` and auxiliary packages.

70 The package can read plain text data files that are separated by white  
71 space (spaces and/or tabs) or commas. If your data are in Microsoft Excel,  
72 you should export them as a comma-separated (CSV) file. If they are in  
73 Word, save them as plain text. The expected data format is that each row of  
74 data represents a haplotype, each column except the last represents samples  
75 from a particular rookery, and the last column is the samples from the mixed  
76 population. Each row and column should be named; your life will be simpler  
77 if the names do not have spaces or punctuation other than periods in them  
78 (a common R convention is to replace spaces with periods, e.g. `North.FL`  
79 for "North FL"). Do not label the haplotype column; R detects the presence  
80 of column names by checking whether the first row has one fewer item than  
81 the rest of the rows in the file.

82 For example, a plain text file (with haplotype labels H1 and H2 and  
83 rookery labels R1–R3) could look like this:

```
84 R1 R2 R3 mix  
85 H1 1 2 3 4  
86 H2 3 4 5 6
```

87 Or a comma-separated file could look like this (note that the first line has  
88 only 4 elements while subsequent lines have 5).

```
89 R1,R2,R3,mix  
90 H1,1,2,3,4  
91 H2,3,4,5,6
```

92 If you have data from multiple mixed stocks, either put those data in a  
93 separate file or run them all together as columns of the same table (you will  
94 get a chance to specify how many sources and how many mixed populations  
95 there are):

```
96 R1,R2,R3,mix1,mix2  
97 H1,1,2,3,4,7  
98 H2,3,4,5,6,0
```

99 To read in your data, you first need to make sure that R knows how  
100 to find them. The easiest thing to do is to use the menu options<sup>2</sup> to move  
101 to a directory (i.e., folder) you will use for analysis, which should contain  
102 the data files you want to use and will contain R's working files. You can  
103 use the `getwd()` (get working directory) command to see where you are,  
104 and `list.files()` to list the files in the current directory. Once you have  
105 changed to the appropriate directory, you can read in your data files and  
106 assign the data to a variable. For example, if you had a file with space-  
107 separated data called `mydata.dat`, you could read it in by typing

```
> mydata = read.table("mydata.dat")
```

108 and if you have a comma-separated file called `mydata.csv` you can use

```
> mydata = read.csv("mydata.csv")
```

109 (You must specify the *extension* of the file — the letters after the dot.  
110 Sometimes your operating system will hide that information from you.)

111 If you have your own data you can read it in now and follow along, or  
112 you can use the `lahanas98raw` data set that comes with the package [5]:

---

<sup>2</sup>File/Change working directory on Windows, Misc/Change working directory or  
Apple-D on MacOS

```
> data(lahanas98raw)
> mydata = lahanas98raw
```

113 To make sure that everything came out OK, type the name of the variable  
 114 alone at the command prompt: e.g.

```
> mydata
```

115 to print out the data, or

```
> head(mydata)
```

	FL	MEXI	CR	AVES	SURI	BRAZ	ASCE	AFRI	CYPR	feed
I	11	7	0	0	0	0	0	0	0	2
II	1	0	0	0	0	0	0	0	0	0
III	12	5	40	3	0	0	0	0	0	62
IV	0	0	1	0	0	0	0	0	0	0
V	0	1	0	27	13	0	0	0	0	10
VI	0	0	0	0	1	0	0	0	0	0

116 to print out just the first few lines, as shown above.

117 Next, use the `as.mixstock.data` command to convert your data to a  
 118 form that the package can use:

```
> mydata = as.mixstock.data(mydata)
```

119 Once your data are converted in this way, you can use `plot(mydata)` to  
 120 produce a summary plot of the data (Figure 1).

121 The default plot is a barplot, with the proportions of each haplotype  
 122 sampled in each rookery represented by a separate bar; the mixed population  
 123 data are shown as the rightmost bar.<sup>3</sup>

124 Before proceeding, you will need to “condense” your data set by (1) ex-  
 125 cluding any haplotype samples that are found only in the mixed population  
 126 (such “singleton” haplotypes will break some estimation methods, and pro-  
 127 vide no useful information on turtle origins) and (2) lumping together all  
 128 haplotypes that are found only in a single rookery and the mixed population  
 129 (distinguishing among such haplotypes provides no extra information in our  
 130 analyses, and may slow down estimation). You can do this by typing

```
> mydata = markfreq.condense(mydata)
```

---

<sup>3</sup>you can change from the default colors by specifying a `colors=` argument: e.g. if you have 10 haplotypes, `colors=topo.colors(10)` or `colors=gray((0:9)/9)`. See `?gray` or `?rainbow` for more information.

