Species distribution models using an empirical Bayes approach via Template Model Builder (TMB)

Mollie Brooks
Danish Technical University
National Institute of Aquatic Resources

Talk Outline

- What are latent variables?
- What are empirical Bayes and TMB?
- Basics of using TMB
- What are species distribution models?
- How do you fit these models in TMB and/or glmmTMB?

"Latent variables" come in several flavors

- Random effects can account for correlation among repeated measures of sampling units (e.g. LMMs or GLMMs).

- Random effects can account for spatial or temporal correlation depending on how close together they are.

- Latent states of an ecological process that’s being observed with error (e.g. hidden Markov models, state-space models, hierarchical models).

- These categories may overlap and some people use "random effects" interchangeably with "latent variables".

Many people use Bayesian methods for these reasons:

- the ability to fit flexible models
- the ability to fit hierarchical models (i.e. with latent variables)
- the ability to quantitatively incorporate prior knowledge
- the ability to work with a joint likelihood (e.g. integrate different data sets)
- the ability to get MCMC samples to use for inference

TMB fulfills all of these criteria, but it's an empirical Bayes method.

Bayesian vs empirical Bayes methods

- Both have abilities listed on previous slide.
- Both are grounded in likelihood theory.
- Empirical Bayes methods are in the "classical statistics" category.
- Cressie et al. (2009) state "Parameter uncertainty can be handled either by estimating the unknown parameters (empirical-Bayesian analysis) or by expressing that uncertainty via a prior probability distribution (Bayesian analysis)."
- Empirical Bayes methods estimate the unknown parameters from the data. TMB does this by assuming that latent variables (or transformations of latent variables) are normally distributed (making TMB unable to automatically handle discrete latent variables).
- See de Valpine 2009 *Eco Apps* for a more thorough discussion.

Ecological Applications, 19(3), 2009, pp. 584–588

Shared challenges and common ground for Bayesian and classical analysis of hierarchical statistical models

Perry de Valpine[1]

- "The difference between Bayesian and classical philosophies is that Bayesian analysis uses the mathematics of probability distributions for model parameters, P as defined by Cressie et al., while classical analysis does not."

- "In a hierarchical model, one has parameters (P, either Bayesian or classical) that define the distribution of unknown ecological states (E) that define the distribution of data (D)."

- "some purported contrasts of Bayesian and classical results are confounded with contrasts of hierarchical and nonhierarchical models, respectively."

- "for many people [Bayesian analysis] is currently the easiest framework for analyzing a hierarchical model."

I use TMB for flexibility and speed

- In meetings where management decisions are made, fisheries stock assessors need to be able to try different models in hours, not days.

- I do simulation testing of fisheries management strategies where I commonly need to fit a model 1 million times.

    - simulate a population 30 years into the future

    - simulate 1000 replicate populations for uncertainty estimation

    - compare more than 10 management strategies for each population

    - check what management strategy will be less harmful to the population

Template Model Builder (TMB)

- Developed by Kasper Kristensen (DTU-Aqua)
- R-package inspired by ADMB
- Continuously developed since 2009
- Uses Laplace approximation for random effects
- Uses automatic differentiation to navigate the likelihood surface

Basics of TMB

- In c++, you define an objective function which is the negative log likelihood.
- Then, TMB uses automatic differentiation (AD) to get gradients of the likelihood surface.
- On the R side of things, you organize data, maximize the negative log likelihood, optionally get MCMC samples, and organize results.

Steps by the user on the c++ side

- Allocate data and parameter objects
- Combine data and parameters to calculate expected values
- Residuals around the expected values are assumed to have some distribution (eg. normal or Poisson)
- Use the assumed distribution to calculate the negative log-likelihood
  - e.g. `-dnorm(y, mu, sd, true)`
  - or `-dpois(y, lambda, true)`

Getting data into TMB

- Get data into R and organize it in the usual ways.
  - e. g. `read.csv(), load(), scan()`, etc.
- Put data into a named list to pass to c++ calculations.
  - e. g. `dat=list(x=x, y=y)`
  - Note that data frames are also lists.
- Names must match the names of data objects in your cpp file.
- The order doesn’t matter.

