

Welcoming words

The team

last updated: 2021-05-15

Montpellier and its charms





CORONAVIRUS DISEASE (COVID-19)



zoom

How it all started

 **Olivier Gimenez** 🙋‍♂️
@oaggimenez 

   Inference about animal demographic parameters: Bayesian analysis of capture-recapture data using Jags/Nimble. May 17-18, 2021. Remote workshop. Free of charge. Video-recorded. 🎥

→ Registration is up and running bit.ly/2Migkyd
→ Details in the thread.

 **Olivier Gimenez** 🙋‍♂️ @oaggimenez
  Anyone interested in a (remote) Bayesian workshop on capture-recapture models (single/multistate, multievent models) with Jags and Nimble, in April/May?

9:22 AM · Jan 29, 2021 

 134  9  Copy link to Tweet

Where we're at

[Questions](#) [Réponses 600](#)



Inference about animal demographic parameters:
Bayesian analysis of capture-recapture data using
Jags/Nimble. May 17-18, 2021.

This workshop is about animal demography. We will learn how to infer demographic parameters (e.g. survival, recruitment, or dispersal). We will cover the analysis of capture-recapture data using single-site, multi-site, multi-state and multi-event models. We will adopt a Bayesian approach and use Jags/Nimble for implementation. Hopefully, you'll get what you need to go your own path.

This is a workshop for ecologists. No previous experience with Jags/Nimble, or Bayesian statistics, is assumed, but knowledge of R is required.

The workshop is free of charge. Everything will be video-recorded.

What this workshop is about

- Estimating demographic parameters with capture-recapture.
- Using a family of models called hidden Markov models (HMM).
- Within the Bayesian framework implemented with Markov chain Monte Carlo methods (MCMC).

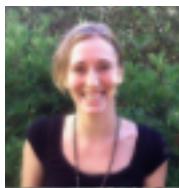
Credits and inspiration

- Past workshops on capture-recapture models w/ Roger Pradel, Rémi Choquet and Jean-Dominique Lebreton.
- Past workshops on Bayesian analyses for population ecology with Ruth King, Steve Brooks and Byron Morgan.
- Workshops on Nimble by Chris Paciorek, Daniel Turek and Perry de Valpine.
- Workshops on integrated population modeling with Michael Schaub and Marc Kéry.
- Books by Marc Kéry, Michael Schaub, Andy Royle and others – check out [curated list](#).
- Daniel Turek's sabbatical in the team.

The team



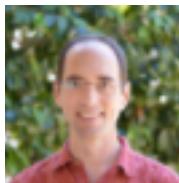
Chloe Rebecca Nater



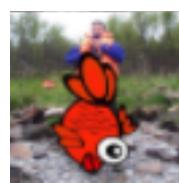
Sarah Cubaynes



Maud Quéroué



Perry de Valpine



Olivier Gimenez

On our plate

Day 1

- Crash course on Bayesian statistics and MCMC algorithms
- Free the modeler in you: Introduction to Nimble
- What you see is not what you get: Hidden Markov models and capture-recapture
- Dead or alive: Survival estimation

Day 2

- On the move: Transition estimation
- Known knowns, unknown knowns and unknowns: Uncertainty in state assignment
- Skip your coffee break: Speed up MCMC convergence
- Take-home messages

Philosophy of teaching

- Lots of attendees, with huge heterogeneity in knowledge of capture-recapture models, Bayesian methods, R and Nimble.
- It is our hope that everyone will find something to take home.
- We've packed a lot of things in two days.
- We do not expect you to digest everything.
- All material (including videos) on website <https://oliviergimenez.github.io/bayesian-cr-workshop/>.
- Feel free to play around with material while we walk through it, and afterwards.
- The workshop is organized in modules, each module is a combination of lectures and live demos.

The way we will interact with each other

- Lectures and live coding demos will happen in Zoom, same link for both days.
- Everything is video recorded.
- Questions and answers via Slack, with a specific channel per module.

Crash course on Bayesian statistics and MCMC algorithms

The team

last updated: 2021-05-11

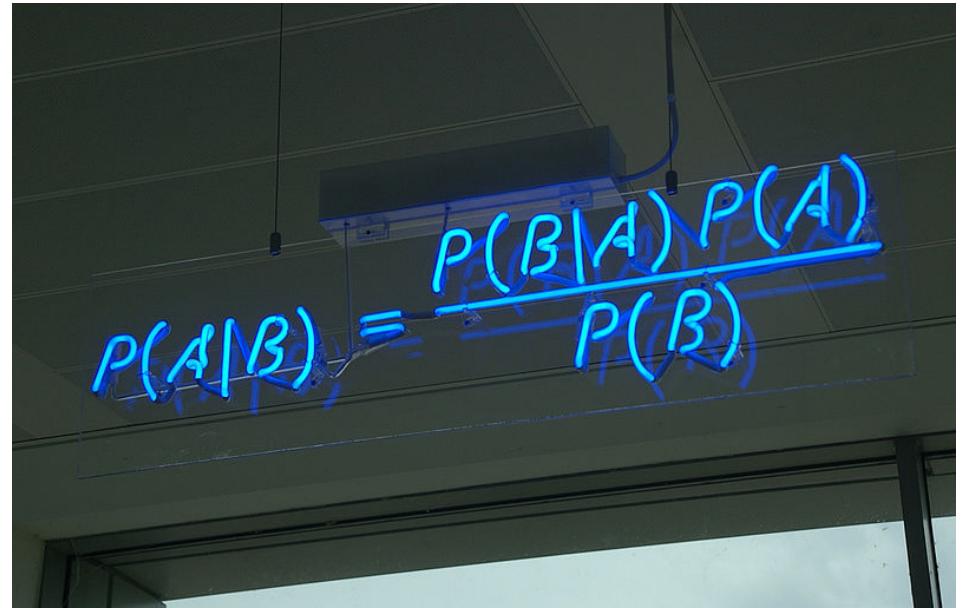
THE AMAZING Thomas Bayes



$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

Bayes' theorem

- A theorem about conditional probabilities.
- $\Pr(B | A) = \frac{\Pr(A | B) \Pr(B)}{\Pr(A)}$



Bayes' theorem spelt out in blue neon at the offices of Autonomy in Cambridge.
Source: Wikipedia

Bayes' theorem

- I always forget what the letters mean.
- Might be easier to remember when written like this:

$$\Pr(\text{hypothesis} \mid \text{data}) = \frac{\Pr(\text{data} \mid \text{hypothesis}) \Pr(\text{hypothesis})}{\Pr(\text{data})}$$

- The "hypothesis" is typically something unobserved or unknown. It's what you want to learn about using the data.
- For regression models, the "hypothesis" is a parameter (intercept, slopes or error terms).
- Bayes theorem tells you the probability of the hypothesis given the data.

What is doing science after all?

How plausible is some hypothesis given the data?

$$\Pr(\text{hypothesis} \mid \text{data}) = \frac{\Pr(\text{data} \mid \text{hypothesis}) \Pr(\text{hypothesis})}{\Pr(\text{data})}$$

Why is Bayesian statistics not the default?

- Due to practical problems of implementing the Bayesian approach, and futile wars between (male) statisticians, little progress was made for over two centuries.
- Recent advances in computational power coupled with the development of new methodology have led to a great increase in the application of Bayesian methods within the last two decades.

Frequentist versus Bayesian

- Typical stats problems involve estimating parameter θ with available data.
- The frequentist approach (maximum likelihood estimation – MLE) assumes that the parameters are fixed, but have unknown values to be estimated.
- Classical estimates are generally point estimates of the parameters of interest.
- The Bayesian approach assumes that the parameters are not fixed but have some fixed unknown distribution - a distribution for the parameter.

What is the Bayesian approach?

- The approach is based upon the idea that the experimenter begins with some prior beliefs about the system.
- And then updates these beliefs on the basis of observed data.
- This updating procedure is based upon the Bayes' Theorem:

$$\Pr(A | B) = \frac{\Pr(B | A) \Pr(A)}{\Pr(B)}$$

What is the Bayesian approach?

- Schematically if $A = \theta$ and $B = \text{data}$, then
- The Bayes' theorem

$$\Pr(A | B) = \frac{\Pr(B | A) \Pr(A)}{\Pr(B)}$$

- Translates into:

$$\Pr(\theta | \text{data}) = \frac{\Pr(\text{data} | \theta) \ Pr(\theta)}{\Pr(\text{data})}$$

Bayes' theorem

$$\Pr(\theta \mid \text{data}) = \frac{\Pr(\text{data} \mid \theta) \Pr(\theta)}{\Pr(\text{data})}$$

- **Posterior distribution:** Represents what you know after having seen the data. The basis for inference, a distribution, possibly multivariate if more than one parameter.
- **Likelihood:** This quantity is the same as in the MLE approach.
- **Prior distribution:** Represents what you know before seeing the data. The source of much discussion about the Bayesian approach.
- $\Pr(\text{data}) = \int L(\text{data} \mid \theta) \Pr(\theta) d\theta$ is a N -dimensional integral if $\theta = \theta_1, \dots, \theta_N$.
- Difficult if not impossible to calculate. This is one of the reasons why we need simulation (MCMC) methods.

Brute force via numerical integration

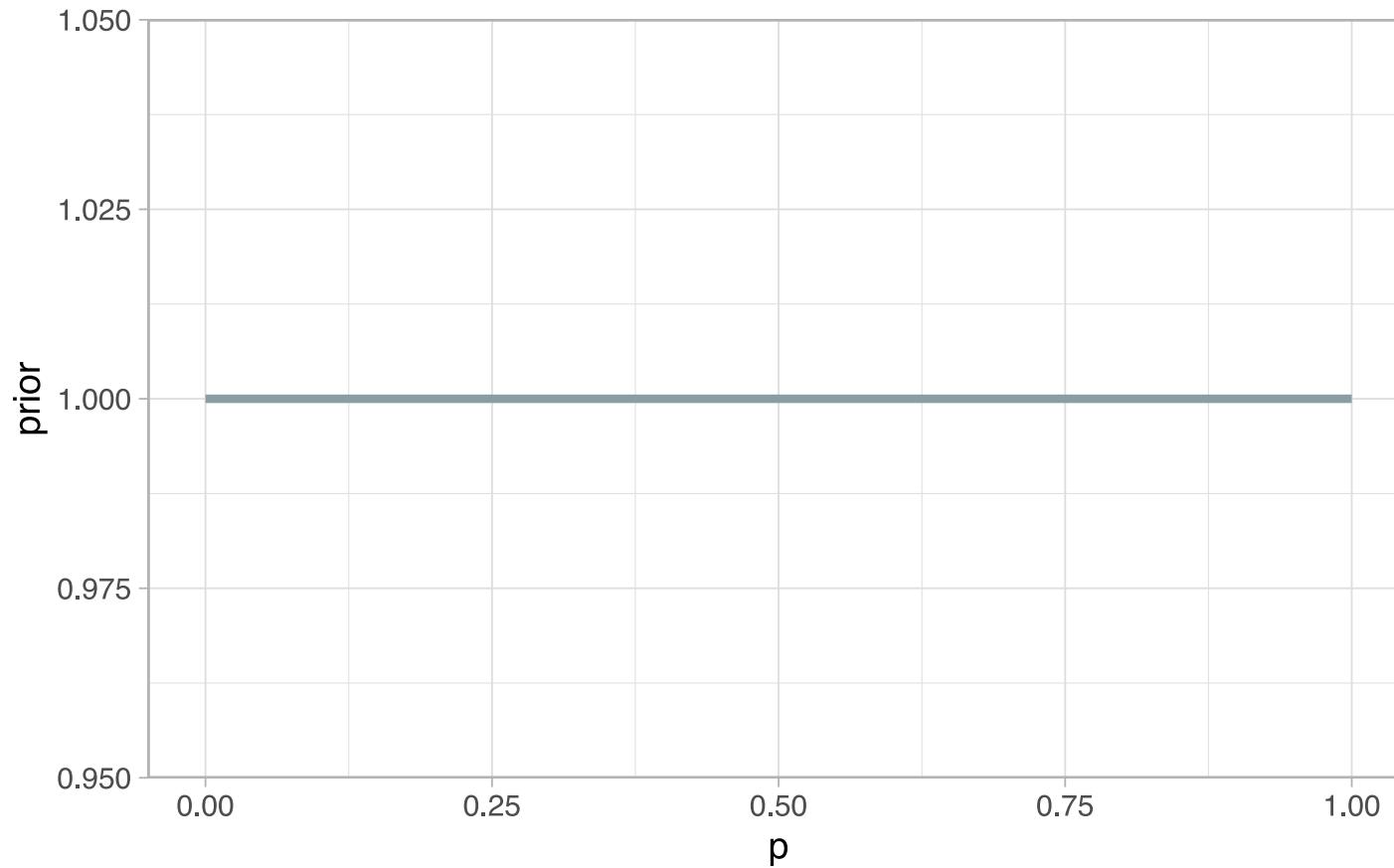
- Say we release n animals at the beginning of the winter, out of which y survive, and we'd like to estimate winter survival θ .

```
y <- 19 # nb of success  
n <- 57 # nb of attempts
```

- Our model:

$$\begin{array}{ll} y \sim \text{Binomial}(n, \theta) & [\text{likelihood}] \\ \theta \sim \text{Beta}(1, 1) & [\text{prior for } \theta] \end{array}$$

Beta prior



Apply Bayes theorem

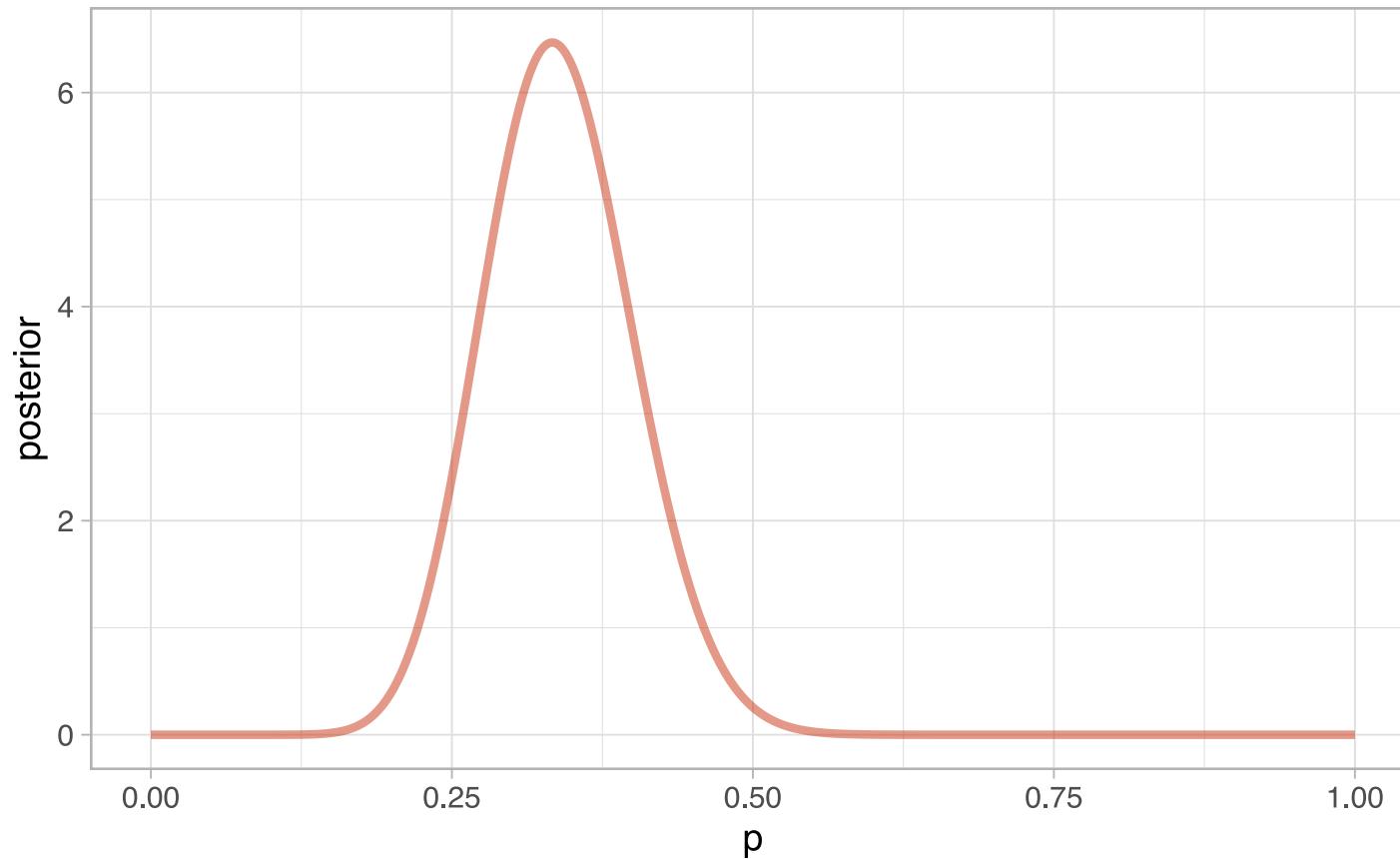
- Likelihood times the prior: $\Pr(\text{data} \mid \theta) \Pr(\theta)$

```
numerator <- function(p) dbinom(y,n,p) * dbeta(p,a,b)
```

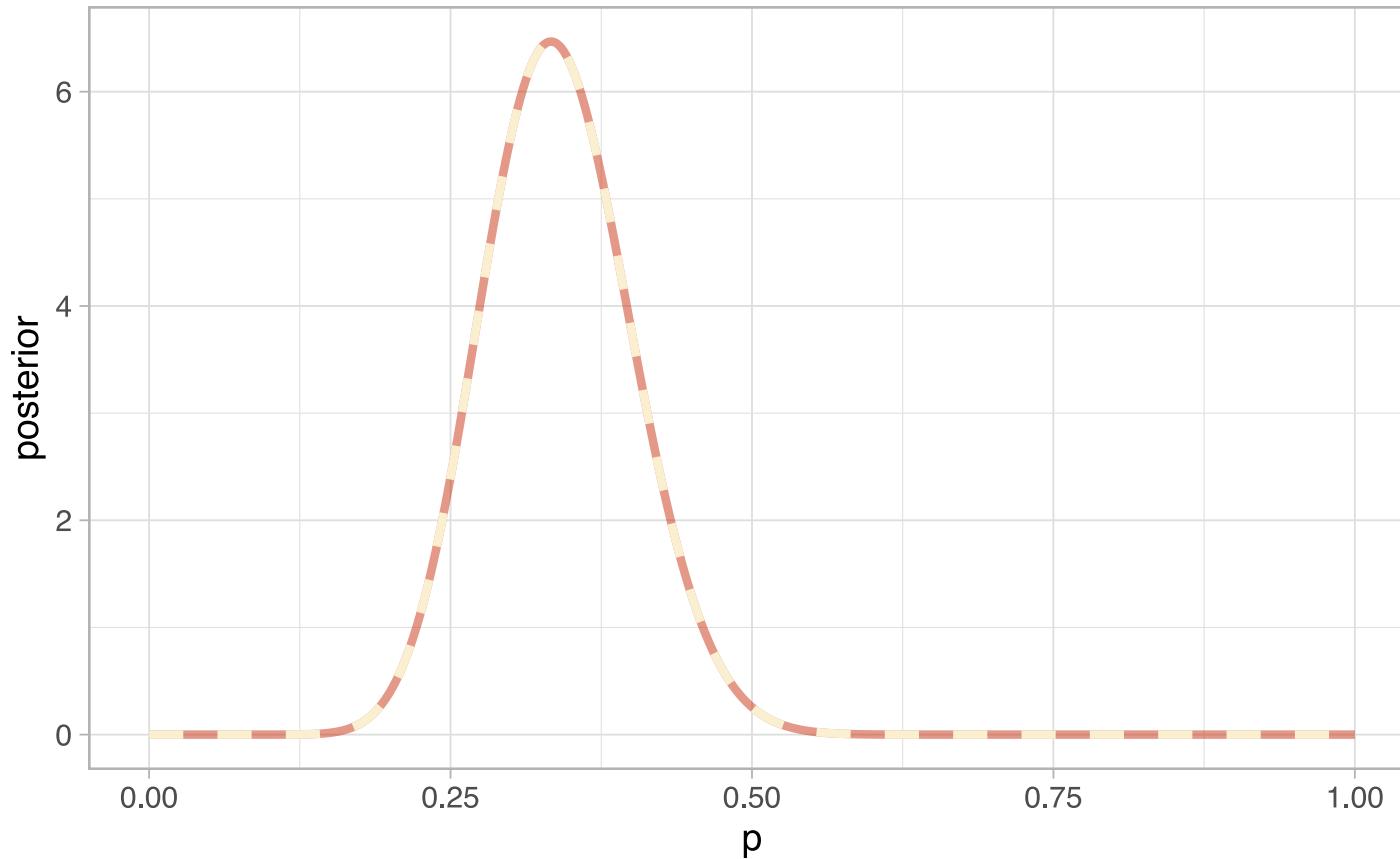
- Averaged likelihood: $\Pr(\text{data}) = \int L(\theta \mid \text{data}) \Pr(\theta) d\theta$

```
denominator <- integrate(numerator, 0, 1)$value
```