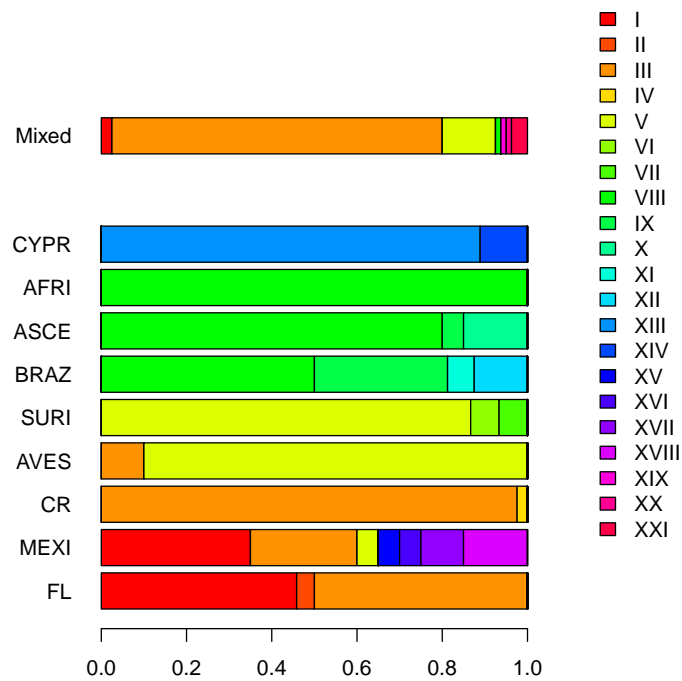


Figure 1: Basic plot of turtle mtDNA haplotype data, using `plot(mydata,mix.off=2)` (`mix.off=2` leaves a slightly larger space between the rookery and mixed stock data)

131 (To examine the condensed form of the data, you can print them by typing  
 132 `mydata` at the command prompt, `head(mydata)` to see just the first few  
 133 lines, or `plot(mydata)` to see the graphical summary [Figure 2].)  
 134 Some data are already entered in the package in the condensed format;  
 135 you can access them using the `data()` command.

```
> data(lahanas98)
```

136 makes the haplotype frequency data from Lahanas et al. 1998 [5] available  
 137 as variable `lahanas98`, while

```
> data(bolten98)
```

138 makes the loggerhead data from Bolten et al. 1998 [3] available as `bolten98`,  
 139 already converted and condensed: `bolten98raw` gives you the raw table.

## 140 4 Stock analysis

141 You can use the `mixstock` package to run various mixed-stock analyses on  
 142 your data.

### 143 4.1 Conditional and unconditional maximum likelihood

144 You can do standard conditional maximum likelihood (CML) analysis using  
 145 `cml(mydata)`. **to do: citations** If you want to save the results, you can  
 146 save them as a variable that you can then print, plot, etc. (Figure 3)

```
> mydata.cml = cml(mydata)
> mydata.cml
```

Estimated input contributions:

	FL	MEXI	CR	AVES	SURI	BRAZ
	5.463021e-02	9.453698e-05	7.833919e-01	1.485493e-01	1.333410e-06	1.333277e-06
	ASCE	AFRI	CYPR			
	1.333144e-06	1.332877e-02	1.333010e-06			

Estimated marker frequencies in sources:  
 (cml: no estimate)

method: cml

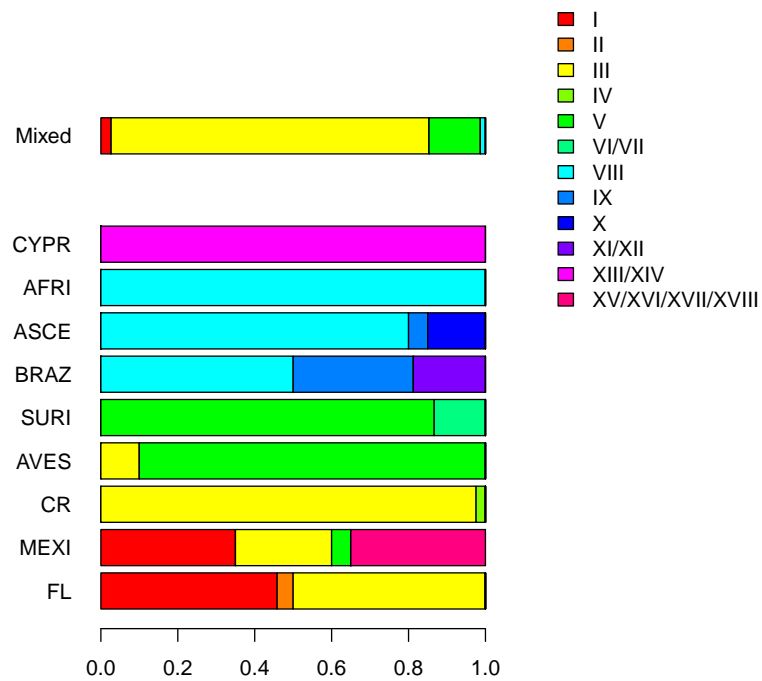


Figure 2: Condensed haplotype data from Lahanas 1998 (`plot(lahanas98, mix.off=2, leg.space=0.4)`; `leg.space=0.4` leaves more room for the legend)