Some possible types of data

What	R side (example)	c++ side
Number	`1`	`DATA_SCALAR`
Vector	`c(1,2,3)`	`DATA_VECTOR`
Matrix	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_MATRIX`
Array	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_ARRAY`
Integer	`1`	`DATA_INTEGER`
Integer Vector	`c(1,2,3)`	`DATA_IVECTOR`
Integer Matrix	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_IMATRIX`
Integer Array	`matrix(c(1,2,3,4), nrow=2, ncol=2)`	`DATA_IARRAY`
Factor	`factor(c("a","b"))`	`DATA_FACTOR`
String	`"a"`	`DATA_STRING`

C++ indexing starts at 0

In R

```
data <- list()
data$y <- c(1.1, 2.2)
data$z <- mymatrix

y[1] ... y[n]
z[1,1] ... z[m,n]
```

In C++

```
DATA_VECTOR(y)
DATA_ARRAY(z)


y(0) ... y(n-1)
z(0,0) ... z(m-1,n-1)
```

Getting results from c++ to R

- If estimated standard errors of X are not needed, then use
  - `REPORT(X)` in the cpp file
  - `obj$report()$X` in the R file
- If estimated standard errors of X are needed, then use
  - `ADREPORT(X)` in the cpp file (for derived quantities)
  - `summary(sdreport(obj))` in the R file
- To get estimates and standard deviations in the same format as they entered the parameter list, try
  - Parameter list: `pl <- as.list(sdreport(obj),"Est")`
  - Parameter Sd list: `plsd <- as.list(sdreport(obj),"Std")`

Examining objects on C++ side using `REPORT( )`

```
library(TMB)
compile("size.cpp")
dyn.load(dynlib("size"))
data <- list()
data$V <- 1:3
data$M <- matrix(1:6, nrow=2, ncol=3)
data$A <- array(1:6, dim=c(1,2,3))

param <- list()
param$mu <- 0
obj <- MakeADFun(data, param)
obj$report()

#Rout>  $Ad
#Rout>  [1] 1 2 3
#Rout>
#Rout>  $Vs
#Rout>  [1] 3
#Rout>
#Rout>  $Ms
#Rout>  [1] 6
#Rout>
#Rout>  $Mnrow
#Rout>  [1] 2
#Rout>
#Rout>  $As
#Rout>  [1] 6
#Rout>
#Rout>  $Mncol
#Rout>  [1] 3
#Rout>
```