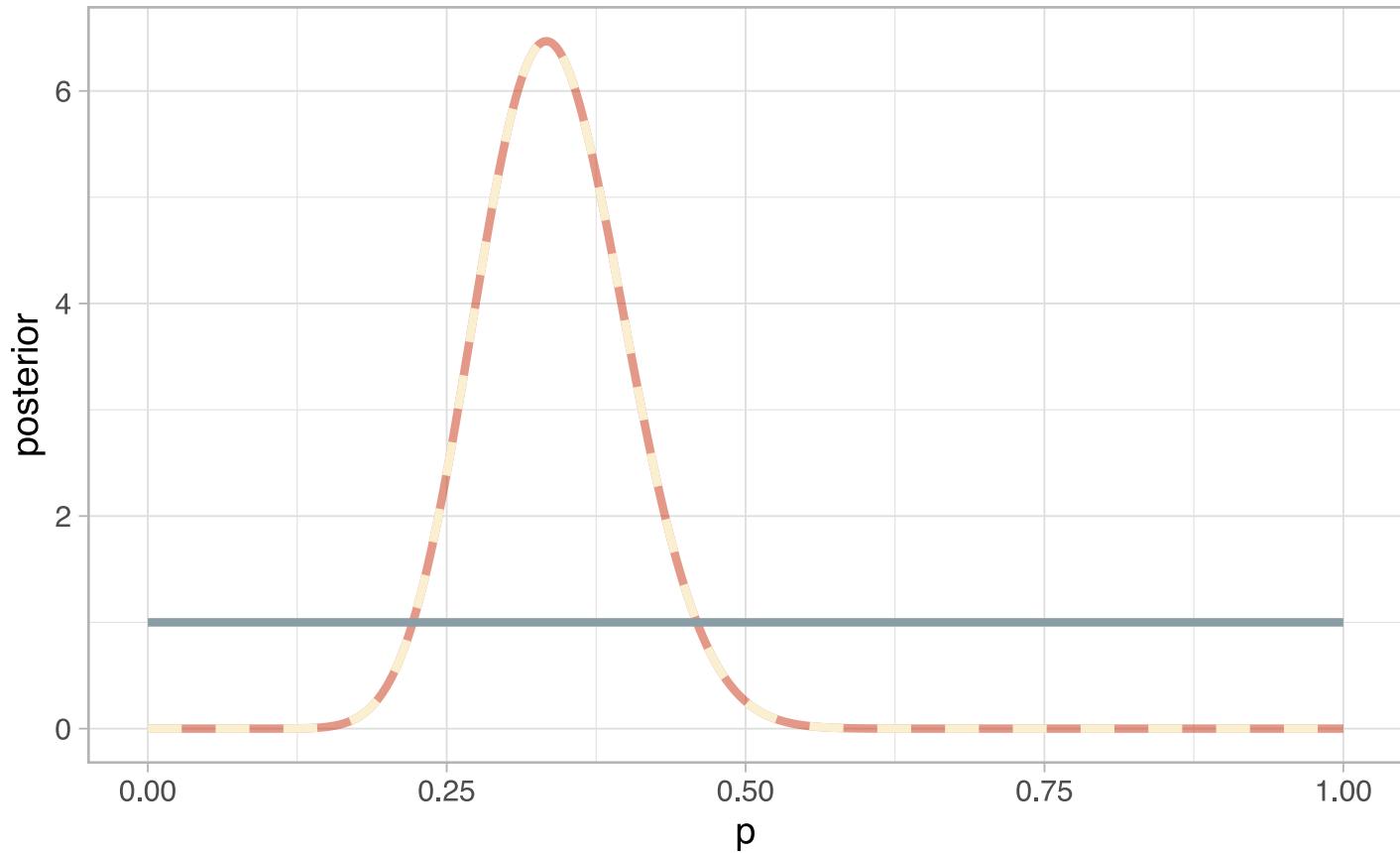
Posterior via numerical integration



Superimpose explicit posterior



And the prior



What if multiple parameters?

- Example of a linear regression with parameters α , β and σ to be estimated.
- Bayes' theorem says:

$$P(\alpha, \beta, \sigma \mid \text{data}) = \frac{P(\text{data} \mid \alpha, \beta, \sigma) P(\alpha, \beta, \sigma)}{\iiint P(\text{data} \mid \alpha, \beta, \sigma) P(\alpha, \beta, \sigma) d\alpha d\beta d\sigma}$$

- Do we really wish to calculate a 3D integral?

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics.

THE JOURNAL OF CHEMICAL PHYSICS

VOLUME 21, NUMBER 6

JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

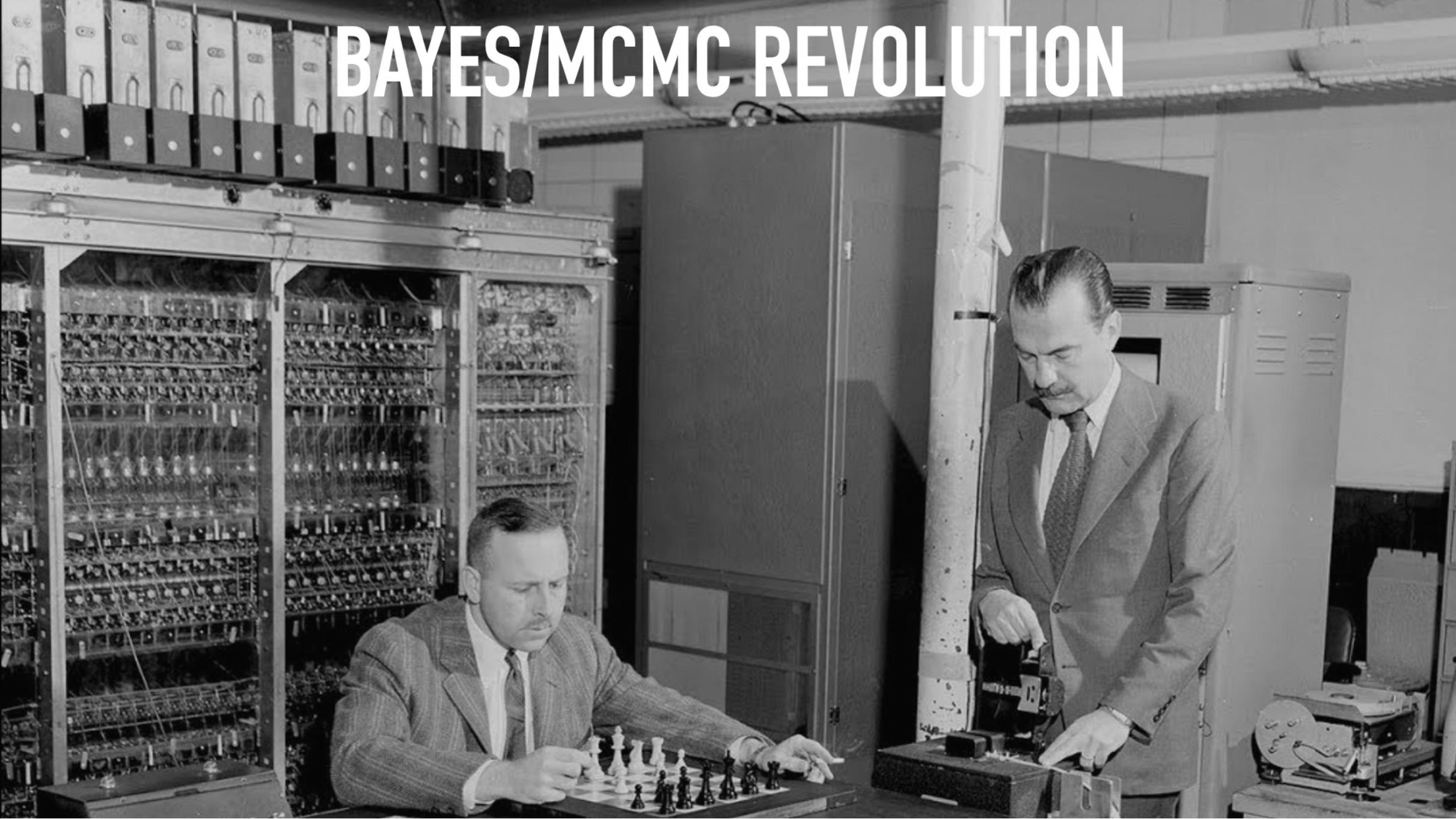
EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics.
- Use stochastic simulation to draw samples from posterior distributions.
- Avoid explicit calculation of integrals in Bayes formula.
- Instead, approx. posterior w/ some precision by drawing large samples.
- Markov chain Monte Carlo (MCMC) gives a boost to Bayesian statistics!

BAYES/MCMC REVOLUTION



Maniac

450 kg

5 Ko bites of memory

70k multiplications per second

Your laptop

2-5 kg

2-8 millions Ko bites of memory

Billions multiplications per second



Why are MCMC methods so useful?

- MCMC are stochastic algorithms to produce sequence of dependent random numbers from a Markov chain.
- A Markov chain is a discrete sequence of states, in which the probability of an event depends only on the state in the previous event.
- A Markov chain has an equilibrium (aka stationary) distribution.
- Equilibrium distribution is the desired posterior distribution!
- Several ways of constructing these chains: e.g., Metropolis-Hastings, Gibbs sampler.
- How to implement them in practice?!

The Metropolis algorithm

- Let's go back to animal survival estimation.
- We illustrate sampling from the posterior distribution.
- We write functions in **R** for the likelihood, the prior and the posterior.

```
# survival data, 19 "success" out of 57 "attempts"
survived <- 19
released <- 57

# log-likelihood function
loglikelihood <- function(x, p){
  dbinom(x = x, size = released, prob = p, log = TRUE)
}

# prior density
logprior <- function(p){
  dunif(x = p, min = 0, max = 1, log = TRUE)
}

# posterior density function (log scale)
posterior <- function(x, p){
  loglikelihood(x, p) + logprior(p) # - log(Pr(data))
}
```

Metropolis algorithm

1. We start at any possible value of the parameter to be estimated.
2. To decide where to visit next, we propose to move away from the current value of the parameter – this is a **candidate** value. To do so, we add to the current value some random value from say a normal distribution with some variance.
3. We compute the ratio of the probabilities at the candidate and current locations $R = \text{posterior}(\text{candidate})/\text{posterior}(\text{current})$. This is where the magic of MCMC happens, in that $\Pr(\text{data})$, the denominator of the Bayes theorem, cancels out.
4. We spin a continuous spinner that lands anywhere from 0 to 1 – call it the random spin X . If X is smaller than R , we move to the candidate location, otherwise we remain at the current location.
5. We repeat 2-4 a number of times – or **steps** (many steps).

```
# propose candidate value
move <- function(x, away = .2){
  logitx <- log(x / (1 - x))
  logit_candidate <- logitx + rnorm(1, 0, away)
  candidate <- plogis(logit_candidate)
  return(candidate)
}

# set up the scene
steps <- 100
theta.post <- rep(NA, steps)
set.seed(1234)

# pick starting value (step 1)
inits <- 0.5
theta.post[1] <- inits
```

```
for (t in 2:steps){ # repeat steps 2-4 (step 5)

  # propose candidate value for prob of success (step 2)
  theta_star <- move(theta.post[t-1])

  # calculate ratio R (step 3)
  pstar <- posterior(survived, p = theta_star)
  pprev <- posterior(survived, p = theta.post[t-1])
  logR <- pstar - pprev
  R <- exp(logR)

  # decide to accept candidate value or to keep current value (:)
  accept <- rbinom(1, 1, prob = min(R, 1))
  theta.post[t] <- ifelse(accept == 1, theta_star, theta.post[t-1])
}
```

Starting at the value 0.5 and running the algorithm for 100 iterations.

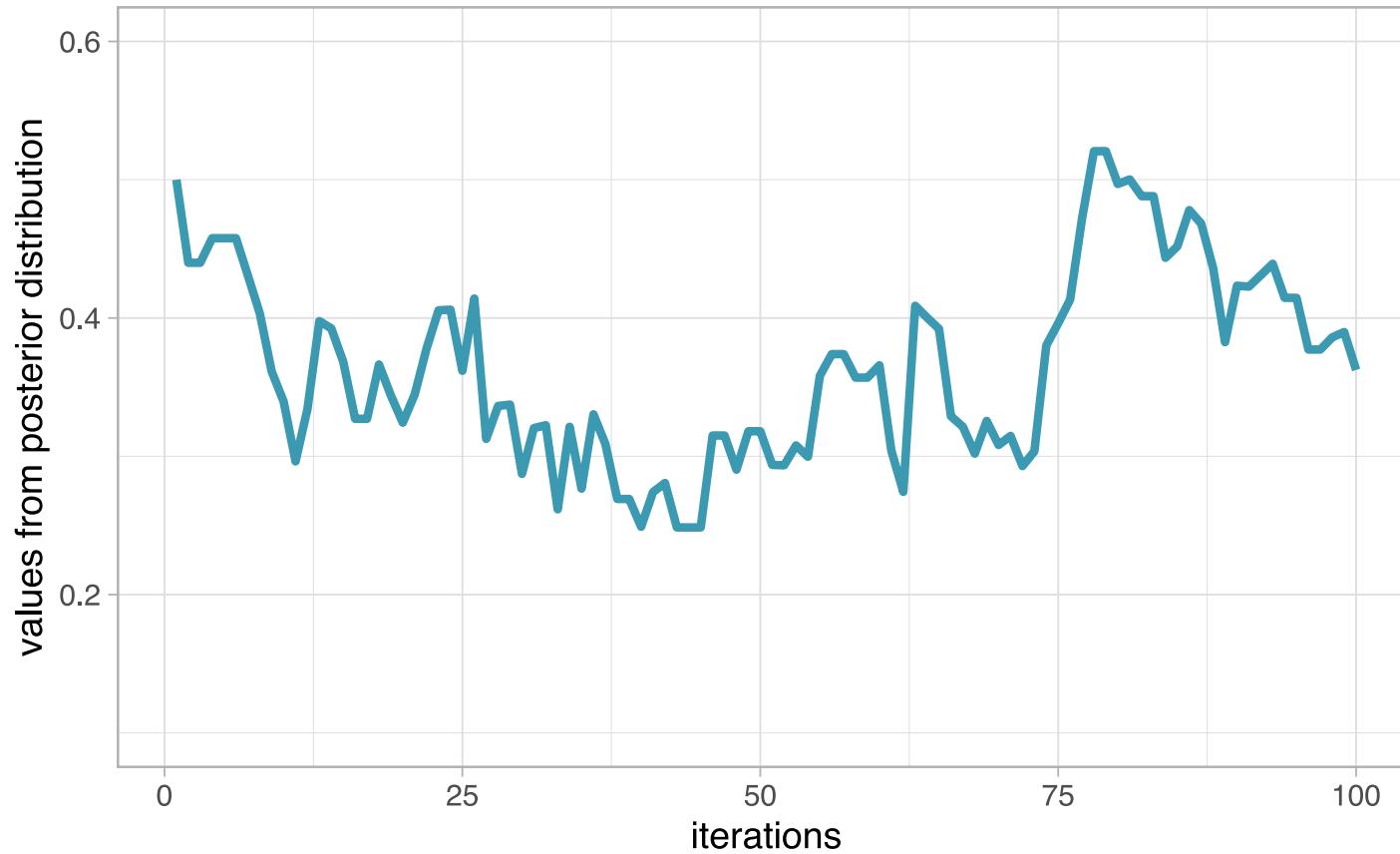
```
head(theta.post)
```

```
[1] 0.5000000 0.4399381 0.4399381 0.4577124 0.4577124 0.4577124
```

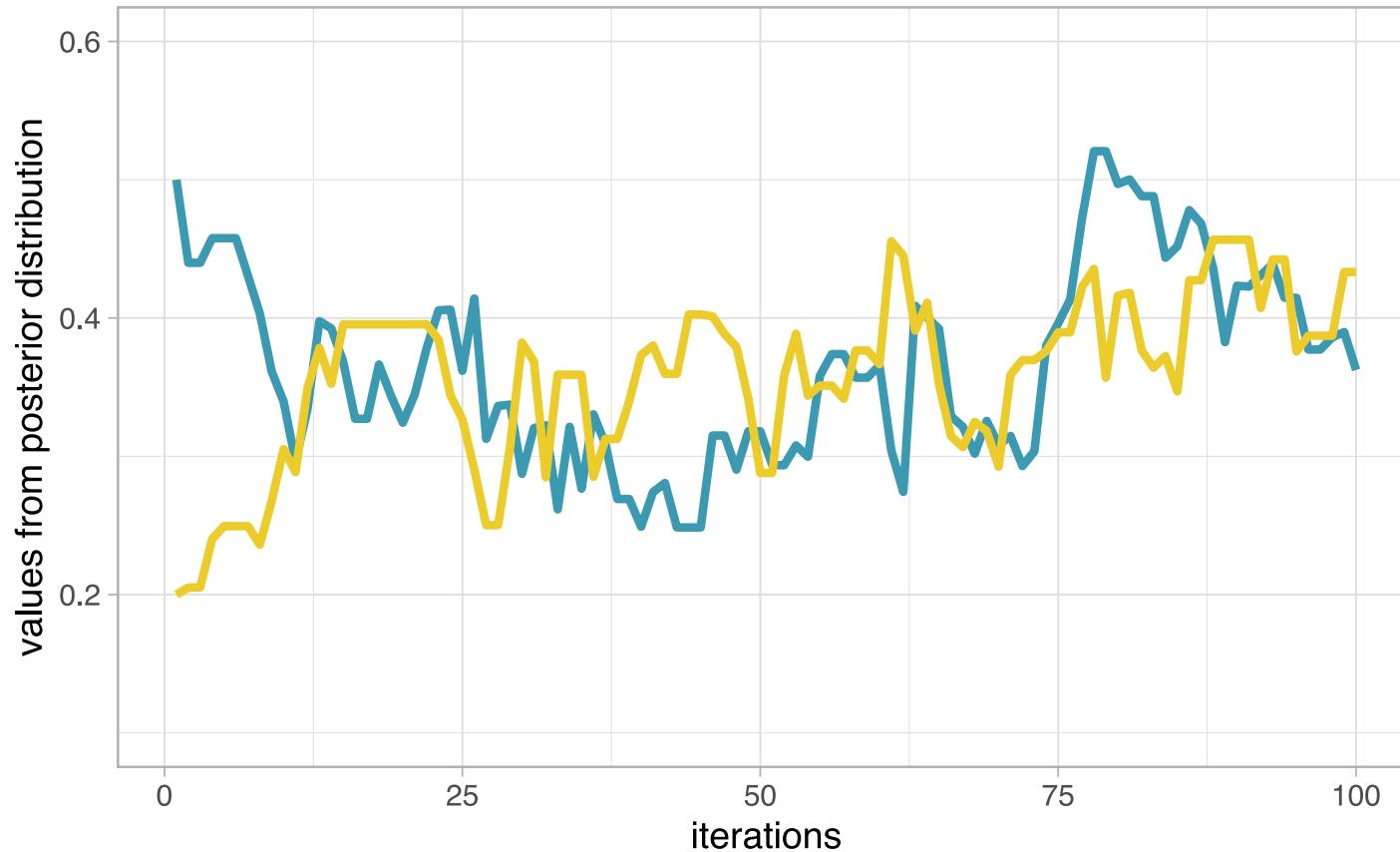
```
tail(theta.post)
```

```
[1] 0.4145878 0.3772087 0.3772087 0.3860516 0.3898536 0.3624450
```