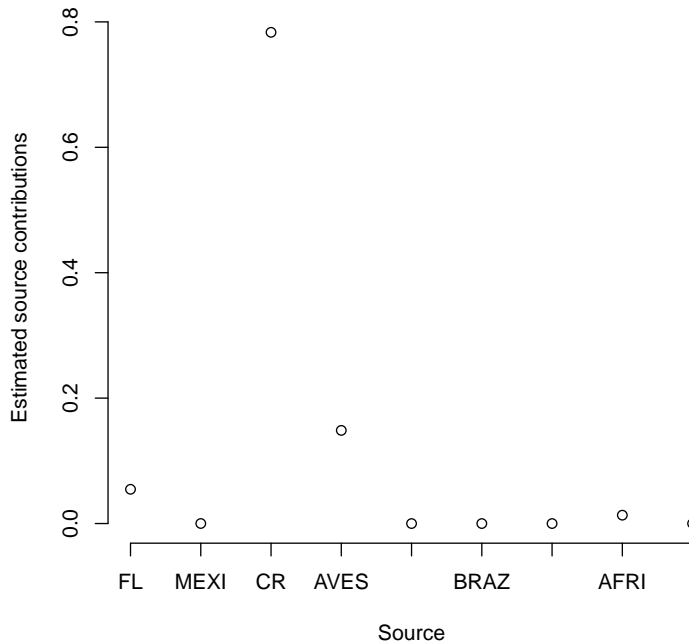


Figure 3: CML estimates for Lahanas 1998 data; `plot(mydata.cml)`

147     Assigning the results of `cml` to a variable doesn't produce any output;  
 148     you need to type the name of the variable to get the answers to print out.  
 149     Plotting the data produces a simple plot of the estimated contributions  
 150     from each source (with no error bars): see Figure 3.

```
> plot(mydata.cml)
```

151     When you print CML results, R will tell you there is no estimate for the  
 152     rookery frequencies, because CML assumes that the true rookery frequencies  
 153     are equal to the sample rookery frequencies, rather than estimating the  
 154     rookery frequencies independently.

155     The default plot for estimation results plots points specifying the esti-  
 156     mated proportions of the mixed population contributed by each rookery (to  
 157     plot this with a logarithmic scale for the vertical axis, use `plot(mydata.cml, log="y")`).

158     Standard unconditional maximum likelihood analysis (UML) takes a lit-  
 159     tle longer, but is equally straightforward [8]:

```

160 > mydata.uml = uml(mydata)
161
162 UML estimates also include estimates of the true haplotype frequencies
163 in each rookery, which are printed with the contribution estimates (as be-
164 fore, print these results by typing mydata.uml on a line by itself). As with
165 CML, you can plot the results with plot(mydata.uml); by default this plot
166 includes just the rookery contribution information. You can include the es-
167 timated haplotype frequencies in the rookeries in the graphical summary as
168 follows:

```

```

> par(ask = TRUE)
> plot(mydata.uml, plot.freqs = TRUE)
> par(ask = FALSE)

```

(par(ask=TRUE) tells R to wait for user input between successive plots).

## 168 4.2 Confidence intervals: CML and UML bootstrapping

```

> mydata.umlboot = genboot(mydata, "uml")

```

169 will generate standard (nonparametric) bootstrap confidence intervals for a  
170 UML fit to mydata, by resampling the data with replacement 1000 times  
171 (by default). *This is slow with a realistic size data set: it took 2.2 minutes*  
172 *to run 1000 bootstrap samples on my laptop.* (You can ignore warnings about  
173 singular matrix, returning equal contribs, Error in qr.solve, etc..)  
174 You can find out the results by typing

```

> confint(mydata.umlboot)

```

	2.5%	97.5%
contrib.FL	1.000000e-04	1.853967e-01
contrib.MEXI	8.255739e-05	9.999000e-05
contrib.CR	6.349666e-01	8.915403e-01
contrib.AVES	6.152913e-02	2.417467e-01
contrib.SURI	1.079622e-09	2.764224e-02
contrib.BRAZ	5.715238e-10	1.844699e-05
contrib.ASCE	1.628700e-13	3.672277e-05
contrib.AFRI	1.232938e-13	3.999982e-02
contrib.CYPR	1.719070e-13	2.407764e-05

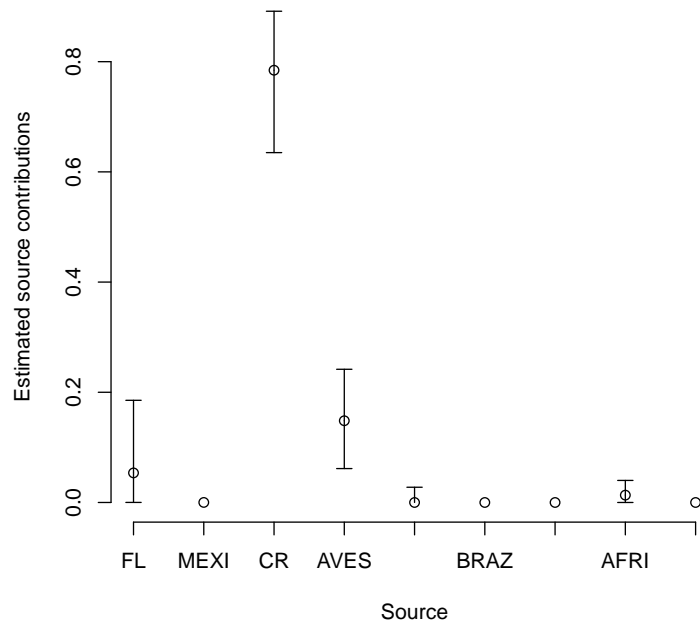


Figure 4: UML estimates with bootstrap confidence limits for Lahanas 1998  
data: `plot(mydata.umlboot)`