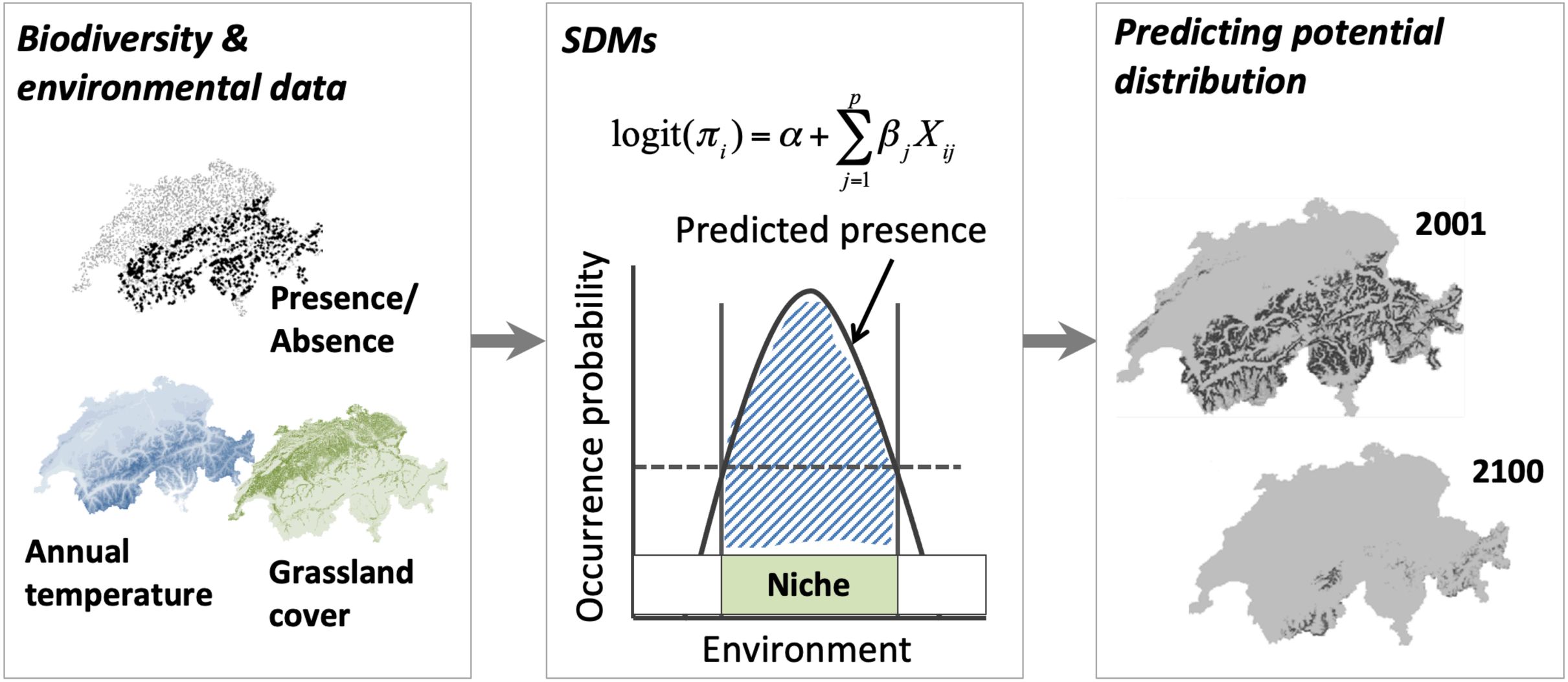
```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator()()
{
  DATA_VECTOR(V);
  DATA_MATRIX(M);
  DATA_ARRAY(A);

  int Vs=V.size();
  REPORT(Vs);

  int Ms=M.size();
  REPORT(Ms);
  int Mnrow=M.rows();
  REPORT(Mnrow);
  int Mncol=M.cols();
  REPORT(Mncol);

  int As=A.size();
  REPORT(As);
  vector<int> Ad(3);
  //Ad(0)=A.dim[0]; Ad(1)=A.dim[1]; Ad(2)=A.dim[2];
  Ad=A.dim;
  REPORT(Ad);

  PARAMETER(mu);
  Type nll = pow(mu,2);
  return nll;
}
```

If you know C++, `std::cout<<` also works in the normal way.

Quick demos of TMB in Rstudio

- https://github.com/kaskr/adcomp/wiki/Code--snippets#rstudio-integration
- Followed by a simple linear regression example