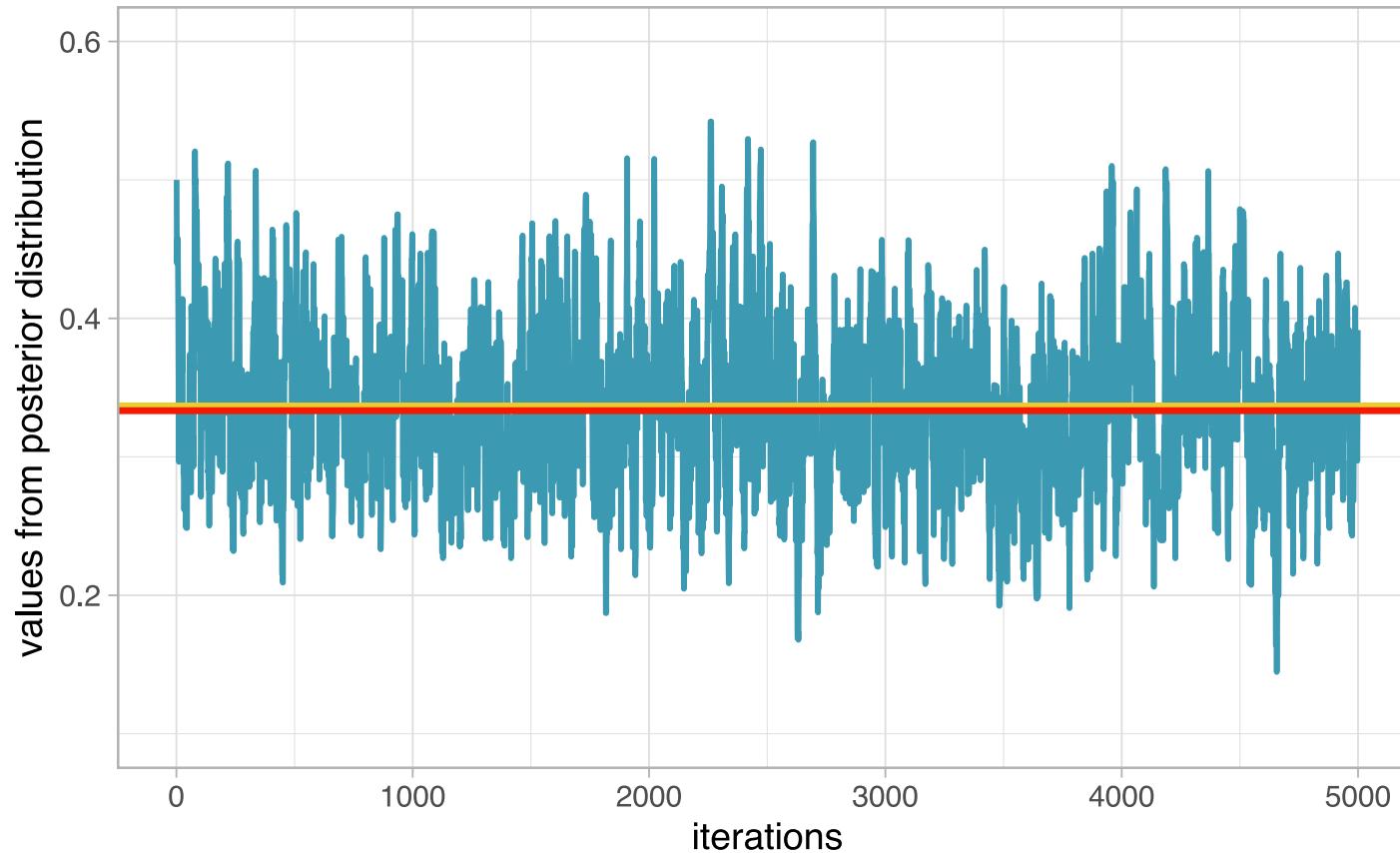
A chain



Another chain

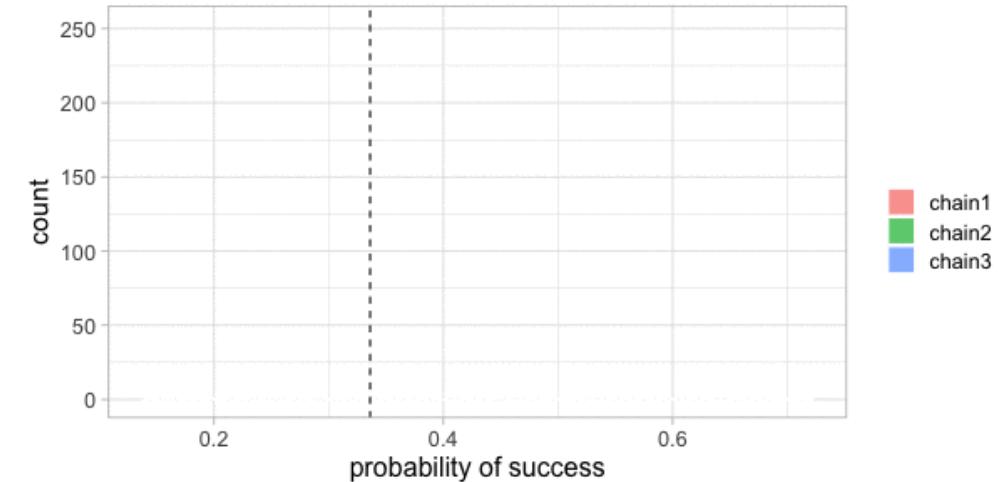
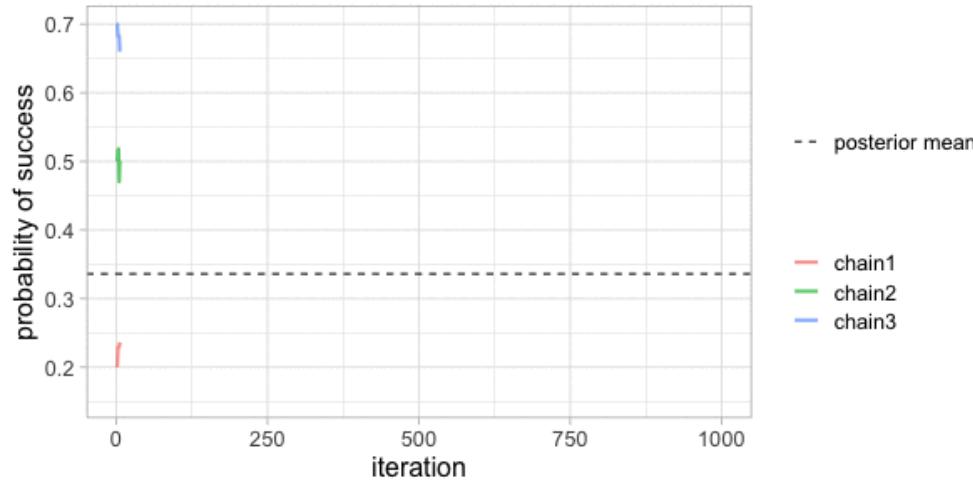


With 5000 steps



In yellow: posterior mean; in red: maximum likelihood estimate.

Animating MCMC - 1D example (code [here](#))



Animating MCMC - 2D example (code [here](#))

The MCMC Interactive Gallery (more [here](#))

The Markov-chain Monte Carlo Interactive Gallery

Click on an algorithm below to view interactive demo:

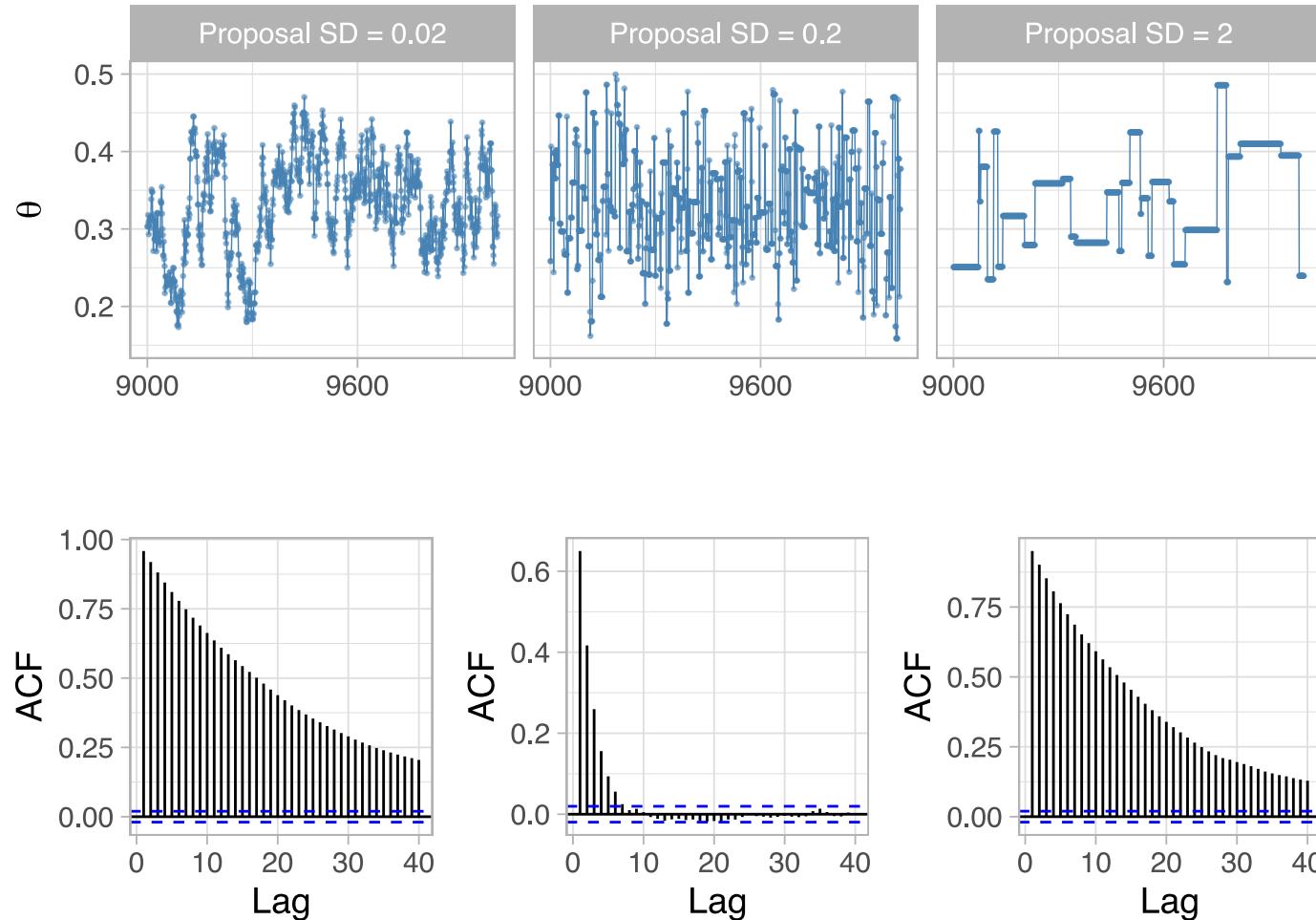
- [Random Walk Metropolis Hastings](#)
- [Adaptive Metropolis Hastings \[1\]](#)
- [Hamiltonian Monte Carlo \[2\]](#)
- [No-U-Turn Sampler \[2\]](#)
- [Metropolis-adjusted Langevin Algorithm \(MALA\) \[3\]](#)
- [Hessian-Hamiltonian Monte Carlo \(H2MC\) \[4\]](#)
- [Gibbs Sampling](#)
- [Stein Variational Gradient Descent \(SVGD\) \[5\]](#)
- [Nested Sampling with RadFriends \(RadFriends-NS\) \[6\]](#)
- [Differential Evolution Metropolis \(Z\) \[7\]](#)

View the source code on github: <https://github.com/chi-feng/mcmc-demo>.

Assessing convergence

- MCMC algorithms can be used to construct a Markov chain with a given stationary distribution (set to be the posterior distribution).
- For the MCMC algorithm, the posterior distribution is only needed to be known up to proportionality.
- Once the stationary distribution is reached, we can regard the realisations of the chain as a (dependent) sample from the posterior distribution (and obtain Monte Carlo estimates).
- We consider some important implementation issues.

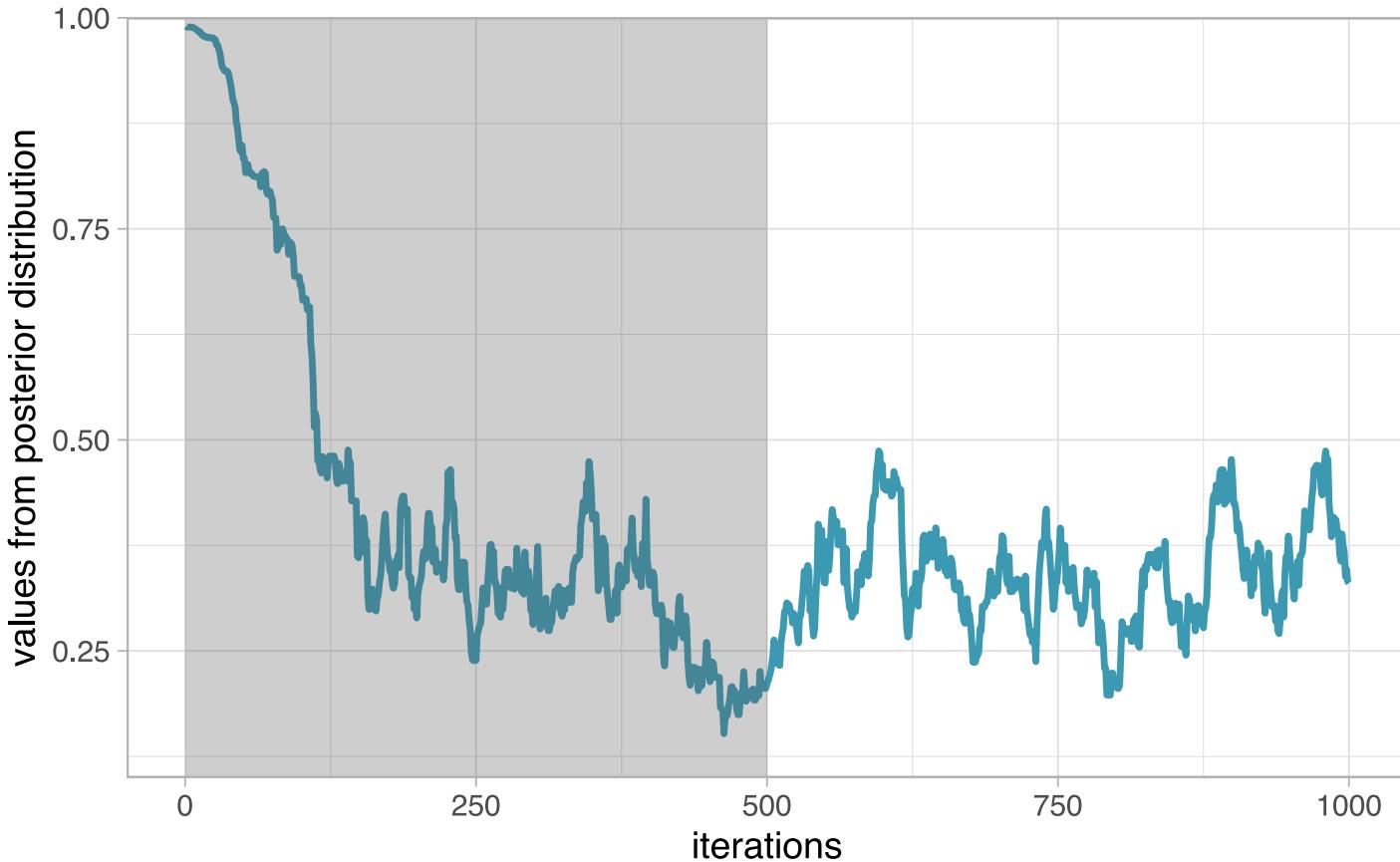
Mixing and autocorrelation



How do good chains behave?

- Converge to same target distribution; discard some realisations of Markov chain before convergence is achieved.
- Once there, explore efficiently: The post-convergence sample size required for suitable numerical summaries.
- Therefore, we are looking to determine how long it takes for the Markov chain to converge to the stationary distribution.
- In practice, we must discard observations from the start of the chain and just use observations from the chain once it has converged.
- The initial observations that we discard are referred to as the **burn-in**.
- Simplest method to determine length of burn-in period is to look at trace plots.

Burn-in



If simulations cheap, be conservative.

Effective sample size `n.eff`

- How long of a chain is needed to produce stable estimates ?
- Most MCMC chains are strongly autocorrelated.
- Successive steps are near each other, and are not independent.
- The effective sample size (`n.eff`) measures chain length while taking into account the autocorrelation of the chain.
 - `n.eff` is less than the number of MCMC iterations.
 - Check the `n.eff` of every parameter of interest.
 - Check the `n.eff` of any interesting parameter combinations.
- We need $n.eff \geq 100$ independent steps.

Potential scale reduction factor

- Gelman-Rubin statistic \hat{R}
- Measures the ratio of the total variability combining multiple chains (between-chain plus within-chain) to the within-chain variability.
- Asks the question is there a chain effect? Very much alike the F test in an ANOVA.
- Values near 1 indicates likely convergence, a value of ≤ 1.1 is considered acceptable.
- Necessary condition, not sufficient; In other words, these diagnostics cannot tell you that you have converged for sure, only that you have not.

To sum up

- Run multiple chains from arbitrary starting places (initial values).
- Assume convergence when all chains reach same regime
- Discard initial burn-in phase.
- Proceed with posterior inference.
- Use traceplot, effective sample size and \hat{R} .

What if you have issues of convergence?

- Increase burn-in, sample more.
- Use more informative priors.
- Pick better initial values (good guess), using e.g. estimates from simpler models.
- Reparameterize:
 - Standardize covariates.
 - Non-centering: $\alpha \sim N(0, \sigma)$ becomes $\alpha = z\sigma$ with $z \sim N(0, 1)$.
- Something wrong with your model?
 - Start with a simpler model (remove complexities).
 - Use simulations.
- Change your sampler. More later on.

Further reading

- McCarthy, M. (2007). *Bayesian Methods for Ecology*. Cambridge: Cambridge University Press.
- McElreath, R. (2020). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan (2nd ed.)*. CRC Press.
- Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models (Analytical Methods for Social Research)*. Cambridge: Cambridge University Press.

Live demo



‘Free the modeler in you’: Intro to Nimble

The team (citation by Marc Kéry)

last updated: 2021-05-17

What is Nimble?



Rob Robinson
[@btorobrob](https://twitter.com/btorobrob)

...

#rstats particularly the JAGS one, some of those who have ventured into that shadowy place have not returned... but rather pushed on through to the paradise that is @R_nimble or so the legends go...
Thanks [@Todd_W_Arnold](https://twitter.com/Todd_W_Arnold) for the smile

(Meme created by Todd Arnold's wonderful students)

What is Nimble?

- Numerical Inference for statistical Models using Bayesian and Likelihood Estimation.
- A framework for hierarchical statistical models and algorithms.
- Uses almost the same model code as WinBUGS, OpenBUGS, and JAGS.
- An extension of the BUGS language: additional syntax, custom functions and distributions.
- A configurable system for MCMC.
- A library of other methods (SMC, MCEM).
- A model-generic programming system to write new analysis methods.

Load nimble package

```
library(nimble)
```

Build model, made of likelihood and priors

```
naive.survival.model <- nimbleCode( 
  # prior
  phi ~ dunif(0, 1)
  # likelihood
  y ~ dbinom(phi, n)
)})
```

Syntax: what's new/better/different?

- Vectorization

```
# JAGS (& Nimble)
for(t in 1:Tmax){
  x[t] <- Mu.x + epsilon[t]
}

# Nimble
x[1:Tmax] <- Mu.x + epsilon[1:Tmax]
```

Syntax: what's new/better/different?

- More flexible specification of distributions

```
# JAGS (& Nimble)
for(t in 1:Tmax){
  epsilon[t] ~ dnorm(0, tau)
}
tau <- pow(sigma, -2)
sigma ~ dunif(0, 5)

# Nimble
for(t in 1:Tmax){
  epsilon[t] ~ dnorm(0, sd = sigma)
}
sigma ~ dunif(0, 5)
```

Syntax: what's new/better/different?

- Your own functions and distributions

```
x[1:Tmax] <- myNimbleFunction(a = Mu.x, b = epsilon[1:Tmax])
```

```
sigma ~ dCustomDistr(c = 0.5, z = 10)
```

Syntax: what's new/better/different?

- The end of empty indices

```
# JAGS
sum.x <- sum(x[ ])

# Nimble
sum.x <- sum(x[1:Tmax])
```

- & more...

Read in data

Back to our naive survival model:

```
naive.survival.model <- nimbleCode( {  
  # prior  
  phi ~ dunif(0, 1)  
  # likelihood  
  y ~ dbinom(phi, n)  
})
```

```
my.data <- list(n = 57, y = 19)
```

Distinguish constants and data

To Nimble, not all "data" is data...

```
my.constants <- list(n = 57)
my.data <- list(y = 19)
```

Constants:

- Can never be changed
- Must be provided when a model is defined (part of the model structure)
- E.g. vector of known index values, variables used to define for-loops, etc.

Distinguish constants and data

To Nimble, not all "data" is data...

```
my.constants <- list(n = 57)
my.data <- list(y = 19)
```

Data:

- Can be changed without re-building the model
- Can be (re-)simulated within a model
- E.g. stuff that *only* appears to the left of a "~"

For computational efficiency, better to specify as much as possible as constants.

Nimble will help you with this!