### 175 4.3 Markov Chain Monte Carlo estimation

```

> mydata.mcmc = tmcmc(mydata)

> mydata.mcmc

Estimated input contributions:
  contrib.FL contrib.MEXI   contrib.CR contrib.AVES contrib.SURI contrib.BRAZ
0.055518267 0.009706668 0.777704826 0.105769897 0.036445990 0.003427765
contrib.ASCE contrib.AFRI contrib.CYPR
0.004219192 0.005680010 0.001527386

Estimated marker frequencies in sources:
NULL

method: mcmc
prior strength: 0.1147742

> confint(mydata.mcmc)

                2.5%      97.5%
contrib.FL  2.009853e-11 0.23823757
contrib.MEXI 1.726347e-17 0.07512486
contrib.CR   5.956080e-01 0.89165907
contrib.AVES 3.616006e-10 0.22608667
contrib.SURI 7.363441e-16 0.17303709
contrib.BRAZ 1.664703e-16 0.02785796
contrib.ASCE 8.067783e-17 0.03001117
contrib.AFRI 3.820586e-15 0.03642586
contrib.CYPR 9.118769e-18 0.01506706

> plot(mydata.mcmc)

```

176     do the standard things: print the results, show confidence intervals, plot  
 177 the results. (By default the information on haplotype frequencies in rookeries  
 178 is not saved — it tends to be voluminous — and so this does not show up  
 179 in the MCMC results.)

### 180 4.4 Convergence diagnostics for MCMC

181 When you are running MCMC analyses, you have to check that the Markov  
 182 chains have *converged* (i.e. that you've run everything long enough for a  
 183 reliable estimate).

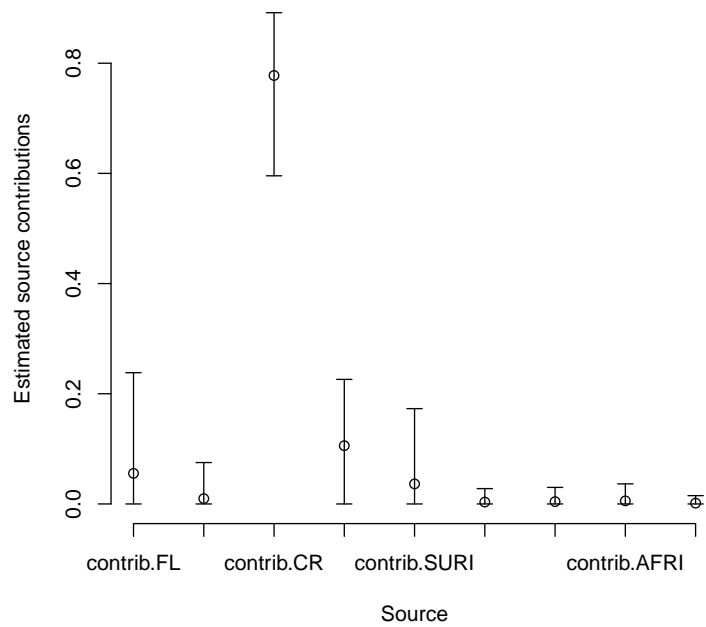


Figure 5: MCMC estimates with confidence limits for Lahanas 1998 data

#### 184 4.4.1 Raftery and Lewis

185 The command

```
> diag1 = calc.RL.0(mydata)
```

186 (The final character is the numeral 0, not the letter O).

187 runs *Raftery and Lewis* diagnostics on your data set: these criteria at-  
188 tempt to determine how long a single chain has to be in order for it to  
189 give “sufficiently good” estimates. This function actually runs an iterative  
190 procedure, repeating the chain until the R&L criterion is satisfied.

191 The results consist of two parts:

- 192 • **diag1\$current** gives the diagnostics for the last chain evaluated. These  
193 diagnostics consist of the predicted required length of the “burn-in”  
194 period (a transient that is discarded); the total number of iterations  
195 required; a lower bound on the total number required; and a “dependence  
196 factor” that tells how much correlation there is between subse-  
197 quent values in the chain (see `?raftery.diag` for more information).  
198 Here are the first few lines of **diag1\$current**:

```
> head(diag1$current)
```

	Burn-in	Total	Lower bound	Dependence factor
contrib.FL	18	1521	235	6.47
contrib.MEXI	14	926	235	3.94
contrib.CR	28	1804	235	7.68
contrib.AVES	4	312	235	1.33
contrib.SURI	15	1230	235	5.23
contrib.BRAZ	5	367	235	1.56

- 199 • **diag1\$suggested** gives the history of how long each suggested chain  
200 was as we went along: the iterations stop once suggested > current,  
201 but note that there is a lot of variability in the results.

```
> diag1$history
```

iteration	Current	Suggested
1	500	647
2	647	3882
3	3882	1804

#### 202 4.4.2 Gelman and Rubin

203 The command

```
> diag2 = calc.GR(mydata)
```

204 tests the *Gelman-Rubin* criterion, which starts multiple chains from widely  
205 spaced starting points and tests to ensure that the chains “overlap” — i.e.,  
206 that between-chain variance is small relative to within-chain variance. The  
207 general rule of thumb is that the criterion should be below 1.2 for all pa-  
208 rameters in order for the chain to be judged to have converged properly.  
209 [4].