Many Species Distribution Models (SDMs) use GLMs as the statistical method



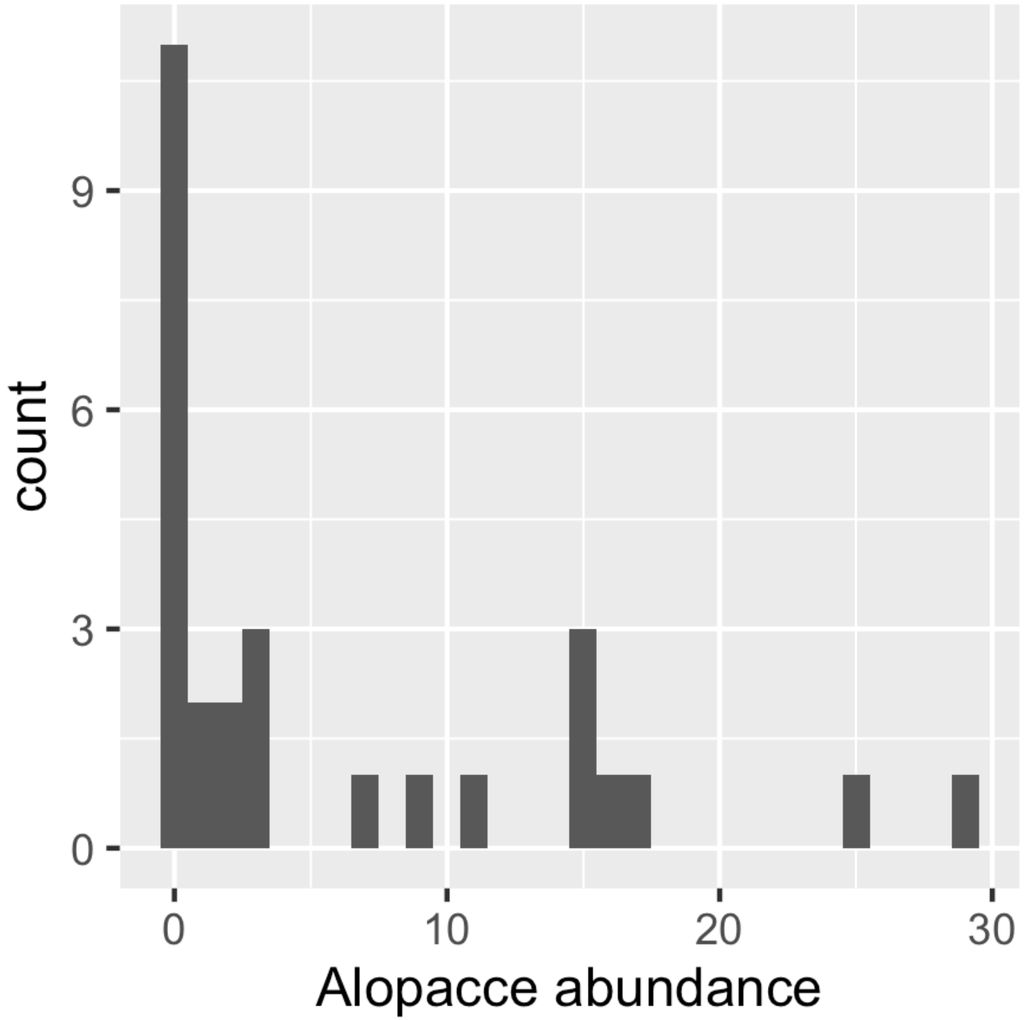
Borrowed from Damaris Zurell https://damariszurell.github.io