Specify initial values

```
initial.values <- function() list(phi = runif(1,0,1))
```

```
initial.values()
```

```
$phi  
[1] 0.9287372
```

Which parameters to save?

```
parameters.to.save <- c("phi")
```

MCMC details

```
n.iter <- 5000  
n.burnin <- 1000  
n.chains <- 2  
n.thin <- 1
```

Number of posterior samples per chain:

$$n.\text{posterior} = \frac{n.\text{iter} - n.\text{burnin}}{n.\text{thin}}$$

Run model, tadaa!

```
mcmc.output <- nimbleMCMC(code = naive.survival.model,
                             data = my.data,
                             constants = my.constants,
                             inits = initial.values,
                             monitors = parameters.to.save,
                             thin = n.thin,
                             niter = n.iter,
                             nburnin = n.burnin,
                             nchains = n.chains)
```

Explore MCMC outputs

```
str(mcmc.output)
```

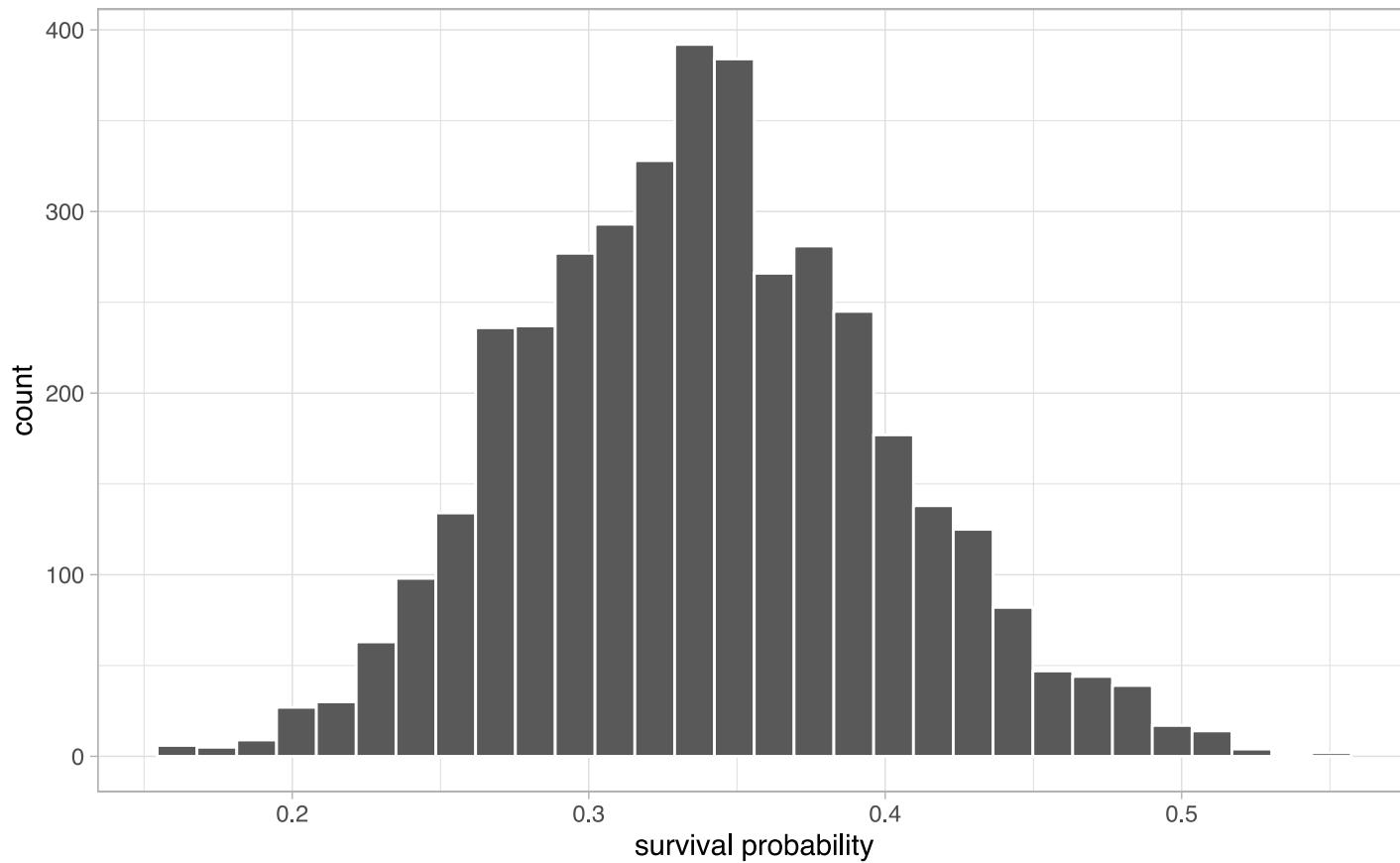
```
List of 2
$ chain1: num [1:4000, 1] 0.406 0.43 0.264 0.264 0.297 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr "phi"
$ chain2: num [1:4000, 1] 0.274 0.274 0.389 0.389 0.389 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr "phi"
```

Explore MCMC outputs

```
head(mcmc.output$chain1)
```

```
    phi
[1,] 0.4057891
[2,] 0.4297459
[3,] 0.2644534
[4,] 0.2644534
[5,] 0.2971877
[6,] 0.2971877
```

Explore MCMC outputs



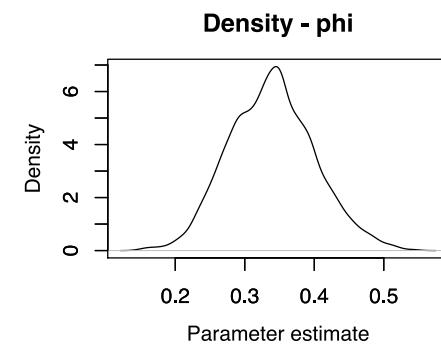
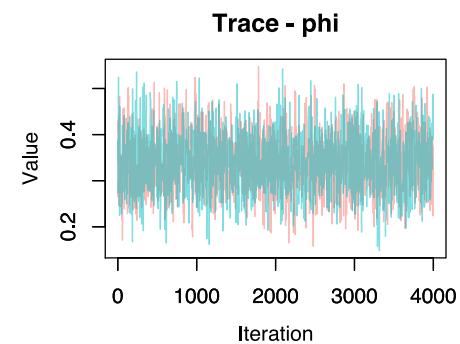
Numerical summaries

```
library(MCMCvis)
MCMCsummary(mcmc.output, round = 2)
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
phi	0.34	0.06	0.23	0.34	0.47	1	1793

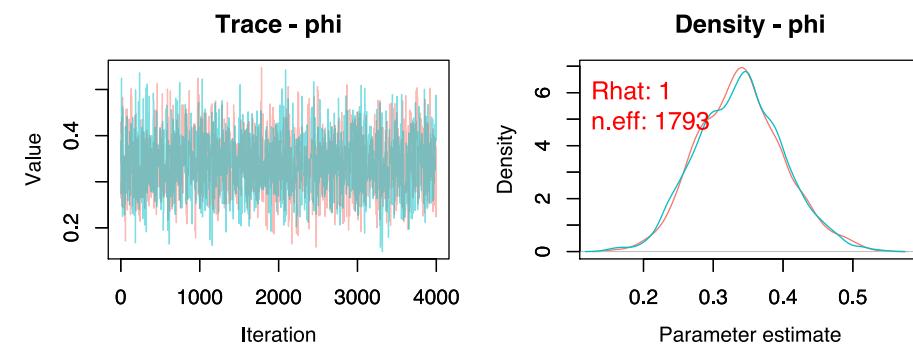
Trace and posterior density

```
MCMCtrace(mcmc.output,  
           pdf = FALSE)
```



Trace and posterior density

```
MCMCtrace(mcmc.output,  
           pdf = FALSE,  
           ind = TRUE,  
           Rhat = TRUE,  
           n.eff = TRUE)
```

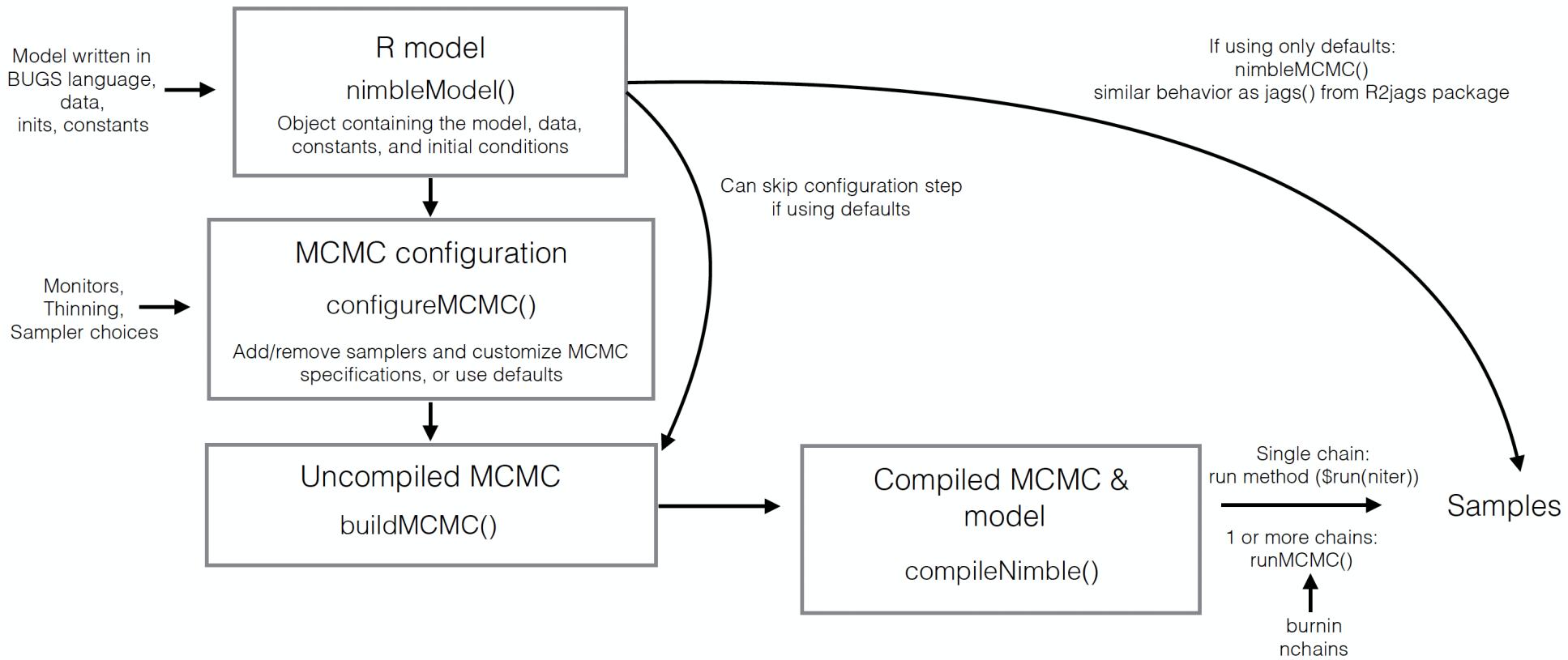


Our nimble workflow so far



Adapted from L. Ponisio

But nimble gives full access to the MCMC engine



Credit: L. Ponisio



Don't even worry about it

Useful resources

- Official website <https://r-nimble.org>
- User Manual https://r-nimble.org/html_manual/cha>Welcome-nimble.html and [cheatsheet](#).
- Users mailing list <https://groups.google.com/forum/#!forum/nimble-users>
- Training material <https://github.com/nimble-training>
- Reference to cite when using nimble in a publication:

de Valpine, P., D. Turek, C. J. Paciorek, C. Anderson-Bergman, D. Temple Lang, and R. Bodik (2017). [Programming With Models: Writing Statistical Algorithms for General Model Structures With NIMBLE](#). *Journal of Computational and Graphical Statistics* **26** (2): 403–13.

Live demo



What you see is not what you get: Hidden Markov models and capture-recapture data

The team

last updated: 2021-05-18

Back to our survival example

- We have z survivors out of n released animals with winter survival probability ϕ
- Our model so far:

$$\begin{aligned} z &\sim \text{Binomial}(n, \phi) && [\text{likelihood}] \\ \phi &\sim \text{Beta}(1, 1) && [\text{prior for } \phi] \end{aligned}$$

- This is also:

$$\begin{aligned} z_i &\sim \text{Bernoulli}(\phi), \quad i = 1, \dots, N && [\text{likelihood}] \\ \phi &\sim \text{Beta}(1, 1) && [\text{prior for } \phi] \end{aligned}$$

- What if we had several winters? Say $T = 5$ winters.

Longitudinal data

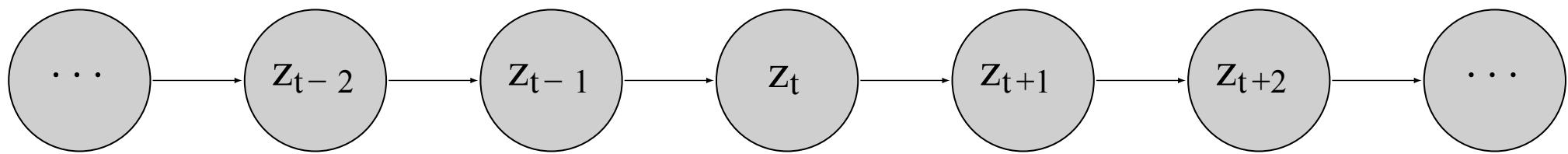
- $z_{i,t} = 1$ if individual i alive at winter t , and $z_{i,t} = 2$ if dead.

id	winter 1	winter 2	winter 3	winter 4	winter 5
1	1	2	2	2	2
2	1	1	1	2	2
3	1	1	2	2	2
4	1	1	2	2	2
5	1	2	2	2	2
6	1	2	2	2	2
7	1	1	1	1	1
8	1	1	1	1	2

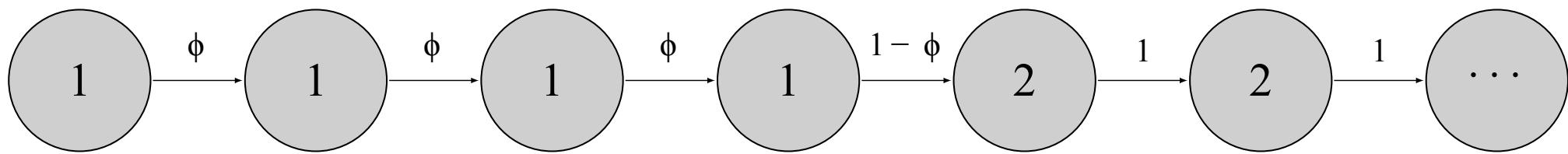
A model for longitudinal survival data

- A model relies on assumptions.
- The state of an animal at a given winter, alive or dead, is only dependent on its state the winter before.
- The future depends only on the present, not the past: Markov process.
- If an animal is alive in a given winter, the probability it survives to the next winter is ϕ .
- The probability it dies is $1 - \phi$.
- If an animal is dead a winter, it remains dead, unless you believe in zombies.

Markov process



Markov process



Transition matrix

- The core of the Markov process is made of the transition probabilities.
- For example, the probability of transitioning from state alive at $t - 1$ to state alive at t is $\Pr(z_t = 1 | z_{t-1} = 1) = \gamma_{1,1}$. It is the survival probability ϕ .
- The probability of dying over the interval $(t - 1, t)$ is
 $\Pr(z_t = 2 | z_{t-1} = 1) = \gamma_{1,2} = 1 - \phi$.
- Now if an animal is dead at $t - 1$, then $\Pr(z_t = 1 | z_{t-1} = 2) = 0$ and
 $\Pr(z_t = 2 | z_{t-1} = 2) = 1$.
- These probabilities can be packed in a transition matrix Γ :

$$\Gamma = \begin{pmatrix} \gamma_{1,1} & \gamma_{1,2} \\ \gamma_{2,1} & \gamma_{2,2} \end{pmatrix} = \begin{pmatrix} \phi & 1 - \phi \\ 0 & 1 \end{pmatrix}$$

Transition matrix:

$$\boldsymbol{\Gamma} = \begin{pmatrix} z_t = A & z_t = D \\ \hline \phi & 1 - \phi \\ 0 & 1 \end{pmatrix} \begin{matrix} z_{t-1} = A \\ z_{t-1} = D \end{matrix}$$

Initial states

- A Markov process has to start somewhere.
- We need the probabilities of initial states, i.e. states at $t = 1$.
- We will use $\delta = (\Pr(z_1 = 1), \Pr(z_1 = 2))$.
- Here we assume that all animals are alive at first winter, i.e. $\Pr(z_1 = 1) = 1$ and $\Pr(z_1 = 2) = 0$.

Likelihood

$$\Pr(\mathbf{z}) = \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1)$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\ &= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1)\end{aligned}$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1}, z_{T-2}, \dots, z_1)\end{aligned}$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}, \dots, z_1) \Pr(z_{T-2}, \dots, z_1)\end{aligned}$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}, \dots, z_1) \Pr(z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \Pr(z_{T-2}, \dots, z_1)\end{aligned}$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}, \dots, z_1) \Pr(z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \Pr(z_{T-2}, \dots, z_1) \\&= \dots\end{aligned}$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}, \dots, z_1) \Pr(z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \Pr(z_{T-2}, \dots, z_1) \\&= \dots \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \dots \Pr(z_2 | z_1) \Pr(z_1)\end{aligned}$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}, \dots, z_1) \Pr(z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \Pr(z_{T-2}, \dots, z_1) \\&= \dots \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \dots \Pr(z_2 | z_1) \Pr(z_1) \\&= \Pr(z_1) \prod_{t=2}^T \Pr(z_t | z_{t-1})\end{aligned}$$

Likelihood

$$\begin{aligned}\Pr(\mathbf{z}) &= \Pr(z_T, z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}, z_{T-2}, \dots, z_1) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1}, z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}, \dots, z_1) \Pr(z_{T-2}, \dots, z_1) \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \Pr(z_{T-2}, \dots, z_1) \\&= \dots \\&= \Pr(z_T | z_{T-1}) \Pr(z_{T-1} | z_{T-2}) \dots \Pr(z_2 | z_1) \Pr(z_1) \\&= \Pr(z_1) \prod_{t=2}^T \Pr(z_t | z_{t-1}) \\&= \Pr(z_1) \prod_{t=2}^T \gamma_{z_{t-1}, z_t}\end{aligned}$$

Example

- Let's assume an animal is alive, alive then dies.
- We have $\mathbf{z} = (1, 1, 2)$. What is the contribution of this animal to the likelihood?

$$\begin{aligned}\Pr(\mathbf{z} = (1, 1, 2)) &= \Pr(z_1 = 1) \gamma_{z_1=1, z_2=1} \gamma_{z_2=1, z_3=2} \\ &= \phi (1 - \phi).\end{aligned}$$

- Remember:

$$\boldsymbol{\Gamma} = \begin{pmatrix} \gamma_{1,1} & \gamma_{1,2} \\ \gamma_{2,1} & \gamma_{2,2} \end{pmatrix} = \begin{pmatrix} \phi & 1 - \phi \\ 0 & 1 \end{pmatrix}$$

Our model

$$z_1 \sim \text{Multinomial}(1, \delta) \quad [\text{likelihood, } t = 1]$$

Our model

$$z_1 \sim \text{Multinomial}(1, \delta) \quad [\text{likelihood}, t = 1]$$

$$\phi \sim \text{Beta}(1, 1) \quad [\text{prior for } \phi]$$

Our model

$$\begin{array}{ll} z_1 \sim \text{Multinomial}(1, \delta) & [\text{likelihood}, t = 1] \\ z_t | z_{t-1} \sim \text{Multinomial}(1, \gamma_{z_{t-1}, z_t}) & [\text{likelihood}, t > 1] \\ \phi \sim \text{Beta}(1, 1) & [\text{prior for } \phi] \end{array}$$

Our model

$$\begin{aligned} z_1 &\sim \text{Multinomial}(1, \delta) && [\text{likelihood, } t = 1] \\ z_t | z_{t-1} &\sim \text{Multinomial}(1, \gamma_{z_{t-1}, z_t}) && [\text{likelihood, } t > 1] \\ \phi &\sim \text{Beta}(1, 1) && [\text{prior for } \phi] \end{aligned}$$

$$\boldsymbol{\Gamma} = \begin{pmatrix} \phi & 1 - \phi \\ 0 & 1 \end{pmatrix}$$

$$\gamma_{z_{t-1}=1, z_t} = (\phi, 1 - \phi)$$

Our model

$$\begin{array}{ll} z_1 \sim \text{Multinomial}(1, \delta) & [\text{likelihood}, t = 1] \\ z_t | z_{t-1} \sim \text{Multinomial}(1, \gamma_{z_{t-1}, z_t}) & [\text{likelihood}, t > 1] \\ \phi \sim \text{Beta}(1, 1) & [\text{prior for } \phi] \end{array}$$

$$\boldsymbol{\Gamma} = \begin{pmatrix} \phi & 1 - \phi \\ 0 & 1 \end{pmatrix}$$

$$\gamma_{z_{t-1}=2, z_t} = (0, 1)$$

Nimble implementation

- In Nimble, we will use the categorical distribution `dcat()`.
- The categorical distribution is a multinomial distribution with a single draw.