## 210 5 Hierarchical models

211 To run hierarchical models, you will need to use either WinBUGS (on Win-  
212 dows, or on Linux or MacOS via a program called WINE, or some sort of  
213 Windows emulator) or JAGS (a newer, less well-tested program, but one that  
214 runs more easily on a variety of platforms).

215 Brief installation instructions for these programs:

- 216 • WinBUGS: go to [http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.](http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml)  
217 [shtml](http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml) and follow the instructions there to download and install WinBUGS  
218 version 1.4 and get a license key. Then make sure that you’ve installed  
219 the R2WinBUGS package (`install.packages("R2WinBUGS")`)
- 220 • JAGS: go to <http://www.fis.iarc.fr/~martyn/software/jags/> and  
221 download the appropriate version for your computer. Then install  
222 R2jags (`install.packages("R2jags", repos="http://r-forge.r-project.org")`)  
223 (*hopefully R2jags will be rolled into R2WinBUGS shortly, eliminating a*  
224 *little bit of complexity and the need to download and install a develop-*  
225 *ment version of the package*)

226 You can use the `pm.wbugs()` command (with the same syntax as `tmcmc`  
227 above) to run basic mixed stock analysis (although `tmcmc` will in general be  
228 much more convenient and efficient: `pm.wbugs` is included for completeness  
229 and testing of WinBUGS methods). Use `mm.wbugs()` to run many-to-many  
230 analyses, with R2WinBUGS (default, `pkg="WinBUGS"`) or JAGS (`pkg="JAGS"`).

### 231 5.1 Many-to-many analysis

232 The `simmixstock2` command does basic simulation of multiple-mixed-stock  
233 systems. At its simplest, it simply generates random uniform values for the

234 haplotype frequencies in each rookery and the proportional contributions of  
 235 each rookery to each mixed stock:

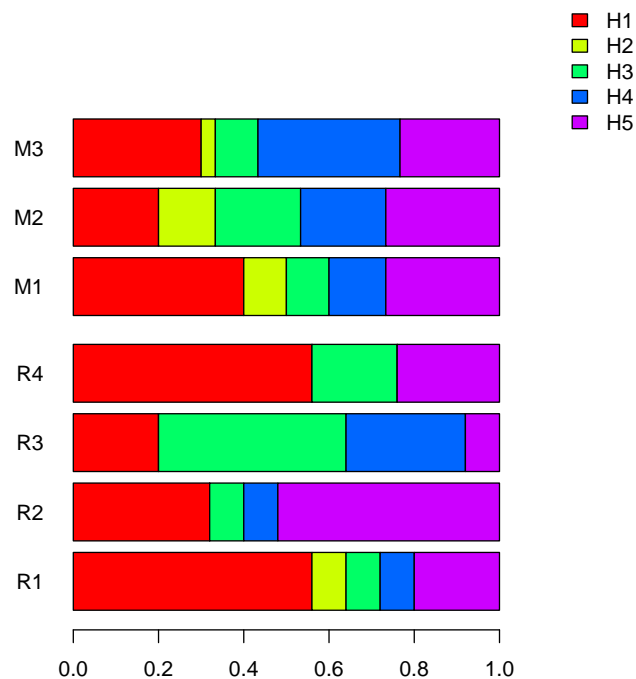
```
> Z = simmixstock2(nsource = 4, nmark = 5, nmix = 3, sourcesize = c(4,
+ 2, 1, 1), sourcesampsize = rep(25, 4), mixsampsize = rep(30,
+ 3), rseed = 1001)
> Z
```

4 sources, 3 mixed stock(s), 5 distinct markers

Sample data:

	R1	R2	R3	R4	M1	M2	M3
H1	14	8	5	14	12	6	9
H2	2	0	0	0	3	4	1
H3	2	2	11	5	3	6	3
H4	2	2	7	0	4	6	10
H5	5	13	2	6	8	8	7

```
> plot(Z)
```



236



237 Now try to fit this via `mm.wbugs`:

238 Or, keeping the run in BUGS format for diagnostic purposes:

```
> Zfit0 = mm.wbugs(Z, sourcesize = c(4, 2, 1, 1), returntype = "bugs")
```

239 This takes about 18.3 minutes to run with the default settings, which run  
240 4 chains (equal to the number of sources) for 20,000 steps each. (There are  
241 two different versions of the BUGS code that can be used with `mm.wbugs`;  
242 in this particular case they give relatively similar answers and take about  
243 the same amount of time (`bugs.code="BB"` took 9.2 minutes), but if you're  
244 having trouble you might try switching from the default `bugs.code="T0"`  
245 to `bugs.code="BB"`.