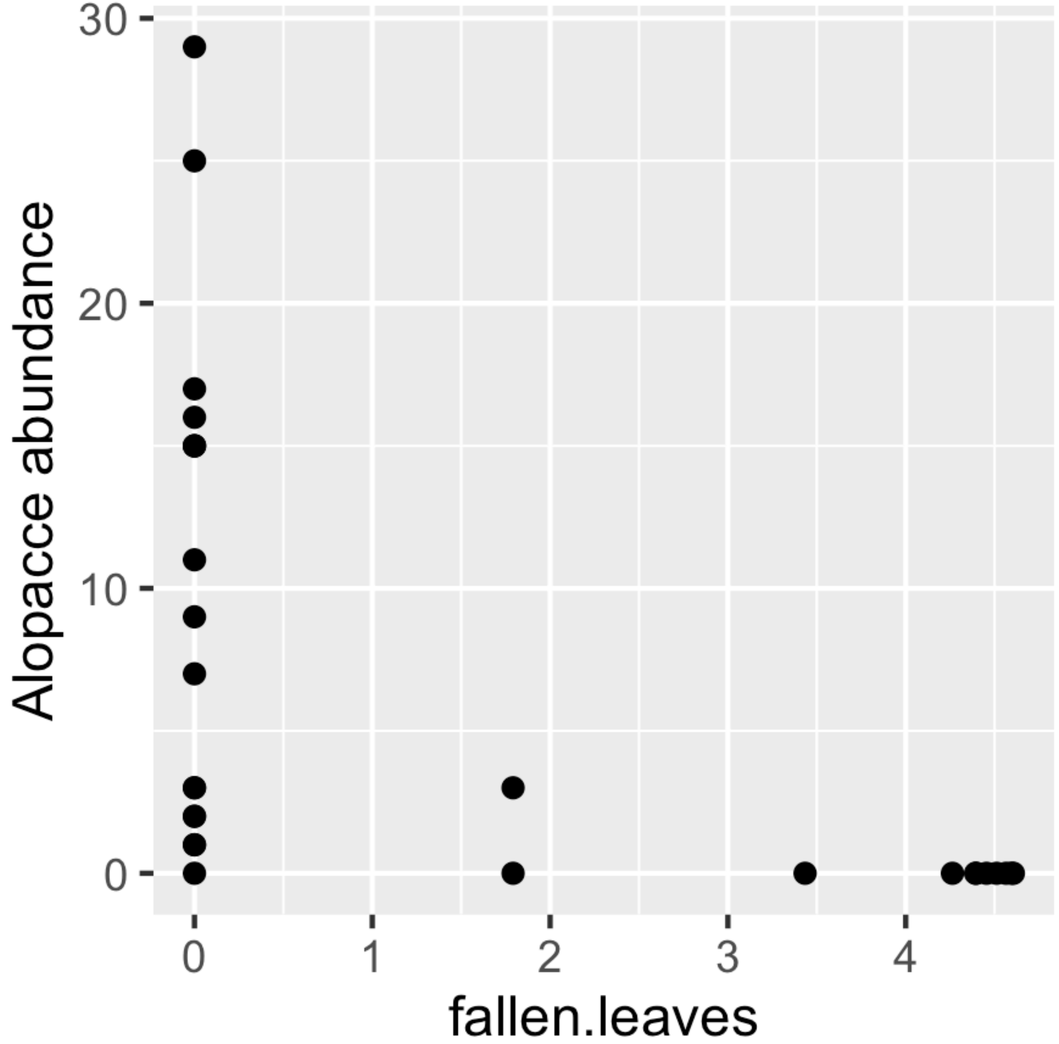
TMB vs glmmTMB

- glmmTMB was written using TMB. It can only fit generalized linear mixed models (including many species distribution models).
- To use either TMB or glmmTMB, you organize your data and run the model in R.
- In TMB you write C++ code which combines data and parameters to calculate the log-likelihood.
- In glmmTMB, the log-likelihood is calculated behind the scenes by pre-written C++ code.