```
nimble::rcat(n = 20, prob = c(0.1, 0.3, 0.6))
```

```
## [1] 1 3 2 3 3 3 2 2 3 1 3 3 2 3 3 3 2 2 3 3
```

```
nimble::rcat(n = 20, prob = c(0.1, 0.1, 0.4, 0.2, 0.2))
```

```
## [1] 3 5 2 3 5 2 2 3 1 2 5 3 3 1 4 1 1 3 3 3
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi    # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  }})
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi   # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0         # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1         # Pr(dead t -> dead t+1)
  delta[1] <- 1           # Pr(alive t = 1) = 1
  delta[2] <- 0           # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  }})
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi   # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  }})
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi    # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  }})
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi    # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  }
})
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi    # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  }})
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi    # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  }})
```

Nimble code

```
markov.survival <- nimbleCode({
  phi ~ dunif(0, 1) # prior
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi    # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,1] ~ dcat(delta[1:2])
    for (j in 2:T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    }
  })
})
```

Note

- Vector δ is used as a placeholder for more complex models to come in Class 7.
- Here, you could write `z[i, 1] <- 1.`

Nimble awesomness

You should be able to define vectors and matrices like you do in R.

```
markov.survival <- nimbleCode({  
  phi ~ dunif(0, 1) # prior  
  gamma[1:2,1:2] <- matrix( c(phi, 0, 1 - phi, 1), nrow = 2)  
  delta[1:2] <- c(1, 0)  
  # likelihood  
  for (i in 1:N){  
    z[i,1] ~ dcat(delta[1:2])  
    for (j in 2:T){  
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])  
    }  
  } })
```

Converting to Nimble from Jags, OpenBUGS or WinBUGS

- Main difference is that Nimble does not guess.
- We need to specify dimensions of vectors and matrices.
- You cannot write `x[]` or `x[i,]`. Just provide index ranges `x[1:n]` or `x[i,1:m]`.
- More tips [here](#).

Constants and data

```
my.constants <- list(N = 57, T = 5)
my.constants
```

```
## $N
## [1] 57
##
## $T
## [1] 5
```

```
my.data <- list(z = z)
```

Initial values

```
initial.values <- function() list(phi = runif(1,0,1))  
initial.values()
```

```
## $phi  
## [1] 0.4695068
```

Parameters to monitor

```
parameters.to.save <- c("phi")
parameters.to.save
## [1] "phi"
```

MCMC details

```
n.iter <- 5000  
n.burnin <- 1000  
n.chains <- 2
```

Run Nimble

```
mcmc.output <- nimbleMCMC(code = markov.survival,  
                           constants = my.constants,  
                           data = my.data,  
                           inits = initial.values,  
                           monitors = parameters.to.save,  
                           niter = n.iter,  
                           nburnin = n.burnin,  
                           nchains = n.chains)
```

Posterior distribution of survival

```
library(MCMCvis)
MCMCsummary(mcmc.output, round = 2)
```

```
##      mean     sd 2.5% 50% 97.5% Rhat n.eff
## phi 0.76 0.03  0.7 0.77  0.83     1 1848
```

- Posterior mean and median are close to 0.8.
- Cool! The data was simulated, with (true) survival $\phi = 0.8$.

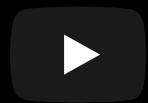
Live demo



Unfortunately, this is the data we wish we had.

In real life

- Animals cannot be monitored exhaustively, like humans in a medical trial.
- Animals are captured, marked or identified then released alive.
- Then, these animals may be detected again, or go undetected – **capture-recapture**
- Whenever animals go undetected, it might be that they were alive but missed, or because they were dead and therefore could not be detected – **imperfect detection**.



In real life

- Animals cannot be monitored exhaustively, like humans in a medical trial.
- Animals are captured, marked or identified then released alive.
- Then, these animals may be detected again, or go undetected – **capture-recapture**
- Whenever animals go undetected, it might be that they were alive but missed, or because they were dead and therefore could not be detected – **imperfect detection**.
- The Markov process for survival is only partially observed – **hidden Markov models**.



The truth is in z

id	winter 1	winter 2	winter 3	winter 4	winter 5
1	1	2	2	2	2
2	1	1	1	2	2
3	1	1	2	2	2
4	1	1	2	2	2
5	1	2	2	2	2
6	1	2	2	2	2

- Unfortunately, we have only partial access to z .
- We do observe y the detections and non-detections.
- How are z and y connected?

Dead animals go undetected

- When an animal is dead i.e. $z = 2$, it cannot be detected, therefore $y = 0$.

id	winter 1	winter 2	winter 3	winter 4	winter 5
1	1	0	0	0	0
2	1	1	1	0	0
3	1	1	0	0	0
4	1	1	0	0	0
5	1	0	0	0	0
6	1	0	0	0	0

Alive animals may be detected or not

- If animal is alive $z = 1$, it is detected $y = 1$ w/ prob p or not $y = 0$ w/ prob $1 - p$.
- Before **first** detection, we know nothing, and we proceed conditional on it.

id	winter 1	winter 2	winter 3	winter 4	winter 5
1	NA	1	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	1	1	0	1
5	1	1	0	0	0
6	1	1	1	1	0

- This table y is what we observe in real life.

Observation matrix

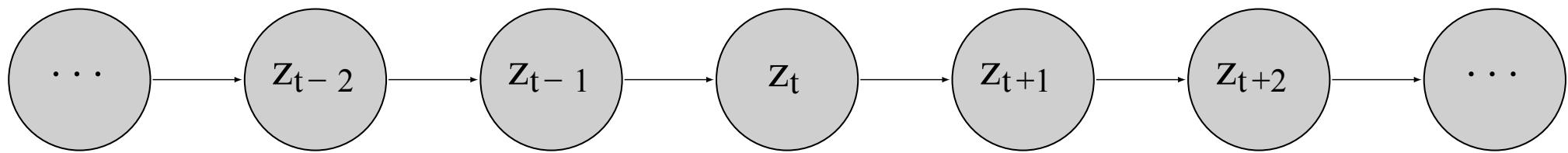
- The observation probabilities can be packed in an observation matrix Ω .
- In rows: the states alive $z = 1$ and dead $z = 2$.
- In columns: the observations non-detected $y = 1$ and detected $y = 2$ (previously coded 0 and 1 respectively).

$$\Omega = \begin{pmatrix} \omega_{1,1} & \omega_{1,2} \\ \omega_{2,1} & \omega_{2,2} \end{pmatrix} = \begin{pmatrix} 1-p & p \\ 1 & 0 \end{pmatrix}$$

Observation matrix:

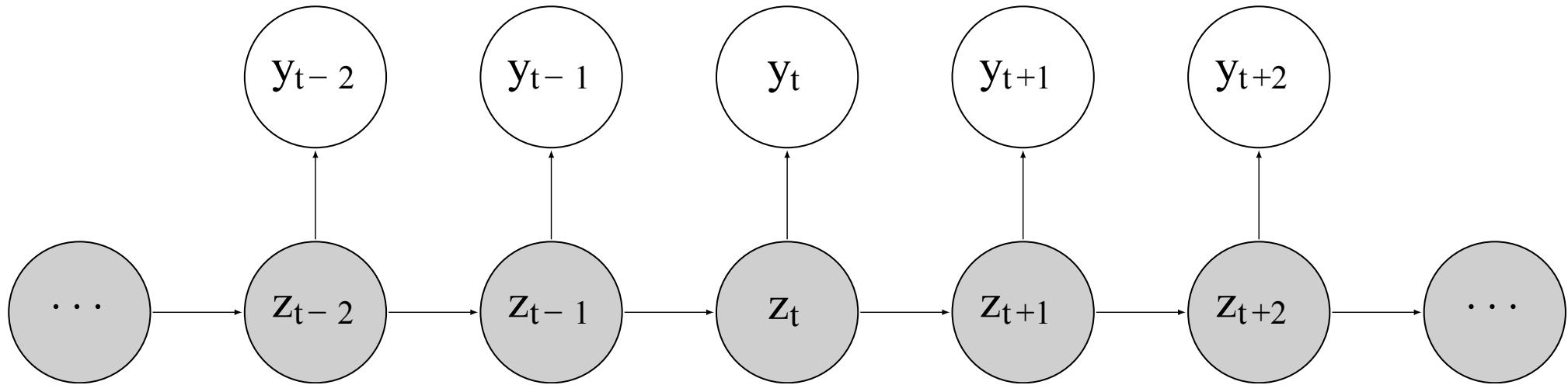
$$\Omega = \left(\begin{array}{cc} \overbrace{\quad}^{y_t = 1} & \overbrace{\quad}^{y_t = 2} \\ \begin{array}{c} 1-p \\ 1 \end{array} & \begin{array}{c} p \\ 0 \end{array} \end{array} \right) \begin{array}{l} z_t = A \\ z_t = D \end{array}$$

Markov model



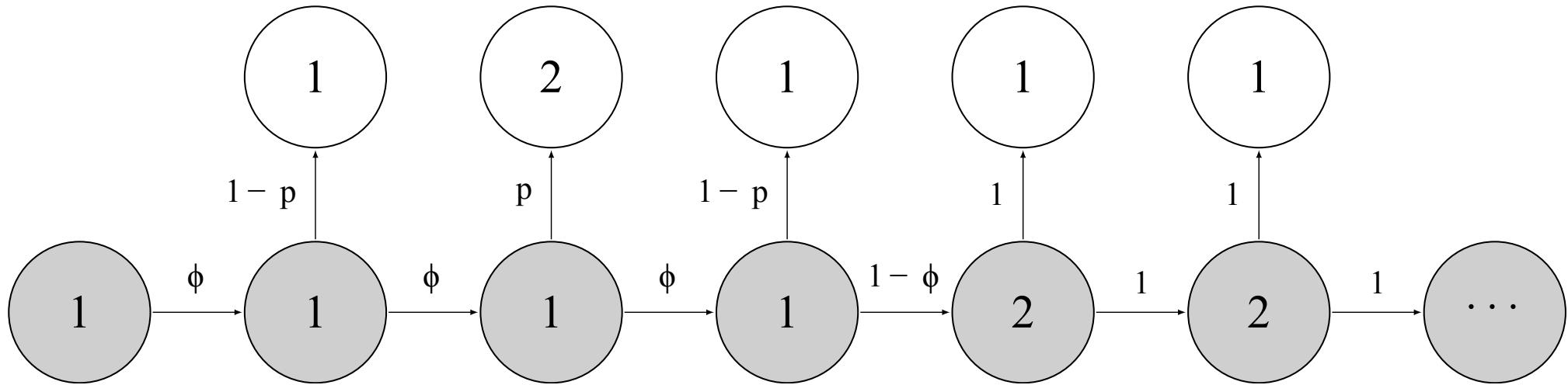
- States z are in gray.

Hidden Markov model



- States z are in gray.
- Observations y are in white.

Hidden Markov model for survival



- For states (in gray), $z = 1$ is alive, $z = 2$ is dead.
- For observations (in white), $y = 1$ is non-detected, $y = 2$ is detected

HMM likelihood

- Using the formula of total probability, then the likelihood of a Markov chain:

$$\begin{aligned}\Pr(\mathbf{y}) &= \Pr(y_1, y_2, \dots, y_T) \\ &= \sum_{z_1} \cdots \sum_{z_T} \Pr(y_1, y_2, \dots, y_T | z_1, z_2, \dots, z_T) \Pr(z_1, z_2, \dots, z_T) \\ &= \sum_{z_1} \cdots \sum_{z_T} \left(\prod_{t=1}^T \omega_{z_t, y_t} \right) \left(\Pr(z_1) \prod_{t=2}^T \gamma_{z_{t-1}, z_t} \right)\end{aligned}$$

- It has a matrix formulation:

$$\Pr(\mathbf{y}) = \delta \boldsymbol{\Omega} \boldsymbol{\Gamma} \cdots \boldsymbol{\Omega} \boldsymbol{\Gamma} \boldsymbol{\Omega} \mathbf{1}$$

Example

- Let assume an animal is detected, then missed.
- We have $\mathbf{y} = (2, 1)$. What is the contribution of this animal to the likelihood?

$$\Pr(\mathbf{y} = (2, 1)) = \sum_{z_1=1}^2 \sum_{z_2=1}^2 w_{z_1, y_1=2} w_{z_2, y_2=1} \Pr(z_1) \gamma_{z_1, z_2}$$

Example

- Let assume an animal is detected, then missed.
- We have $\mathbf{y} = (2, 1)$. What is the contribution of this animal to the likelihood?

$$\begin{aligned}\Pr(\mathbf{y} = (2, 1)) &= \sum_{z_1=1}^2 \sum_{z_2=1}^2 w_{z_1, y_1=2} w_{z_2, y_2=1} \Pr(z_1) \gamma_{z_1, z_2} \\ &= \sum_{z_1=1}^2 (w_{z_1, y_1=2} w_{z_2=1, y_2=1} \Pr(z_1) \gamma_{z_1, z_2=1} + w_{z_1, y_1=2} w_{z_2=2, y_2=1} \Pr(z_1) \gamma_{z_1, z_2=2})\end{aligned}$$

Example

- Let assume an animal is detected, then missed.
- We have $\mathbf{y} = (2, 1)$. What is the contribution of this animal to the likelihood?

$$\begin{aligned}\Pr(\mathbf{y} = (2, 1)) &= \sum_{z_1=1}^2 \sum_{z_2=1}^2 w_{z_1, y_1=2} w_{z_2, y_2=1} \Pr(z_1) \gamma_{z_1, z_2} \\ &= \sum_{z_1=1}^2 (w_{z_1, y_1=2} w_{z_2=1, y_2=1} \Pr(z_1) \gamma_{z_1, z_2=1} + w_{z_1, y_1=2} w_{z_2=2, y_2=1} \Pr(z_1) \gamma_{z_1, z_2=2}) \\ &= w_{z_1=1, y_1=2} w_{z_2=1, y_2=1} \delta_1 \gamma_{z_1=1, z_2=1} + w_{z_1=1, y_1=2} w_{z_2=2, y_2=1} \delta_1 \gamma_{z_1=1, z_2=2}\end{aligned}$$

Note: $\Pr(z_1 = 1) = \delta_1 = 1$ and $\Pr(z_1 = 2) = 0$.

Example

- Let assume an animal is detected, then missed.
- We have $\mathbf{y} = (2, 1)$. What is the contribution of this animal to the likelihood?

$$\begin{aligned}\Pr(\mathbf{y} = (2, 1)) &= \sum_{z_1=1}^2 \sum_{z_2=1}^2 w_{z_1, y_1=2} w_{z_2, y_2=1} \Pr(z_1) \gamma_{z_1, z_2} \\ &= \sum_{z_1=1}^2 (w_{z_1, y_1=2} w_{z_2=1, y_2=1} \Pr(z_1) \gamma_{z_1, z_2=1} + w_{z_1, y_1=2} w_{z_2=2, y_2=1} \Pr(z_1) \gamma_{z_1, z_2=2}) \\ &= w_{z_1=1, y_1=2} w_{z_2=1, y_2=1} \delta_1 \gamma_{z_1=1, z_2=1} + w_{z_1=1, y_1=2} w_{z_2=2, y_2=1} \delta_1 \gamma_{z_1=1, z_2=2} \\ &= (1 - p)\phi + (1 - \phi)\end{aligned}$$

Note: $w_{z_1=1, y_1=2} = \Pr(y_1 = 2 | z_1 = 1) = 1$ because we condition on first capture.

Estimating the latent states z or not?

- In previous example, we got rid of the states, so that likelihood was a function of ϕ and p only. This is the function we would maximize in a Frequentist approach.
- The Bayesian approach with MCMC methods allows treating the latent states as if they were parameters, and to be estimated as such.
- Inferring the latent states z can be useful to estimate prevalence, e.g. in animal epidemiology with prevalence of a disease, in evolutionary ecology with sex ratio or in conservation biology with prevalence of hybrids.
- Estimating the latent states is costly though, and if not required, marginalisation may speed up computations. Actually, you can estimate the states afterwards (Viterbi).
- More about so-called marginalisation in Yackulic et al. (2020).
- The neat thing with Nimble is that it provides marginalised models through nimbleEcology, we'll get back to it in Class 8.

Our model

$z_{\text{first}} \sim \text{Multinomial}(1, \delta)$	[likelihood]
$z_t z_{t-1} \sim \text{Multinomial}(1, \gamma_{z_{t-1}, z_t})$	[likelihood]
$y_t z_t \sim \text{Multinomial}(1, \omega_{z_t})$	[likelihood]
$\phi \sim \text{Beta}(1, 1)$	[prior for ϕ]
$p \sim \text{Beta}(1, 1)$	[prior for p]

Nimble implementation

Priors

```
hmm.survival <- nimbleCode({  
  phi ~ dunif(0, 1) # prior survival  
  p ~ dunif(0, 1) # prior detection  
  ...
```

HMM ingredients

```
...
# parameters
gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
gamma[1,2] <- 1 - phi   # Pr(alive t -> dead t+1)
gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
delta[1] <- 1            # Pr(alive t = 1) = 1
delta[2] <- 0            # Pr(dead t = 1) = 0
omega[1,1] <- 1 - p     # Pr(alive t -> non-detected t)
omega[1,2] <- p          # Pr(alive t -> detected t)
omega[2,1] <- 1          # Pr(dead t -> non-detected t)
omega[2,2] <- 0          # Pr(dead t -> detected t)
...
```

Likelihood

```
...
# likelihood
for (i in 1:N){
  z[i,first[i]] ~ dcat(delta[1:2])
  for (j in (first[i]+1):T){
    z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
    y[i,j] ~ dcat(omega[z[i,j], 1:2])
  }
}
})
```

Constants

```
first <- apply(y, 1, function(x) min(which(x !=0)))
my.constants <- list(N = nrow(y), T = 5, first = first)
my.constants
```

```
## $N
## [1] 44
##
## $T
## [1] 5
##
## $first
## [1] 2 1 1 1 1 1 4 1 1 2 1 1 1 2 1 1 1 1 1 3 1 1 1 2 1 3 1 2 3 3 1 4
## [39] 3 1 2 1 1 1
```

Data

- The data are made of 0s for non-detections and 1s for detections.
- To use the categorical distribution, we need to code 1, 2, etc. Value 0 is not accepted.
- Add 1 to get the correct format $y = 1$ for non-detection and $y = 2$ for detection.