246 Other important options when running `mm.wbugs` are:

- 247 • **n.iter**: the default is 20,000 iterations per chain, with the first half  
248 used as burn-in (`n.burnin=floor(n.iter/2)`); this may be conserva-  
249 tive, and could take a long time with realistically large data sets. Use  
250 CODA's diagnostics as described above (`raftery.diag`, `gelman.diag`,  
251 etc.) to figure out an appropriate number of iterations.
- 252 • **n.chains**: equal to the number of sources by default, which may again  
253 be overkill. ([2] used three chains for an 11-source problem.)
- 254 • **inittype**: "`dispersed`" starts the chains from a starting point where  
255 95% of the contributions are assumed to come from a single source;  
256 "`random`" starts the chains from random starting points. If `which.init`  
257 is specified, these sources will be used as the dominant starting points:  
258 for example, `mm.wbugs(...,n.chains=3,inittype="dispersed",which.init=c(1,5,7))`  
259 will start 3 chains with dominant contributions from sources 1, 5, and  
260 7. If `which.init` is unspecified and `n.chains` is less than the number  
261 of sources, dominant sources will be picked at random.
- 262 • **returntype**: specifies what format to use for the answer. The de-  
263 fault is a `mixstock.est` object that can be plotted or summarized  
264 like the results from any other mixed-stock analysis. However, for  
265 diagnostic purposes, it may be worth running the code initially with  
266 `returntype="bugs"` and using `as.mcmc.bugs` and `as.mixstock.est.bugs`  
267 to convert the result to either CODA format or mixstock format. Plot-  
268 ting bugs format and CODA format gives different diagnostic plots;  
269 CODA format can also be used to run convergence diagnostics such as  
270 `raftery.diag` or `gelman.diag`.

```

271   Plots from many-to-many runs:
272   Plot BUGS format diagnostics (plot not shown):

> plot(Zfit0)

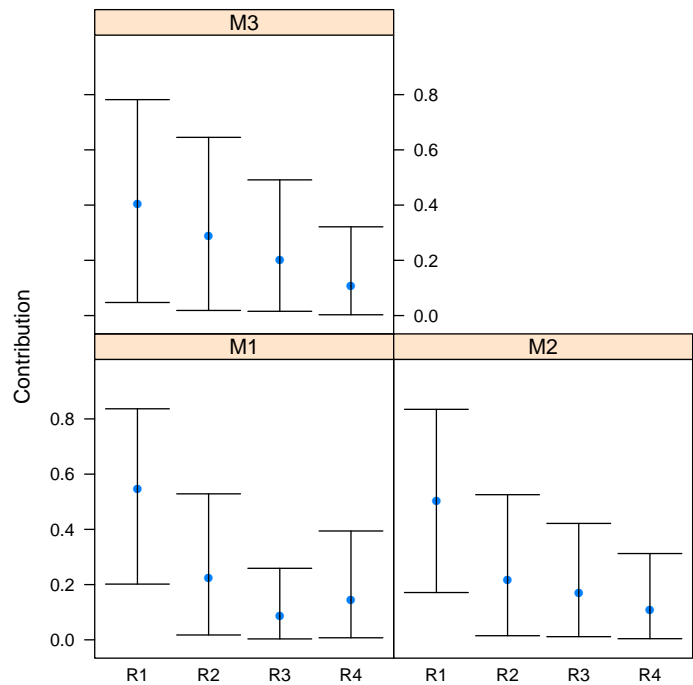
273   Plot CODA diagnostics (plot not shown):

> plot(as.mcmc.bugs(Zfit0))

274   Plot results:

> print(plot(Zfit))

```

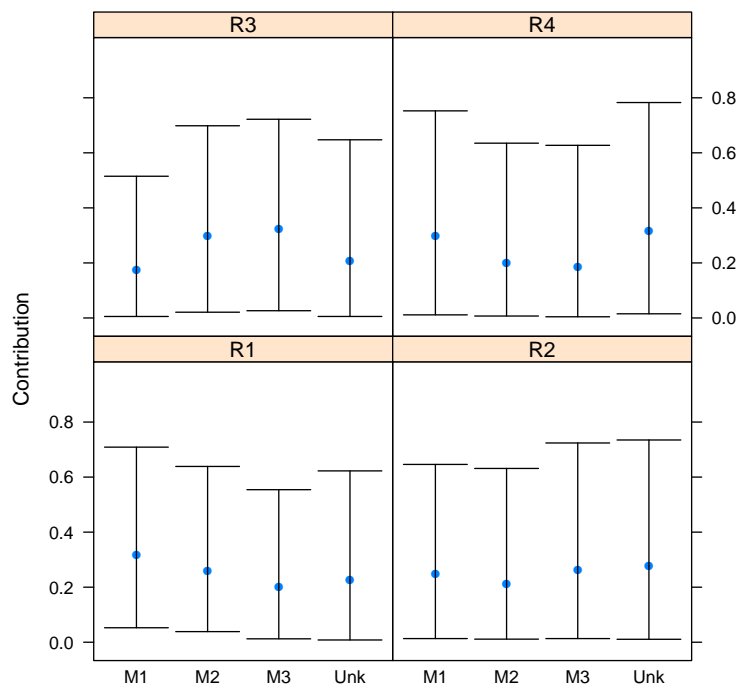


```

275   Source-centric form:
276

> print(plot(Zfit, sourcectr = TRUE))

```



277

278

Summary/confidence intervals:

```
> head(summary(Zfit))
```

4 sources, 3 mixed stock(s), 5 distinct markers

Sample data:

	R1	R2	R3	R4	M1	M2	M3
H1	14	8	5	14	12	6	9
H2	2	0	0	0	3	4	1
H3	2	2	11	5	3	6	3
H4	2	2	7	0	4	6	10
H5	5	13	2	6	8	8	7

Estimates:

Mixed-stock-centric:

