Using MixSIAR

Andrew Parnell, School of Mathematics and Statistics,
University College Dublin

Learning outcomes

- Run MixSIAR on one of the standard examples
- Check convergence and produce plots in MixSIAR
- Be able to understand output from MixSIAR

Intro to MixSIAR

- MixSIAR implements a version of the CLR jags code already shown, and through this allows the introduction of random effects and covariates
- It's a bit limited in that:
  1. It's recommended to work with a GUI
  2. You can only put a certain number of random effects/covariates into the model
  3. It doesn't currently take account of uncertainty in TEFs (it calls this *discrimination*)
  4. It doesn't allow for much choice in prior distributions
- It's a great way of finding simple ways to run more complicated models, though beware it's very slow to run in most cases.

The main MixSIAR input screen

Running the Wolves example

- In the MixSIAR folder there are three files: `wolves_consumer.csv`, `wolves_dicrimination.csv`, and `wolves_sources.csv`
- These contain the consumer isotope data with two covariates, the discrimination data (no standard deviations), and the source data (for each different region)
- Note that MixSIAR allows you to input either the sources means/sds (as we have been using), or the raw source data, to which it fits a model to estimate the source parameters
- These data are similar to those used in the Semmens *et al* PLoS ONE paper already mentioned. The methods are more fully described in that paper
- The variables pack and region here are included as *nested* random effects, meaning that there are two layers of random effects which measure variability between regions and variability between packs. The overall residual measures variability within pack

Wolves example - consumers

Wolves example - sources

Wolves example - isospace plot

Wolves example - running

- First run the model as a test go, then choose a normal run
- These models are starting to get complicated, they take a while to run
- MixSIAR write JAGS code on the fly. Check out the JAGS model structure in `MixSIAR_model.txt`
- When it's finished click on `process output` to get at the (many) results
- You can also access an R script version of this (much better) in `mixsiar_script.r`

Analysing output

- First, check convergence using Brooks-Gelman-Rubin or Geweke (both previously covered)
- You then have access to histograms of the posterior distributions of the overall means, and also for each level of the hierarchy; here pack and region
- They also produce a pairs plot (like the matrix plot of yesterday) which can tell you about model inadequacy
- Use the `mixsiar_script.r` if you want full access to the JAGS output for any further analysis

Alternative version

- It's recently become much easier to use MixSIAR through the command line (it's faster, more repeatable and easier to install)
- There are number of excellent vignettes which show how to use MixSIAR from the command line: see `https://cran.r-project.org/web/packages/MixSIAR`
- We will follow the Wolves script available at `https://cran.r-project.org/web/packages/MixSIAR/vignettes/wolves_ex.html`

How to run MixSIAR

1. Load the data into MixSIAR
2. Create the iso-space plot
3. Write the JAGS model
4. Run the model
5. Look at diagnostics
6. Save the bits you want

Loading in the data

```
library(MixSIAR)

# Find the data
mix.filename = system.file("extdata", "wolves_consumer.csv"

# Load into MixSIAR
mix = load_mix_data(filename=mix.filename,
                    iso_names=c("d13C","d15N"),
                    factors=c("Region","Pack"),
                    fac_random=c(TRUE,TRUE),
                    fac_nested=c(FALSE,TRUE),
                    cont_effects=NULL)
```

- You can do the same with the sources (`load_source_data`) and the discrimination factors (`load_discr_data`)

Iso-space plot

```
plot_data(filename="isospace_plot", plot_save_pdf = FALSE,
          mix = mix, source = source, discr = discr)
```

Write the JAGS model

```
model_filename = "MixSIAR_model.txt"   # Name of the JAGS m
write_JAGS_model(model_filename, resid_err = TRUE, process_
                 mix, source)
```

- You can go and find this file now. It's saved in the current working directory.
- Beware: it can be quite hard to read!

Run the model

- First do a test run:

```
jags.1 = run_model(run="test", mix, source, discr, model_f
                   alpha.prior = 1, resid_err = TRUE, proce
```

- Then do a full run:

```
jags.2 = run_model(run="very short", mix, source, discr, mo
                   alpha.prior = 1, resid_err = TRUE, proce
```

Create output

```
output_JAGS(jags.2, mix = mix, source = source)
```

- Warning - this creates EVERYTHING!
- Adding an extra argument `output_options` as a list allows for finer-grained controlled

MixSIAR output - trace plot

MixSIAR output - random effect densities

MixSIAR output - overall proportion densities

MixSIAR output - matrix plot

Summary

- We now know how to load in special types of consumers, sources, and discrimination factors in MixSIAR
- We can use MixSIAR via the GUI or the command line (command line is better in the long run)
- We have run one of the more complicated MixSIAR examples