```
my.data <- list(y = y + 1)
```

Initial values

```
zinit <- y + 1 # non-detection -> alive  
zinit[zinit == 2] <- 1 # dead -> alive  
initial.values <- function() list(phi = runif(1,0,1),  
                                p = runif(1,0,1),  
                                z = zinit)
```

Parameters to monitor

```
parameters.to.save <- c("phi", "p")
parameters.to.save
## [1] "phi" "p"
```

MCMC details

```
n.iter <- 5000  
n.burnin <- 1000  
n.chains <- 2
```

Run Nimble

```
mcmc.output <- nimbleMCMC(code = hmm.survival,  
                           constants = my.constants,  
                           data = my.data,  
                           inits = initial.values,  
                           monitors = parameters.to.save,  
                           niter = n.iter,  
                           nburnin = n.burnin,  
                           nchains = n.chains)
```

Posterior distribution of survival

```
library(MCMCvis)
MCMCsummary(mcmc.output, round = 2)
```

```
##      mean     sd 2.5% 50% 97.5% Rhat n.eff
## p    0.57  0.06 0.45 0.57  0.69 1.00    503
## phi 0.86  0.04 0.78 0.86  0.93 1.01    499
```

The data is simulated, with true survival $\phi = 0.8$ and detection $p = 0.6$.

Further reading

- Zucchini, MacDonald and Langrock (2016) **Hidden Markov Models for Time Series: An Introduction Using R** (2nd ed). Chapman and Hall/CRC.
- McClintock, B.T., Langrock, R., Gimenez, O., Cam, E., Borchers, D.L., Glennie, R. and Patterson, T.A. (2020), **Uncovering ecological state dynamics with hidden Markov models**. Ecology Letters, 23: 1878-1903.
- Yackulic, C. B. Dodrill, M., Dzul, M., Sanderlin, J. S., and Reid, J. A.. (2020). **A need for speed in Bayesian population models: a practical guide to marginalizing and recovering discrete latent states**. Ecological Applications 30:e02112.
- L. R. Rabiner (1989). **A tutorial on hidden Markov models and selected applications in speech recognition**. Proceedings of the IEEE, 77:257-286.

Live demo



Dead or alive: Survival estimation

The team

last updated: 2021-05-18

MODELING SURVIVAL AND TESTING BIOLOGICAL HYPOTHESES USING MARKED ANIMALS: A UNIFIED APPROACH WITH CASE STUDIES¹

JEAN-DOMINIQUE LEBRETON

CEFE/CNRS, BP 5051, 34033 Montpellier Cedex, France

KENNETH P. BURNHAM

*Colorado Cooperative Fish and Wildlife Research Unit, U.S. Fish and Wildlife Service,
201 Wagar Building, Fort Collins, Colorado 80523 USA*

JEAN CLOBERT

Laboratoire d'Ecologie, Ecole Normale Supérieure, 46 rue d'Ulm 75231, Paris Cedex 05 France

DAVID R. ANDERSON

*Colorado Cooperative Fish and Wildlife Research Unit, U.S. Fish and Wildlife Service,
201 Wagar Building, Fort Collins, Colorado 80523 USA*

History of the Cormack-Jolly-Seber (CJS) model

S.T. Buckland (2016). A Conversation with Richard M. Cormack. Statistical Science 31: 142-150.

Buckland: George Jolly was a colleague of yours in the 1960s. Could you describe your interactions with him?

Cormack: George was in the ARC Unit of Statistics in the same corridor as I was. His main job was designing and conducting agricultural surveys in Scotland. There wasn't a practice of giving seminars in the department to talk to colleagues about what one was doing, and David Finney's appointees had been chosen to cover all the varied areas of statistics rather than build a research group in a particular area. So despite the fact that I met George every day at coffee, and, indeed, we caused David a lot of angst as, on many mornings, we played kriegspiel (a version of chess where you don't see the other person's board and a referee judges—very good for developing inference), we never mentioned work and mark-recapture. I don't remember George noticing my Biometrika paper in 1964 (Cormack, 1964), or indeed the practical paper in British Birds in 1963 (Dunnet, Anderson and Cormack, 1963). It was completely unknown to the two of us that we were working in the same area.

Buckland: What interactions did you have with George Seber?

Cormack: Before the 1965 papers (Jolly, 1965; Seber, 1965), George Seber and I had no contact whatsoever. After the papers, yes, we did. We got into deep communication after the first papers, and he was all for sending me drafts of everything he did. He produced stuff at a colossal rate and his encyclopaedic knowledge was unbelievable. I'm not sure he ever actually worked closely with biologists, but, when he was writing his book, he asked if I would comment on the draft chapters on the bits I knew about. But you have to realise that communication between opposite corners of the world took time. At one point, I received a plaintive handwritten letter saying "The University has cut down on postage and I'm not allowed to post the draft chapter airmail and you will have to wait for it to come by surface mail from New Zealand." By the time it arrived, I already had another airmail letter from him saying, "I'm sorry you haven't been able to comment on the chapter—I've had to submit it!" To some extent, the opposite is true now: response is too quick.

REVIEW

A review of Bayesian state-space^{*} modelling of capture–recapture–recovery data

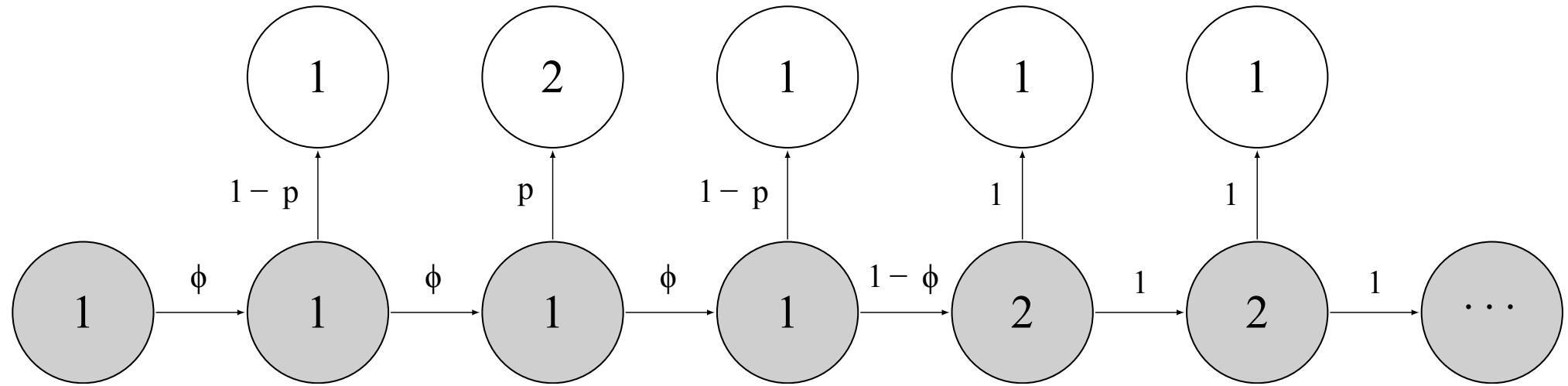
Ruth King*

School of Mathematics and Statistics and Centre for Research into Ecological and Environmental Modelling, University of St Andrews, St Andrews, Fife KY16 9LZ, UK

Traditionally, state-space models are fitted to data where there is uncertainty in the observation or measurement of the system. State-space models are partitioned into an underlying system process describing the transitions of the true states of the system over time and the observation process linking the observations of the system to the true states. Open population capture–recapture–recovery data can be modelled in this framework by regarding the system process as the state of each individual observed within the study in terms of being alive or dead, and the observation process the recapture and/or recovery process. The traditional observation error of a state-space model is incorporated via the recapture/recovery probabilities being less than unity. The models can be fitted using a Bayesian data augmentation approach and in standard BUGS packages. Applying this state-space framework to such data permits additional complexities including individual heterogeneity to be fitted to the data at very little additional programming effort. We consider the efficiency of the state-space model fitting approach by considering a random effects model for capture–recapture data relating to dippers and compare different Bayesian model-fitting algorithms within WinBUGS.

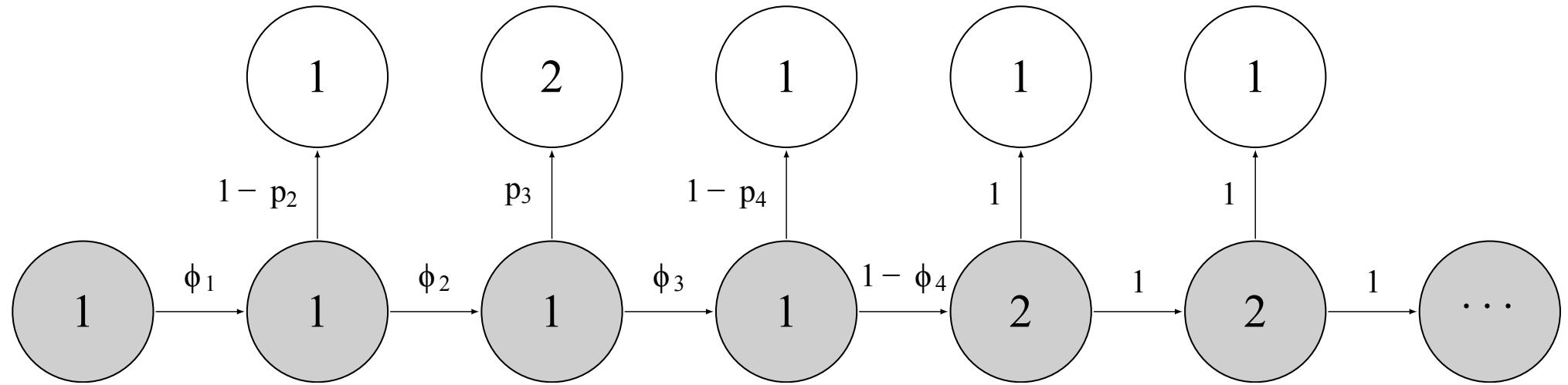
* State-space models may include continuous latent states

What we've seen so far



- For states (in gray), $z = 1$ is alive, $z = 2$ is dead.
- For observations (in white), $y = 1$ is non-detected, $y = 2$ is detected

In the CJS model, survival and recapture are time-varying



- Survival probability is $\phi_t = \Pr(z_{t+1} = 1 | z_t = 1)$.
- Recapture (detection) probability is $p_t = \Pr(y_t = 1 | z_t = 1)$.
- Accounts for variation in e.g. environmental conditions (survival) or sampling effort (detection).

Capture, mark and recapture



Emmanuelle Cam & Jean-Yves Monnat



Kelly Powell

Artificial marks

Capture, mark and recapture



Natural marks



Alexander Kopatz

The famous Dipper example



White-throated Dipper (*Cinclus cinclus*)



Gilbert Marzolin

294 dippers captured and recaptured between 1981 and 1987 with known sex and wing length

year_1981	year_1982	year_1983	year_1984	year_1985	year_1986	year_1987	sex
1	1	1	1	1	1	1	0 M
1	1	1	1	1	0	0	0 F
1	1	1	1	0	0	0	0 M
1	1	1	1	0	0	0	0 F
1	1	0	1	1	1	0	0 F
1	1	0	0	0	0	0	0 M
1	1	0	0	0	0	0	0 M
1	1	0	0	0	0	0	0 M

Back to Nimble.

Our model so far (ϕ, p)

```
hmm.phpip <- nimbleCode({
  phi ~ dunif(0, 1) # prior survival
  p ~ dunif(0, 1) # prior detection
  # likelihood
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi   # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  omega[1,1] <- 1 - p     # Pr(alive t -> non-detected t)
  omega[1,2] <- p          # Pr(alive t -> detected t)
  omega[2,1] <- 1          # Pr(dead t -> non-detected t)
  omega[2,2] <- 0          # Pr(dead t -> detected t)
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
      y[i,j] ~ dcat(omega[z[i,j], 1:2])
    }
  }
})
```

Our model so far (ϕ, p)

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
phi	0.56	0.03	0.52	0.56	0.62	1.00	500
p	0.89	0.03	0.83	0.89	0.94	1.13	273

The CJS model (ϕ_t, p_t)

```
hmm.phitpt <- nimbleCode({
  delta[1] <- 1           # Pr(alive t = 1) = 1
  delta[2] <- 0           # Pr(dead t = 1) = 0
  for (t in 1:(T-1)){
    phi[t] ~ dunif(0, 1) # prior survival
    gamma[1,1,t] <- phi[t]      # Pr(alive t -> alive t+1)
    gamma[1,2,t] <- 1 - phi[t]  # Pr(alive t -> dead t+1)
    gamma[2,1,t] <- 0          # Pr(dead t -> alive t+1)
    gamma[2,2,t] <- 1          # Pr(dead t -> dead t+1)
    p[t] ~ dunif(0, 1) # prior detection
    omega[1,1,t] <- 1 - p[t]    # Pr(alive t -> non-detected t)
    omega[1,2,t] <- p[t]       # Pr(alive t -> detected t)
    omega[2,1,t] <- 1          # Pr(dead t -> non-detected t)
    omega[2,2,t] <- 0          # Pr(dead t -> detected t)
  }
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, j-1])
      y[i,j] ~ dcat(omega[z[i,j], 1:2, j-1])
    }
  }
}
```

The CJS model (ϕ_t, p_t)

```
hmm.phitpt <- nimbleCode({
  delta[1] <- 1           # Pr(alive t = 1) = 1
  delta[2] <- 0           # Pr(dead t = 1) = 0
  for (t in 1:(T-1)){
    phi[t] ~ dunif(0, 1) # prior survival
    gamma[1,1,t] <- phi[t]      # Pr(alive t -> alive t+1)
    gamma[1,2,t] <- 1 - phi[t]  # Pr(alive t -> dead t+1)
    gamma[2,1,t] <- 0          # Pr(dead t -> alive t+1)
    gamma[2,2,t] <- 1          # Pr(dead t -> dead t+1)
    p[t] ~ dunif(0, 1) # prior detection
    omega[1,1,t] <- 1 - p[t]   # Pr(alive t -> non-detected t)
    omega[1,2,t] <- p[t]       # Pr(alive t -> detected t)
    omega[2,1,t] <- 1          # Pr(dead t -> non-detected t)
    omega[2,2,t] <- 0          # Pr(dead t -> detected t)
  }
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, j-1])
      y[i,j] ~ dcat(omega[z[i,j], 1:2, j-1])
    }
  }
}
```

The CJS model (ϕ_t, p_t)

```
hmm.phitpt <- nimbleCode({
  delta[1] <- 1           # Pr(alive t = 1) = 1
  delta[2] <- 0           # Pr(dead t = 1) = 0
  for (t in 1:(T-1)){
    phi[t] ~ dunif(0, 1) # prior survival
    gamma[1,1,t] <- phi[t]      # Pr(alive t -> alive t+1)
    gamma[1,2,t] <- 1 - phi[t]  # Pr(alive t -> dead t+1)
    gamma[2,1,t] <- 0          # Pr(dead t -> alive t+1)
    gamma[2,2,t] <- 1          # Pr(dead t -> dead t+1)
    p[t] ~ dunif(0, 1) # prior detection
    omega[1,1,t] <- 1 - p[t]   # Pr(alive t -> non-detected t)
    omega[1,2,t] <- p[t]       # Pr(alive t -> detected t)
    omega[2,1,t] <- 1          # Pr(dead t -> non-detected t)
    omega[2,2,t] <- 0          # Pr(dead t -> detected t)
  }
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, j-1])
      y[i,j] ~ dcat(omega[z[i,j], 1:2, j-1])
    }
  }
}
```

The CJS model (ϕ_t, p_t)

```
hmm.phitpt <- nimbleCode({
  delta[1] <- 1           # Pr(alive t = 1) = 1
  delta[2] <- 0           # Pr(dead t = 1) = 0
  for (t in 1:(T-1)){
    phi[t] ~ dunif(0, 1) # prior survival
    gamma[1,1,t] <- phi[t]      # Pr(alive t -> alive t+1)
    gamma[1,2,t] <- 1 - phi[t]  # Pr(alive t -> dead t+1)
    gamma[2,1,t] <- 0          # Pr(dead t -> alive t+1)
    gamma[2,2,t] <- 1          # Pr(dead t -> dead t+1)
    p[t] ~ dunif(0, 1) # prior detection
    omega[1,1,t] <- 1 - p[t]   # Pr(alive t -> non-detected t)
    omega[1,2,t] <- p[t]       # Pr(alive t -> detected t)
    omega[2,1,t] <- 1          # Pr(dead t -> non-detected t)
    omega[2,2,t] <- 0          # Pr(dead t -> detected t)
  }
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, j-1])
      y[i,j] ~ dcat(omega[z[i,j], 1:2, j-1])
    }
  }
})
```

The CJS model (ϕ_t, p_t)

```
hmm.phitpt <- nimbleCode({
  delta[1] <- 1           # Pr(alive t = 1) = 1
  delta[2] <- 0           # Pr(dead t = 1) = 0
  for (t in 1:(T-1)){
    phi[t] ~ dunif(0, 1) # prior survival
    gamma[1,1,t] <- phi[t]      # Pr(alive t -> alive t+1)
    gamma[1,2,t] <- 1 - phi[t]  # Pr(alive t -> dead t+1)
    gamma[2,1,t] <- 0          # Pr(dead t -> alive t+1)
    gamma[2,2,t] <- 1          # Pr(dead t -> dead t+1)
    p[t] ~ dunif(0, 1) # prior detection
    omega[1,1,t] <- 1 - p[t]   # Pr(alive t -> non-detected t)
    omega[1,2,t] <- p[t]       # Pr(alive t -> detected t)
    omega[2,1,t] <- 1          # Pr(dead t -> non-detected t)
    omega[2,2,t] <- 0          # Pr(dead t -> detected t)
  }
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, j-1])
      y[i,j] ~ dcat(omega[z[i,j], 1:2, j-1])
    }
  }
}
```

The CJS model (ϕ_t, p_t)

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
phi[1]	0.73	0.14	0.46	0.72	0.99	1.02	199
phi[2]	0.45	0.07	0.32	0.44	0.59	1.02	410
phi[3]	0.48	0.06	0.35	0.48	0.59	1.01	506
phi[4]	0.63	0.06	0.52	0.63	0.75	1.03	415
phi[5]	0.60	0.06	0.49	0.60	0.72	1.01	365
phi[6]	0.74	0.13	0.51	0.74	0.97	1.10	38
p[1]	0.66	0.14	0.38	0.67	0.89	1.01	344
p[2]	0.87	0.08	0.68	0.89	0.98	1.02	249
p[3]	0.88	0.07	0.73	0.89	0.97	1.02	307
p[4]	0.87	0.06	0.74	0.88	0.96	1.05	333
p[5]	0.90	0.05	0.77	0.91	0.98	1.01	224
p[6]	0.72	0.13	0.50	0.72	0.97	1.08	37

Time-varying survival (ϕ_t, p)

```
hmm.phitp <- nimbleCode({
  for (t in 1:(T-1)){
    phi[t] ~ dunif(0, 1) # prior survival
    gamma[1,1,t] <- phi[t]      # Pr(alive t -> alive t+1)
    gamma[1,2,t] <- 1 - phi[t]  # Pr(alive t -> dead t+1)
    gamma[2,1,t] <- 0          # Pr(dead t -> alive t+1)
    gamma[2,2,t] <- 1          # Pr(dead t -> dead t+1)
  }
  p ~ dunif(0, 1) # prior detection
  delta[1] <- 1           # Pr(alive t = 1) = 1
  delta[2] <- 0           # Pr(dead t = 1) = 0
  omega[1,1] <- 1 - p     # Pr(alive t -> non-detected t)
  omega[1,2] <- p         # Pr(alive t -> detected t)
  omega[2,1] <- 1           # Pr(dead t -> non-detected t)
  omega[2,2] <- 0           # Pr(dead t -> detected t)
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, j-1])
      y[i,j] ~ dcat(omega[z[i,j], 1:2])
    }
  }
})
```

Time-varying survival (ϕ_t, p)

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
phi[1]	0.63	0.10	0.42	0.63	0.82	1.04	564
phi[2]	0.46	0.06	0.35	0.46	0.59	1.01	629
phi[3]	0.48	0.05	0.37	0.48	0.59	1.00	610
phi[4]	0.62	0.06	0.51	0.62	0.73	1.00	553
phi[5]	0.61	0.05	0.50	0.61	0.72	1.00	568
phi[6]	0.59	0.05	0.48	0.59	0.69	1.03	463
p	0.89	0.03	0.82	0.89	0.95	1.04	211

Time-varying detection (ϕ, p_t)

```
hmm.phipt <- nimbleCode({
  phi ~ dunif(0, 1) # prior survival
  gamma[1,1] <- phi      # Pr(alive t -> alive t+1)
  gamma[1,2] <- 1 - phi    # Pr(alive t -> dead t+1)
  gamma[2,1] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2] <- 1          # Pr(dead t -> dead t+1)
  delta[1] <- 1            # Pr(alive t = 1) = 1
  delta[2] <- 0            # Pr(dead t = 1) = 0
  for (t in 1:(T-1)){
    p[t] ~ dunif(0, 1) # prior detection
    omega[1,1,t] <- 1 - p[t]    # Pr(alive t -> non-detected t)
    omega[1,2,t] <- p[t]        # Pr(alive t -> detected t)
    omega[2,1,t] <- 1            # Pr(dead t -> non-detected t)
    omega[2,2,t] <- 0            # Pr(dead t -> detected t)
  }
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
      y[i,j] ~ dcat(omega[z[i,j], 1:2, j-1])
    }
  }
})
```

Time-varying detection (ϕ, p_t)

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
phi	0.56	0.03	0.52	0.56	0.61	1.02	381
p[1]	0.75	0.12	0.48	0.77	0.93	1.03	452
p[2]	0.85	0.08	0.68	0.86	0.97	1.02	359
p[3]	0.85	0.07	0.69	0.85	0.96	1.00	316
p[4]	0.89	0.05	0.77	0.89	0.97	1.00	412
p[5]	0.91	0.04	0.82	0.92	0.98	1.00	376
p[6]	0.90	0.07	0.73	0.91	1.00	1.07	111

How to select a best model?

Model selection

- Which of the four models above is best supported by the data?
- The proportion of explained variance R^2 is problematic, because the more variables you have, the bigger R^2 is.
- The idea is to penalize models with too many parameters.

Akaike information criterion (AIC)

$$AIC = -2 \log(L(\hat{\theta}_1, \dots, \hat{\theta}_K)) + 2K$$

with L the likelihood and K the number of parameters θ_i .

Akaike information criterion (AIC)

$$\text{AIC} = -2 \log(L(\hat{\theta}_1, \dots, \hat{\theta}_K)) + 2K$$

A measure of goodness-of-fit of the model to the data: the more parameters you have, the smaller the deviance is (or the bigger the likelihood is).

Akaike information criterion (AIC)

$$\text{AIC} = -2 \log(L(\hat{\theta}_1, \dots, \hat{\theta}_K)) + 2K$$

A penalty: twice the number of parameters K

Akaike information criterion (AIC)

- AIC makes the balance between *quality of fit* and *complexity* of a model.
- Best model is the one with lowest AIC value.
- Two models are difficult to distinguish if $\Delta \text{AIC} < 2$.

Bayesian version

- Watanabe-Akaike (Widely-Applicable) Information Criteria or WAIC:

$$\text{WAIC} = -2 \sum_{i=1}^n \log E[\Pr(y_i \mid \theta)] + 2p_{\text{WAIC}}$$

- where $E[p(y_i \mid \theta)]$ is the posterior mean of the likelihood evaluated pointwise at each i th observation.
- p_{WAIC} is a penalty computed using the posterior variance of the likelihood.
- More in this video <https://www.youtube.com/watch?v=vSjL2Zc-gEQ> by R. McElreath.
- Nimble provides the conditional WAIC, where all parameters directly involved in the likelihood are considered. If you would want to calculate the marginal WAIC, integrating over latent variables, you could monitor the relevant nodes and carry out the calculations yourself based on the MCMC output.

How to compute WAIC in Nimble?