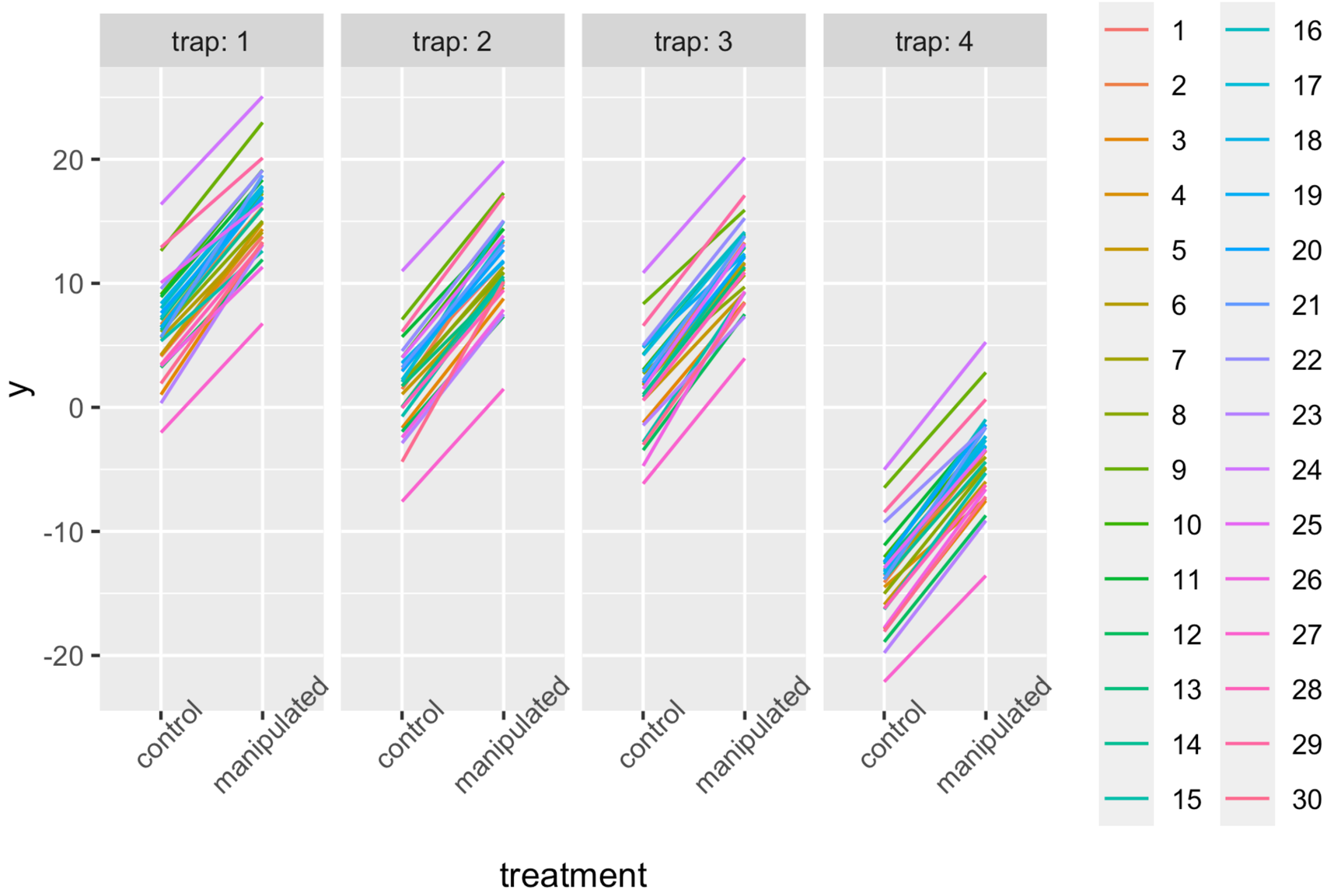
GLM examples in TMB and glmmTMB

Using data from package mvabund (Wang et al. 2021)