		2.5%	97.5%
M1.R1	0.5473780	0.201795000	0.8366150
M1.R2	0.2235784	0.017553250	0.5286050

```

M1.R3 0.0850429 0.003377650 0.2590050
M1.R4 0.1440014 0.007369775 0.3941075
M2.R1 0.5043251 0.171260000 0.8346125
M2.R2 0.2178163 0.014860500 0.5255300
M2.R3 0.1712309 0.011442625 0.4215025
M2.R4 0.1066277 0.004133800 0.3124100
M3.R1 0.4046099 0.047320750 0.7818925
M3.R2 0.2877887 0.018549000 0.6452925
M3.R3 0.2017308 0.015441500 0.4913425
M3.R4 0.1058681 0.002893225 0.3213625

```

Source-centric:

```

                2.5%    97.5%
R1.M1 0.3171615 0.052617250 0.7088300
R1.M2 0.2584727 0.038580500 0.6387150
R1.M3 0.1997042 0.012389250 0.5542900
R1.Unk 0.2246619 0.008175600 0.6225700
R2.M1 0.2492528 0.013269500 0.6460600
R2.M2 0.2118914 0.011240250 0.6314400
R2.M3 0.2626997 0.013295500 0.7239800
R2.Unk 0.2761556 0.010689750 0.7348300
R3.M1 0.1740109 0.005432050 0.5149200
R3.M2 0.2972163 0.020928500 0.6983675
R3.M3 0.3223322 0.026362250 0.7219875
R3.Unk 0.2064394 0.005509450 0.6473575
R4.M1 0.2988757 0.011309500 0.7524525
R4.M2 0.2004035 0.007036625 0.6351050
R4.M3 0.1847740 0.004338375 0.6272475
R4.Unk 0.3159484 0.015142750 0.7827350

```

\$data

4 sources, 3 mixed stock(s), 5 distinct markers

Sample data:

```

      R1 R2 R3 R4 M1 M2 M3
H1 14  8  5 14 12  6  9
H2  2  0  0  0  3  4  1
H3  2  2 11  5  3  6  3
H4  2  2  7  0  4  6 10
H5  5 13  2  6  8  8  7

```

\$fit

\$fit\$input.freq

	R1	R2	R3	R4
M1	0.5473780	0.2235784	0.0850429	0.1440014
M2	0.5043251	0.2178163	0.1712309	0.1066277
M3	0.4046099	0.2877887	0.2017308	0.1058681

\$fit\$source.freq

NULL

\$fit\$sourcectr.freq

	M1	M2	M3	Unknown
R1	0.3171615	0.2584727	0.1997042	0.2246619
R2	0.2492528	0.2118914	0.2626997	0.2761556
R3	0.1740109	0.2972163	0.3223322	0.2064394
R4	0.2988757	0.2004035	0.1847740	0.3159484

\$resample.sum

	mean	median	sd	Q02.5	Q05	Q95	Q97.5
M1.R1	0.5473780	0.553600	0.16110594	0.201795000	0.2595000	0.799230	0.8366150
M1.R2	0.2235784	0.204450	0.13855505	0.017553250	0.0260570	0.474390	0.5286050
M1.R3	0.0850429	0.068225	0.06931447	0.003377650	0.0065684	0.233520	0.2590050
M1.R4	0.1440014	0.126050	0.10132943	0.007369775	0.0140235	0.334100	0.3941075
M2.R1	0.5043251	0.503550	0.16885282	0.171260000	0.2143150	0.782120	0.8346125
M2.R2	0.2178163	0.204700	0.13563086	0.014860500	0.0260610	0.468530	0.5255300
M2.R3	0.1712309	0.154500	0.10862593	0.011442625	0.0224255	0.379490	0.4215025
M2.R4	0.1066277	0.087870	0.08396023	0.004133800	0.0089415	0.272715	0.3124100
M3.R1	0.4046099	0.399100	0.20215962	0.047320750	0.0800140	0.738310	0.7818925
M3.R2	0.2877887	0.274750	0.17065027	0.018549000	0.0354680	0.596360	0.6452925
M3.R3	0.2017308	0.184400	0.12848814	0.015441500	0.0253800	0.435915	0.4913425
M3.R4	0.1058681	0.084805	0.08726567	0.002893225	0.0070214	0.287610	0.3213625
R1.M1	0.3171615	0.292000	0.17826667	0.052617250	0.0752155	0.651575	0.7088300
R1.M2	0.2584727	0.225500	0.16266044	0.038580500	0.0508010	0.574510	0.6387150
R1.M3	0.1997042	0.161000	0.15118056	0.012389250	0.0201575	0.504265	0.5542900
R1.Unk	0.2246619	0.185450	0.17268818	0.008175600	0.0161995	0.551420	0.6225700
R2.M1	0.2492528	0.221400	0.17715397	0.013269500	0.0206450	0.579150	0.6460600
R2.M2	0.2118914	0.175000	0.16305664	0.011240250	0.0201865	0.522395	0.6314400
R2.M3	0.2626997	0.223000	0.19132121	0.013295500	0.0223965	0.634180	0.7239800
R2.Unk	0.2761556	0.241950	0.19892308	0.010689750	0.0219895	0.644830	0.7348300
R3.M1	0.1740109	0.135750	0.14152211	0.005432050	0.0128130	0.451170	0.5149200

R3.M2	0.2972163	0.272700	0.18146115	0.020928500	0.0434125	0.629540	0.6983675
R3.M3	0.3223322	0.298150	0.19033388	0.026362250	0.0460470	0.656430	0.7219875
R3.Unk	0.2064394	0.158350	0.17602759	0.005509450	0.0108000	0.571265	0.6473575
R4.M1	0.2988757	0.256650	0.20717218	0.011309500	0.0235090	0.687640	0.7524525
R4.M2	0.2004035	0.150150	0.16932025	0.007036625	0.0121855	0.531450	0.6351050
R4.M3	0.1847740	0.134400	0.16408396	0.004338375	0.0093100	0.520820	0.6272475
R4.Unk	0.3159484	0.269400	0.21798576	0.015142750	0.0292240	0.729235	0.7827350

279 (check this!)