```
parameters.to.save <- c("phi", "p")
mcmc.phitpt <- nimbleMCMC(code = hmm.phitpt,
                             constants = my.constants,
                             data = my.data,
                             inits = initial.values,
                             monitors = parameters.to.save,
                             niter = n.iter,
                             nburnin = n.burnin,
                             nchains = n.chains)
```

How to compute WAIC in Nimble?

```
parameters.to.save <- c("phi", "p", "z")
mcmc.phitpt <- nimbleMCMC(code = hmm.phitpt,
                             constants = my.constants,
                             data = my.data,
                             inits = initial.values,
                             monitors = parameters.to.save,
                             niter = n.iter,
                             nburnin = n.burnin,
                             nchains = n.chains,
                             WAIC = TRUE)
```

Dipper example - continued

	model	WAIC
1	(phi,p)	265.9168
2	(phit,p)	277.5514
3	(phi,pt)	270.2175
4	(phit,pt)	308.8417

Live demo



Can we explain time variation?

Embrace heterogeneity

- Include temporal covariates, say x_t .
- $\text{logit}(\phi_t) = \beta_1 + \beta_2 x_t$.
- Let's investigate the effect of water flow on dipper survival ([Marzolin 2002](#)).

```

hmm.phiflowp <- nimbleCode({
  delta[1] <- 1           #  $Pr(alive \ t = 1) = 1$ 
  delta[2] <- 0           #  $Pr(dead \ t = 1) = 0$ 
  for (t in 1:(T-1)){
    logit(phi[t]) <- beta[1] + beta[2] * flow[t]
    gamma[1,1,t] <- phi[t]      #  $Pr(alive \ t \rightarrow alive \ t+1)$ 
    gamma[1,2,t] <- 1 - phi[t]  #  $Pr(alive \ t \rightarrow dead \ t+1)$ 
    gamma[2,1,t] <- 0          #  $Pr(dead \ t \rightarrow alive \ t+1)$ 
    gamma[2,2,t] <- 1          #  $Pr(dead \ t \rightarrow dead \ t+1)$ 
  }
  p ~ dunif(0, 1) # prior detection
  omega[1,1] <- 1 - p      #  $Pr(alive \ t \rightarrow non-detected \ t)$ 
  omega[1,2] <- p         #  $Pr(alive \ t \rightarrow detected \ t)$ 
  omega[2,1] <- 1          #  $Pr(dead \ t \rightarrow non-detected \ t)$ 
  omega[2,2] <- 0          #  $Pr(dead \ t \rightarrow detected \ t)$ 
  beta[1] ~ dnorm(0, 1.5) # prior intercept
  beta[2] ~ dnorm(0, 1.5) # prior slope
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, j-1])
      y[i,j] ~ dcat(omega[z[i,j], 1:2])
    }
  }
})

```

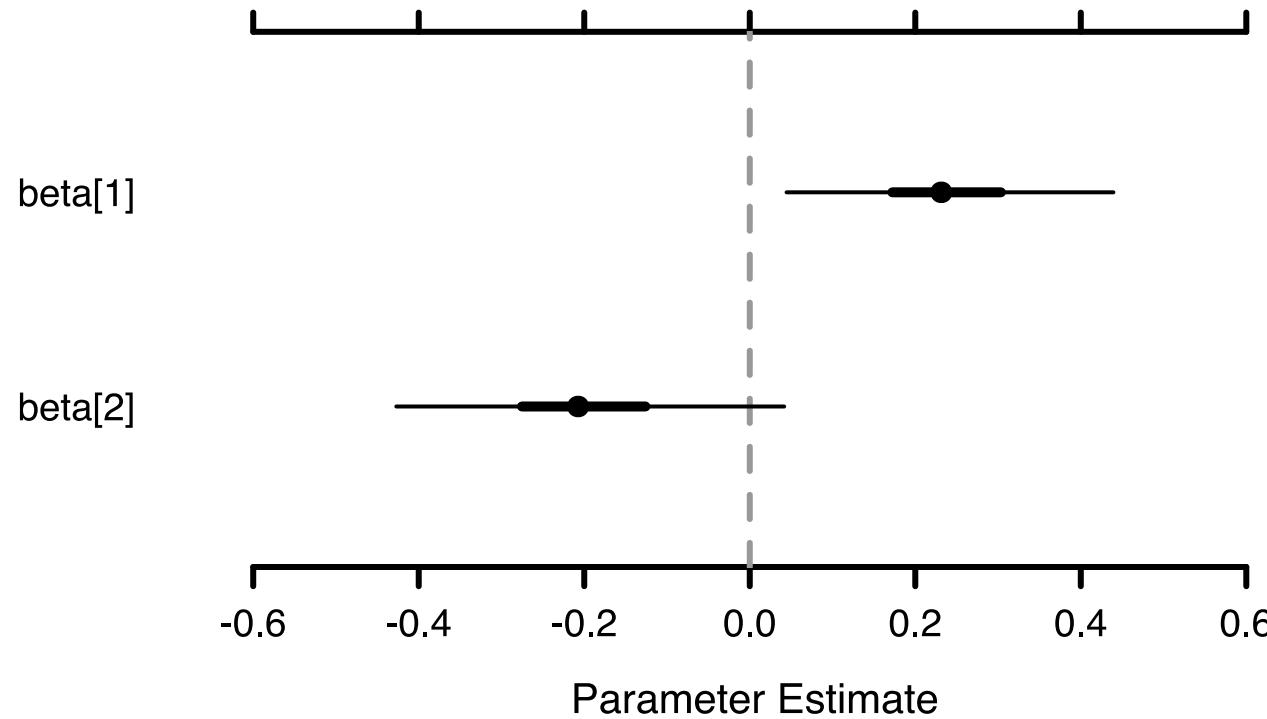
```
# water flow in L/s
water_flow <- c(443, 1114, 529, 434, 627, 466) # 1981, 1982, ..., 1987
water_flow_st <- (water_flow - mean(water_flow))/sd(water_flow)
```

```
my.constants <- list(N = nrow(y),
                      T = ncol(y),
                      first = first,
                      flow = water_flow_st)
```

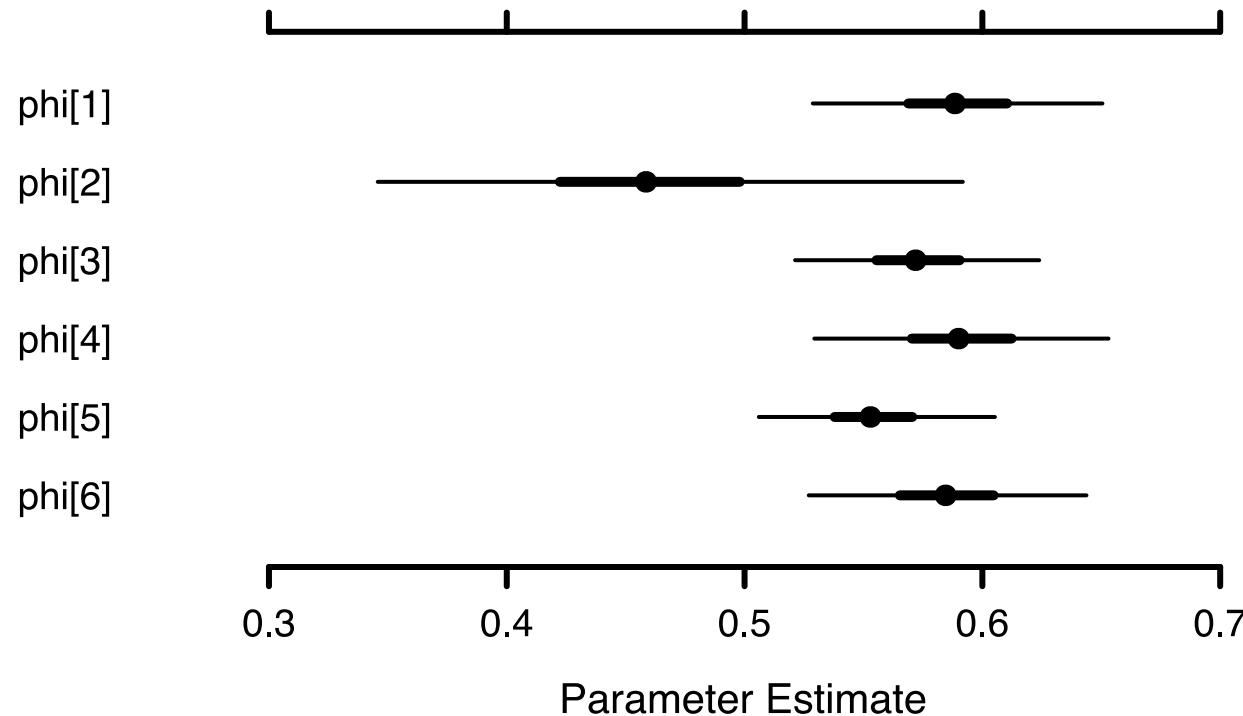
```
initial.values <- function() list(beta = rnorm(2,0,1),
                                      p = runif(1,0,1),
                                      z = zinits)
```

```
parameters.to.save <- c("beta", "p", "phi")
```

Regression intercept and slope



Time-dependent (covariate constrained) survival probability estimates



Embrace heterogeneity

- Include temporal covariates, say x_t
- $\text{logit}(\phi_t) = \beta_1 + \beta_2 x_t$
- If temporal variation not fully explained by covariates, add random effects
- $\text{logit}(\phi_t) = \beta_1 + \beta_2 x_t + \varepsilon_t, \varepsilon_t \sim N(0, \sigma^2)$

```
hmm.phiflowREp <- nimbleCode( {
  for (t in 1:(T-1)){
    logit(phi[t]) <- beta[1] + beta[2] * flow[t] + eps[t]
    eps[t] ~ dnorm(0, sd = sdeps)
    ...
  }
  sdeps ~ dunif(0, 10)
  ...
}
```

What about individual heterogeneity?

- Discrete covariate like, e.g., sex
- Continuous covariate like, e.g., mass or size

Sex and wing length in Dipper

year_1981	year_1982	year_1983	year_1984	year_1985	year_1986	year_1987	sex
1	1	1	1	1	1	1	0 M
1	1	1	1	1	0	0	0 F
1	1	1	1	0	0	0	0 M
1	1	1	1	0	0	0	0 F
1	1	0	1	1	1	0	0 F
1	1	0	0	0	0	0	0 M
1	1	0	0	0	0	0	0 M
1	1	0	0	0	0	0	0 M

Sex effect

- Let's use a covariate sex that takes value 0 if male, and 1 if female
- And write $\text{logit}(\phi_i) = \beta_1 + \beta_2 \text{ sex}_i$ for bird i
- Then male survival is

$$\text{logit}(\phi_i) = \beta_1$$

- And female survival is

$$\text{logit}(\phi_i) = \beta_1 + \beta_2$$

Nimble implementation with sex as a covariate

```
hmm.phisexp <- nimbleCode({  
  ...  
  for (i in 1:N){  
    logit(phi[i]) <- beta[1] + beta[2] * sex[i]  
    gamma[1,1,i] <- phi[i]      # Pr(alive t -> alive t+1)  
    gamma[1,2,i] <- 1 - phi[i]  # Pr(alive t -> dead t+1)  
    gamma[2,1,i] <- 0          # Pr(dead t -> alive t+1)  
    gamma[2,2,i] <- 1          # Pr(dead t -> dead t+1)  
  }  
  beta[1] ~ dnorm(mean = 0, sd = 1.5)  
  beta[2] ~ dnorm(mean = 0, sd = 1.5)  
  phi_male <- 1/(1+exp(-beta[1]))  
  phi_female <- 1/(1+exp(-(beta[1]+beta[2])))  
  ...  
  # likelihood  
  for (i in 1:N){  
    z[i,first[i]] ~ dcat(delta[1:2])  
    for (j in (first[i]+1):T){  
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, i])  
      y[i,j] ~ dcat(omega[z[i,j], 1:2])  
    }  
  }  
})
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
beta[1]	0.29	0.14	0.01	0.29	0.57	1.01	237
beta[2]	-0.09	0.19	-0.47	-0.10	0.29	1.01	241
p	0.90	0.03	0.83	0.90	0.95	1.02	253
phi_female	0.55	0.04	0.48	0.55	0.62	1.02	698
phi_male	0.57	0.03	0.50	0.57	0.64	1.01	237

Nimble implementation with nested indexing

- Let's use a covariate sex that contains 1s and 2s, indicating the sex of each individual: 1 if male, and 2 if female

```
...
for (i in 1:N){
  phi[i] <- beta[sex[i]]
  gamma[1,1,i] <- phi[i]      # Pr(alive t -> alive t+1)
  gamma[1,2,i] <- 1 - phi[i]   # Pr(alive t -> dead t+1)
  gamma[2,1,i] <- 0            # Pr(dead t -> alive t+1)
  gamma[2,2,i] <- 1            # Pr(dead t -> dead t+1)
}
beta[1] ~ dunif(0,1) # male survival
beta[2] ~ dunif(0,1) # female survival
...
```

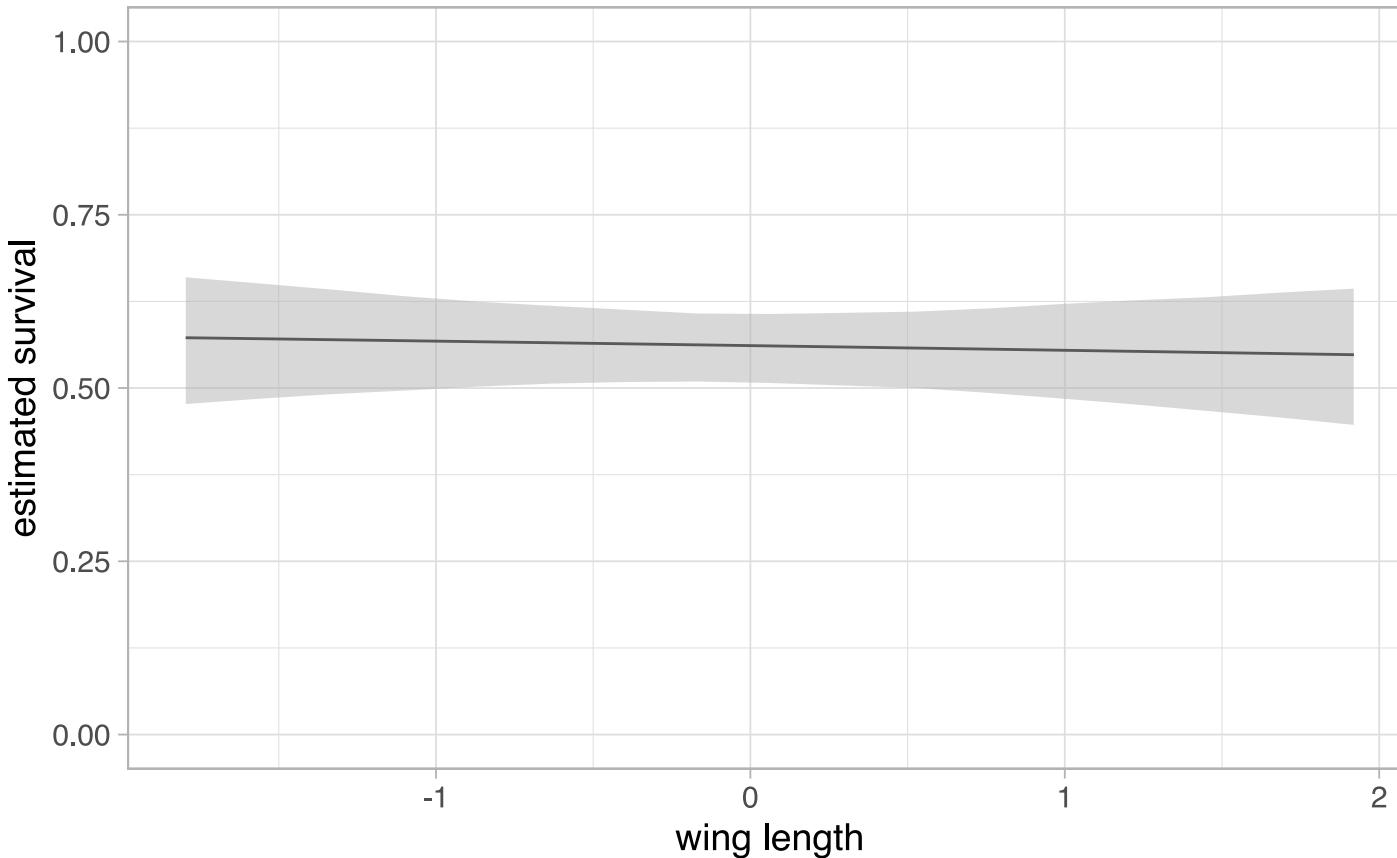
- E.g. for individual $i = 2$, `beta[sex[i]]` gives `beta[sex[2]]` which will be `beta[1]` or `beta[2]` depending on whether `sex[2]` is 1 or 2.

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
beta[1]	0.57	0.03	0.50	0.57	0.63	1.00	616
beta[2]	0.55	0.03	0.48	0.55	0.62	1.02	657
p	0.90	0.03	0.83	0.90	0.95	1.10	229

What about wing length?

```
...
for (i in 1:N){
  logit(phi[i]) <- beta[1] + beta[2] * winglength[i]
  gamma[1,1,i] <- phi[i]      # Pr(alive t -> alive t+1)
  gamma[1,2,i] <- 1 - phi[i]  # Pr(alive t -> dead t+1)
  gamma[2,1,i] <- 0          # Pr(dead t -> alive t+1)
  gamma[2,2,i] <- 1          # Pr(dead t -> dead t+1)
}
beta[1] ~ dnorm(mean = 0, sd = 1.5) # intercept
beta[2] ~ dnorm(mean = 0, sd = 1.5) # slope
...
```

Wing length



- You may test an effect of both sex and wing length, see exercise in Worksheets.

What if covariates vary with individual and time?

- Think of age for example (see exercises in Worksheets); covariate or nested indexing works fine.
- Now, think of body size across life.
- Problem is we cannot record size when animal is non-detected.
- Discretize in small, medium and large and treat as a state — more later.
- Assume a model for covariate and fill in missing values (imputation).

Live demo



Why Bayes?

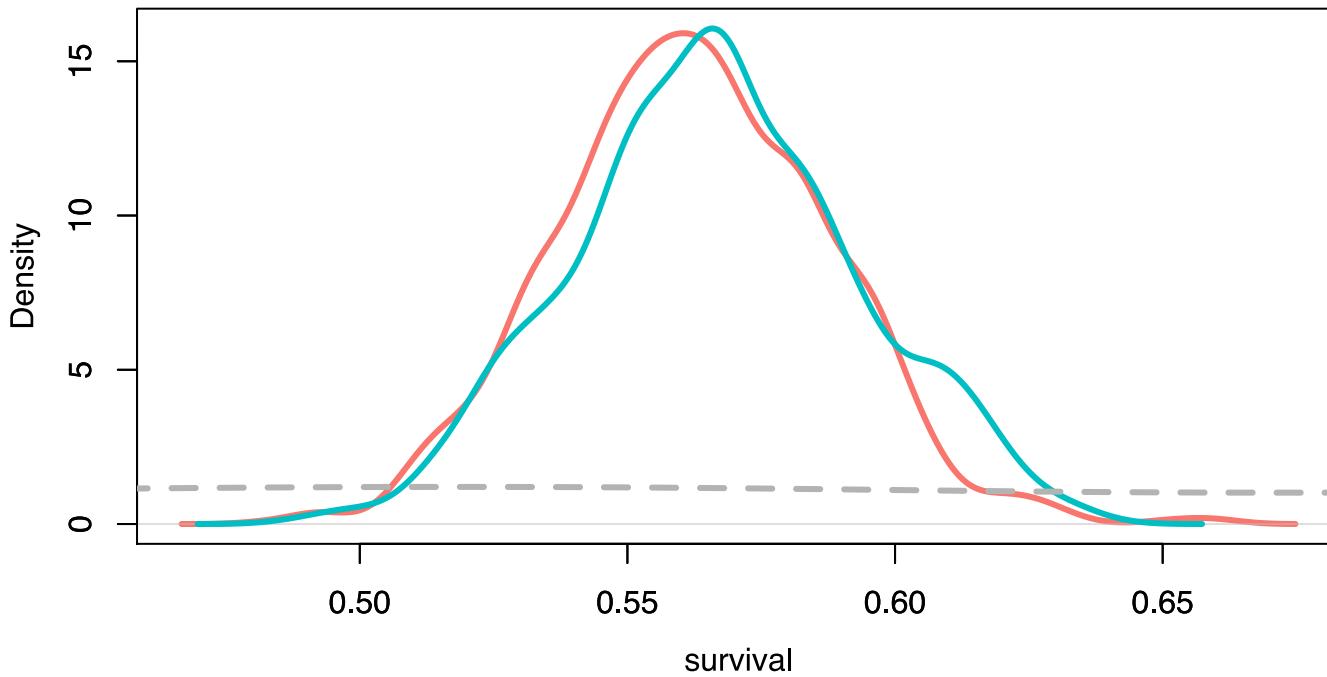
Why Bayes? Incorporate prior information.