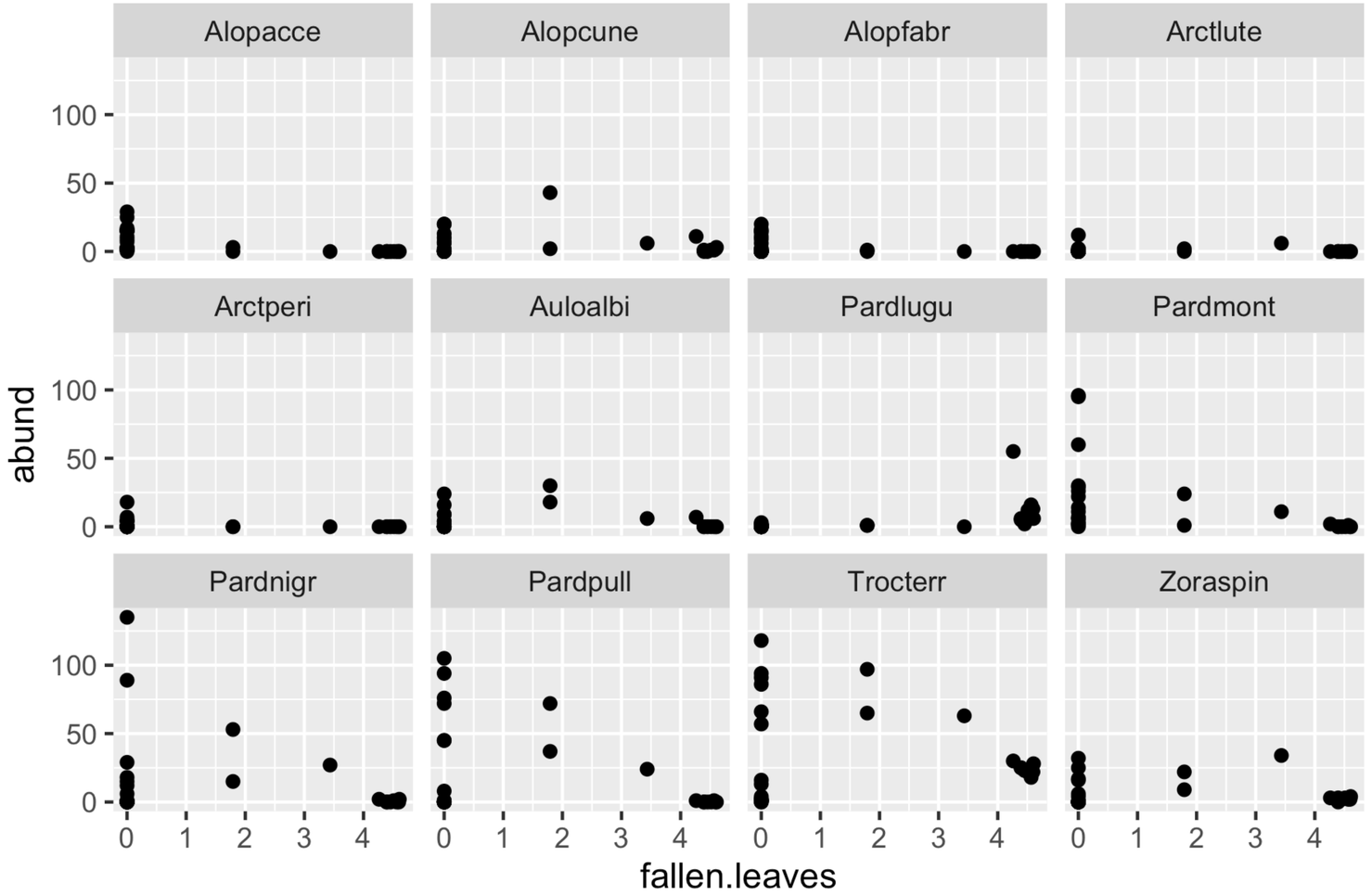
LMM example using simulated data



Joint species distribution model in glmmTMB

Using data from package mvabund (Wang et al. 2021)

and code inspired by Maeve McGillycuddy from vignette("covstruct")



Exercises

- Fir fecundity - The model presented here is found in chapter 6 of Ecological Models and Data in R by Bolker 2008 and the data originally appeared in Silvertown and Dodd 1999, and Dodd and Silvertown 2000. The data on *Abies balsamea* includes counts of the cones produced, measurements of tree size (diameter at breast height, DBH), and the status (wave / non-wave) indicating wave-like die-offs experienced by some populations. The mean fir fecundity is predicted to follow a power-law and the number of cones should be negative binomially distributed around the mean.

$\mu = a\ DBH^{b}$
cones ~NegativeBinomial(μ, k)

- Read and run the Fir fecundity code as it is. Then, change the response distribution; see "R style probability distributions" at http://kaskr.github.io/adcomp/modules.html and consider dpois() and/or dcompois().

- For the GLM code in TMB:

    - Try different design matrices (via model.matrix())

    - Then, see if you can get a dispersion parameter from TMB that matches the one from glmmTMB by outputting k.

    - Then, try messing up different parts of the code and see what error messages look like (e.g. mismatched names, sizes, shorter for-loops).

References

- Cressie, N. A. C., C. A. Calder, J. S. Clark, J. M. Ver Hoef, and C. K. Wikle. 2009. Accounting for uncertainty in ecological analysis: the strengths and limitations of hierarchical statistical modeling. Ecological Applications 19:553–570.

- de Valpine, P. 2009. Shared challenges and common ground for Bayesian and classical analysis of hierarchical statistical models. Ecological Applications 19:584–588.

- Yi Wang, Ulrike Naumann, Dirk Eddelbuettel, John Wilshire and David Warton (2021). mvabund: Statistical Methods for Analysing Multivariate Abundance Data. R package version 4.1.12. https://CRAN.R-project.org/package=mvabund