## 280 6 Quick start

- 281 • Download and install R from CRAN (find the site closest to you at  
282 <http://cran.r-project.org/mirrors.html>; go to “Precompiled bi-  
283 nary distributions” and from there to the base package; pick your  
284 operating system; download the setup program; and run the setup  
285 program).

- 286 • Start R.

- 287 • From within R, download and install the `mixstock` package and aux-  
288 iliary packages:

```
> bbcontrib = "http://www.zoo.ufl.edu/bolker/R/windows"
> install.packages("mixstock", contriburl = bbcontrib)
> install.packages("plotrix")
> install.packages("coda")
> install.packages("abind")
> install.packages("R2WinBUGS")
```

289 (This installation procedure needs to be done only once, although the  
290 `library` command below, loading the package, needs to be done for  
291 every new R session.)

- 292 • Load the package: `library(mixstock)`

- 293 • Load data from a comma-separated value (CSV) file, convert to proper  
294 format, and condense haplotypes:

```
> mydata = hapfreq.condense(as.mixstock.data(read.csv("myfile.dat")))
```

- 295 • analyze, e.g:

```

> mydata.mcmc = tmcmc(mydata)
> mydata.mcmc
> intervals(mydata.mcmc)
> plot(mydata.mcmc)

```

## 296 7 To do

- 297 • read.csv/read.table + as.mixstock.data combined into a single read.mixstock.data
- 298 command? (also incorporate hapfreq.condense as a default option)
- 299 • print.mixstock.est could print sample frequencies instead of saying
- 300 “no estimate” for CML
- 301 • MCMC section could be cleaned up considerably, explained better,
- 302 R&L parameters not hard-coded, more efficient — don’t re-run chains
- 303 every time
- 304 • incorporate rookery sizes in data
- 305 • keep CODA objects or potential for CODA plots in MCMC results
- 306 • make MCMC convergence process more efficient: more explanation
- 307 • add hierarchical models????
- 308 • describe fuzz and bounds parameters on CML/UML, E-M algorithm
- 309 • plot(...,legend=TRUE) doesn’t work for CML. add unstacked/beside=TRUE
- 310 option to plot.mixstock.est
- 311 • incorporate source size data as part of data object
- 312 • some functions don’t work with uncondensed data: fix or issue warning
- 313 • use HPDinterval from CODA for confidence intervals, rather than
- 314 quantiles?

## 315 References

- 316 [1] Benjamin Bolker, Toshinori Okuyama, Karen Bjorndal, and Alan Bolten.
- 317 Stock estimation for sea turtle populations using genetic markers: ac-
- 318 counting for sampling error of rare genotypes. *Ecological Applications*,
- 319 13(3):763–775, 2003.

- 320 [2] Benjamin M. Bolker, Toshinori Okuyama, Karen A. Bjorndal, and  
321 Alan B. Bolten. Incorporating multiple mixed stocks in mixed stock  
322 analysis: 'many-to-many' analyses. *Molecular Ecology*, 2007. in press.
- 323 [3] Alan B. Bolten, Karen A. Bjorndal, Helen R. Martins, Thomas Dellinger,  
324 Manuel J. Biscotio, Sandra E. Encalada, and Brian W. Bowen. Transat-  
325 lantic developmental migrations of loggerhead sea turtles demonstrated  
326 by mtDNA sequence analysis. *Ecological Applications*, 8(1):1–7, 1998.
- 327 [4] A. Gelman, J. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data*  
328 *analysis*. Chapman and Hall, New York, New York, USA, 1996.
- 329 [5] P. N. Lahanas, K. A. Bjorndal, A. B. Bolten, S. E. Encalada, M. M.  
330 Miyamoto, R. A. Valverde, and B. W. Bowen. Genetic composition of a  
331 green turtle (*Chelonia mydas*) feeding ground population: evidence for  
332 multiple origins. *Marine Biology*, 130:345–352, 1998.
- 333 [6] J. Pella and M. Masuda. Bayesian methods for analysis of stock mixtures  
334 from genetic characters. *Fisheries Bulletin*, 99:151–167, 2001.
- 335 [7] R Development Core Team. *R: A language and environment for statisti-*  
336 *cal computing*. R Foundation for Statistical Computing, Vienna, Austria,  
337 2005. ISBN 3-900051-07-0.
- 338 [8] P. E. Smouse, R. S. Waples, and J. A. Tworek. A genetic mixture analysis  
339 for use with incomplete source population data. *Canadian Journal of*  
340 *Fisheries and Aquatic Sciences*, 47:620–634, 1990.