Vague prior

- So far, we have assumed a vague prior:

$$\phi_{prior} \sim \text{Beta}(1, 1) = \text{Uniform}(0, 1)$$

- With a vague prior, mean posterior survival is $\phi_{posterior} = 0.56$
- With credible interval $[0.52, 0.62]$



Posterior distribution of survival in color (two chains), prior in gray dashed line.

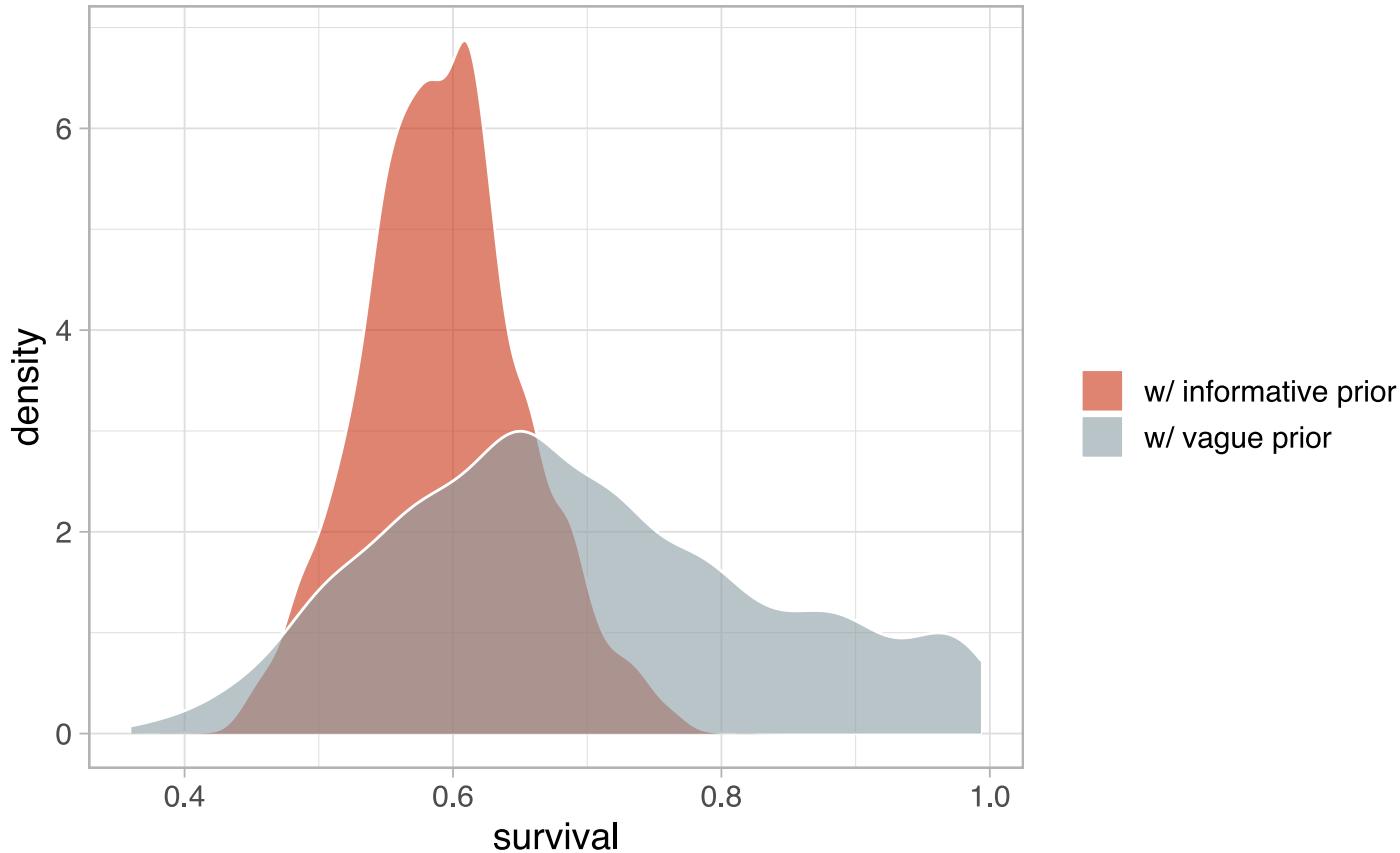
How to incorporate prior information?

- Using information on body mass and annual survival of 27 European passersines, we can predict survival of European dippers using only body mass.
- For dippers, body mass is 59.8g, therefore $\phi = 0.57$ with $sd = 0.073$.
- Assuming an informative prior $\phi_{prior} \sim \text{Normal}(0.57, 0.073^2)$.
- Mean posterior $\phi_{posterior} = 0.56$ with credible interval $[0.52, 0.61]$.
- No increase of precision in posterior inference.

How to incorporate prior information?

- Now if you had only the three first years of data, what would have happened?
- Width of credible interval is 0.53 (vague prior) vs. 0.24 (informative prior).
- Huge increase of precision in posterior inference, a 120% gain!

Compare survival posterior with and without informative prior



Prior elicitation via moment matching

- The prior $\phi_{prior} \sim \text{Normal}(0.57, 0.073^2)$ is not entirely satisfying
- Remember the Beta distribution
- Recall that the Beta distribution is a continuous distribution with values between 0 and 1. Useful for modelling survival or detection probabilities.
- If $X \sim Beta(\alpha, \beta)$, then the first and second moments of X are:

$$\mu = E(X) = \frac{\alpha}{\alpha + \beta}$$

$$\sigma^2 = \text{Var}(X) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

Moment matching

- In the capture-recapture example, we know a priori that the mean of the probability we're interested in is $\mu = 0.57$ and its variance is $\sigma^2 = 0.073^2$.
- Parameters μ and σ^2 are seen as the moments of a $Beta(\alpha, \beta)$ distribution.
- Now we look for values of α and β that match the observed moments of the Beta distribution μ and σ^2 .
- We need another set of equations:

$$\alpha = \left(\frac{1 - \mu}{\sigma^2} - \frac{1}{\mu} \right) \mu^2$$

$$\beta = \alpha \left(\frac{1}{\mu} - 1 \right)$$

Moment matching

- For our model, that means:

```
(alpha <- ( (1 - 0.57)/(0.073*0.073) - (1/0.57) )*0.57^2)
```

```
[1] 25.64636
```

```
(beta <- alpha * ( (1/0.57) - 1))
```

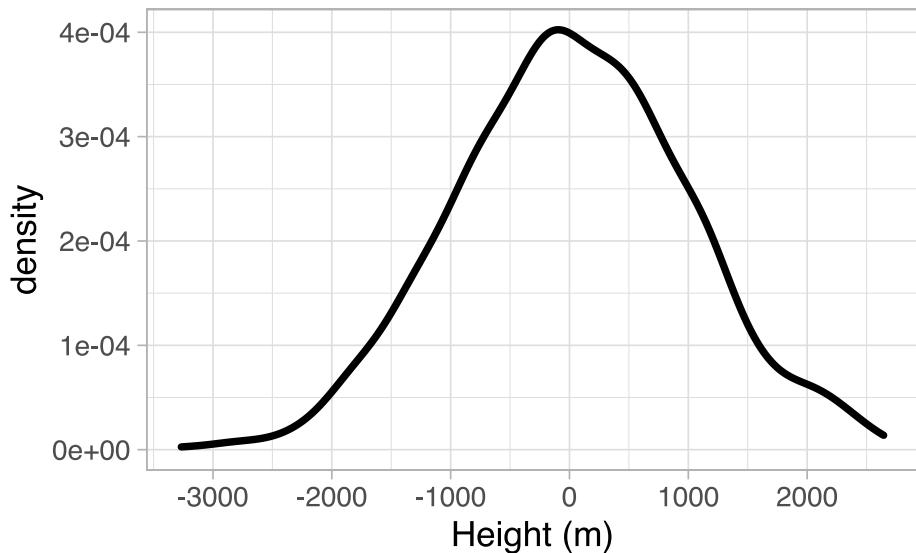
```
[1] 19.34726
```

- Now use $\phi_{prior} \sim \text{Beta}(\alpha = 25.6, \beta = 19.3)$ instead of $\phi_{prior} \sim \text{Normal}(0.57, 0.073^2)$

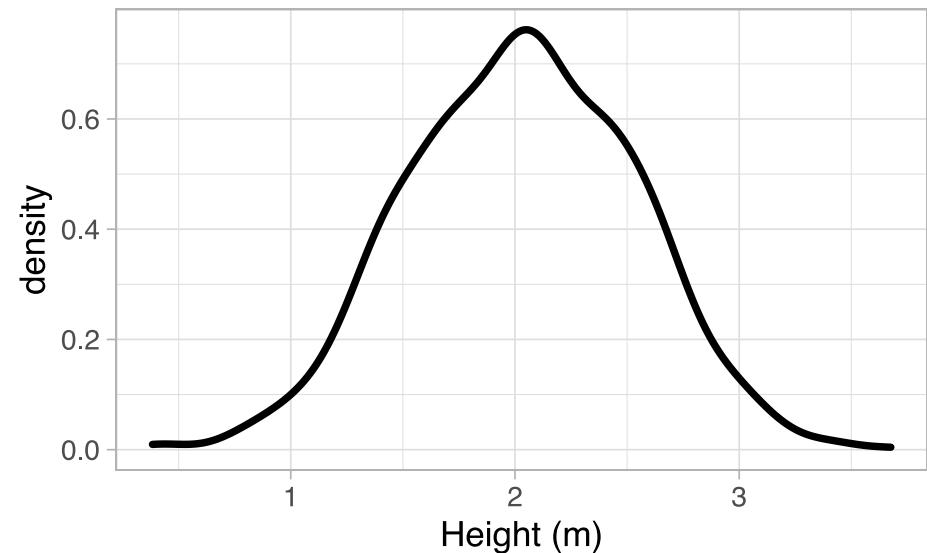
Prior predictive checks

Linear regression

Unreasonable prior $\beta \sim N(0, 1000^2)$



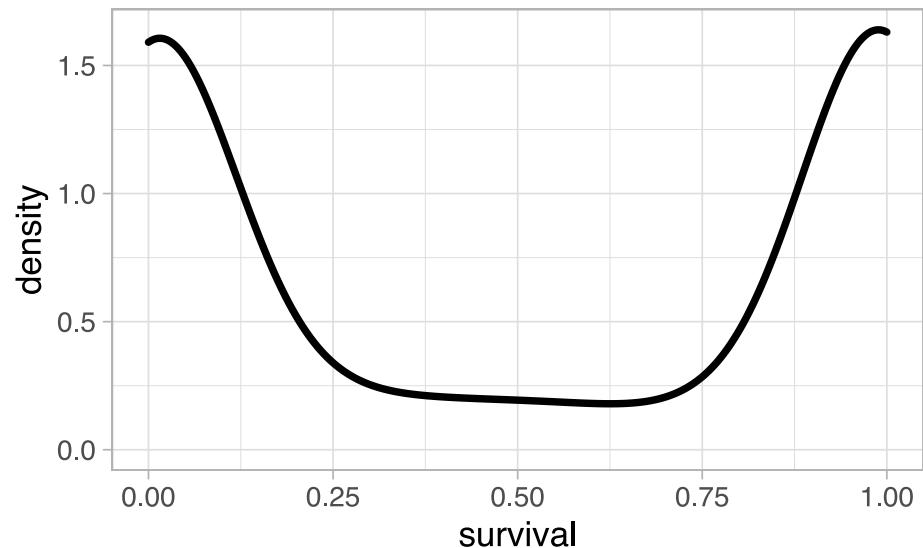
Reasonable prior $\beta \sim N(2, 0.5^2)$



Logistic regression

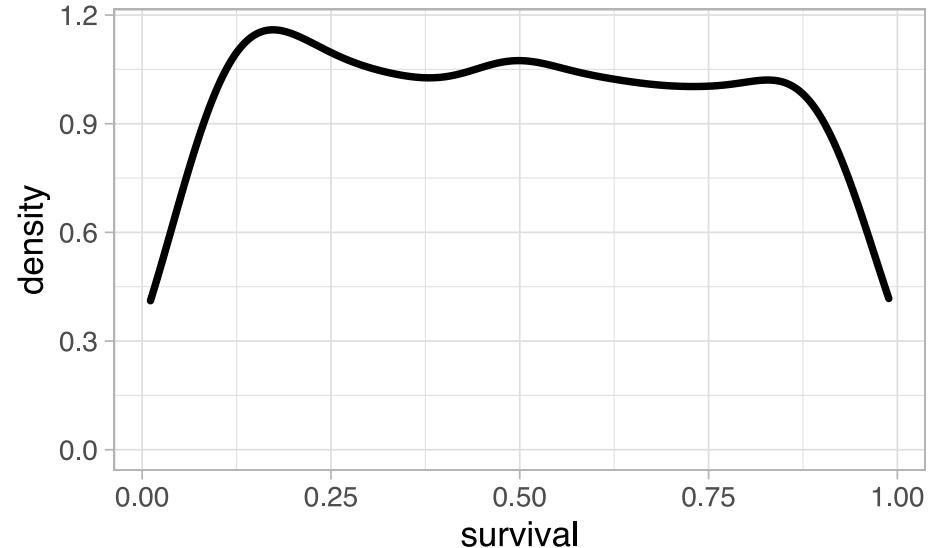
Unreasonable prior

$$\text{logit}(\phi) = \beta \sim N(0, 10^2)$$



Reasonable prior

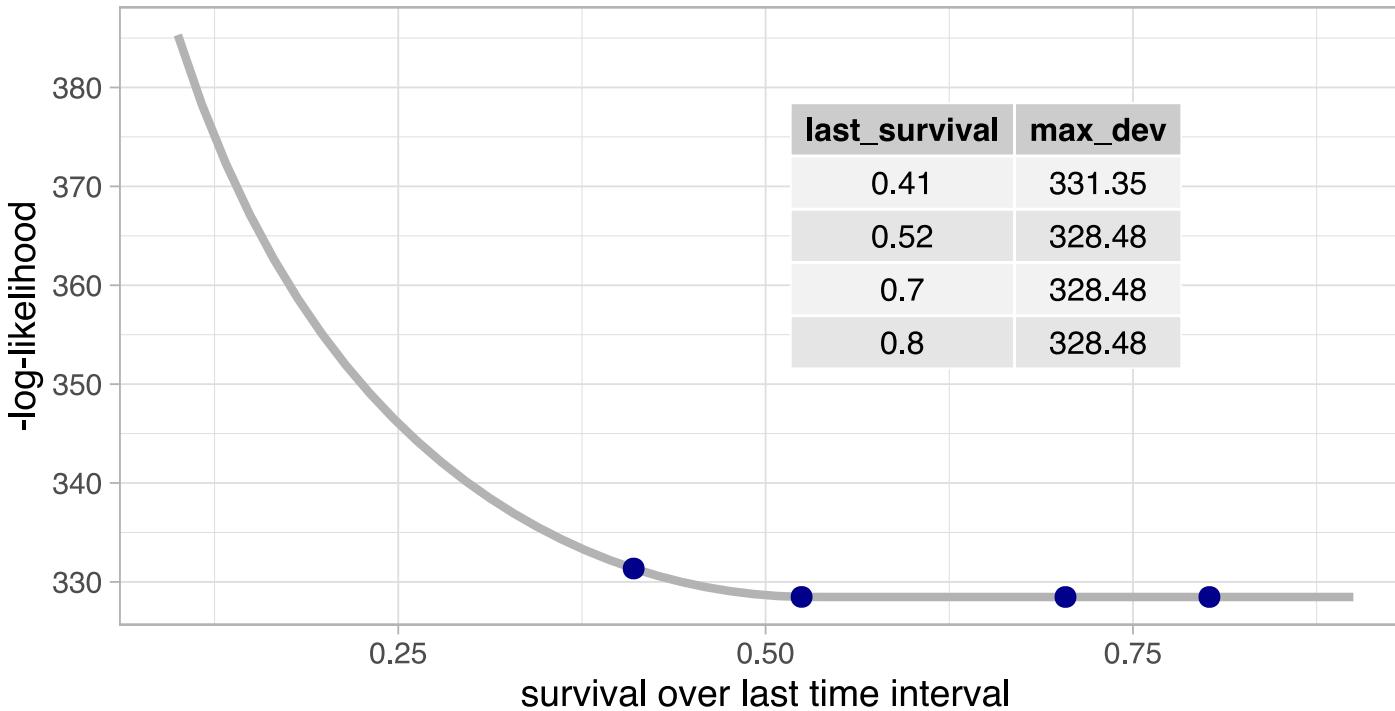
$$\text{logit}(\phi) = \beta \sim N(0, 1.5^2)$$



Capture-recapture models rely on assumptions

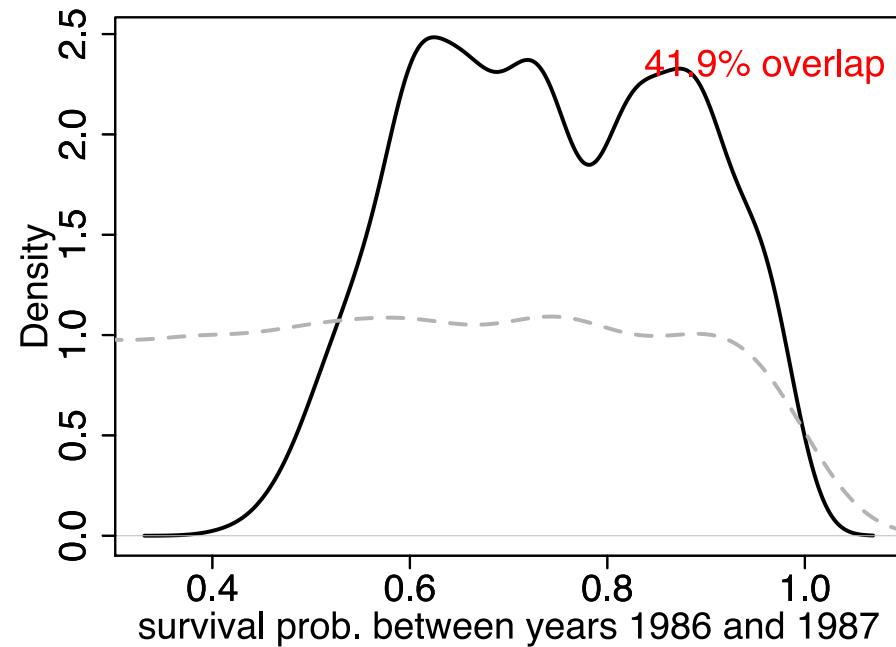
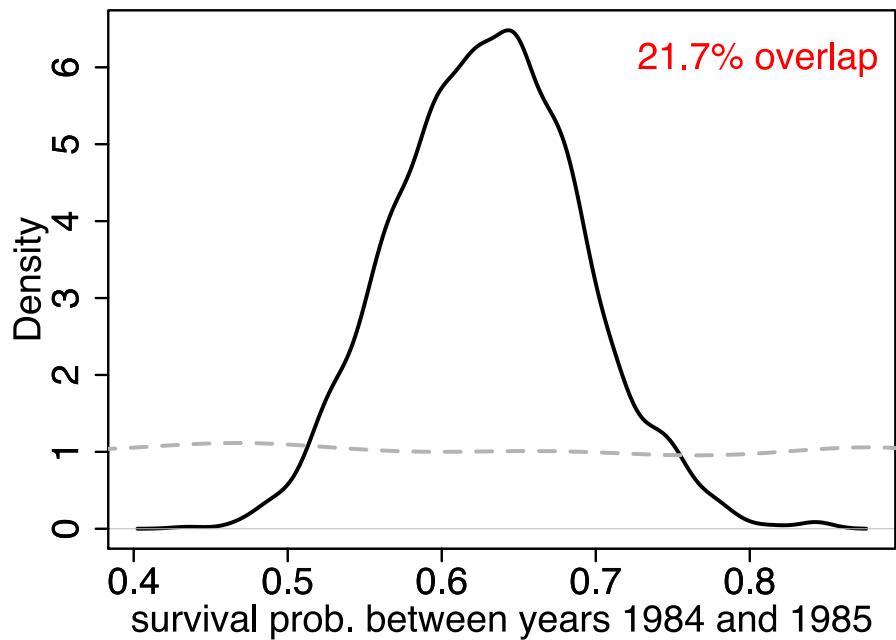
- Design
 - No mark lost
 - Identity of individuals recorded without error (no false positives)
 - Captured individuals are a random sample
- Model
 - Homogeneity of survival and recapture probabilities
 - Independence between individuals (overdispersion)
- Test validity of assumptions
 - These assumptions should be valid, whatever inferential framework
 - Use goodness-of-fit tests – Pradel et al. (2005)
 - R implementation with package `R2ucare`
 - Posterior predictive checks can also be used (not covered; Gelman et al. 2020)

Parameter-redundancy issue

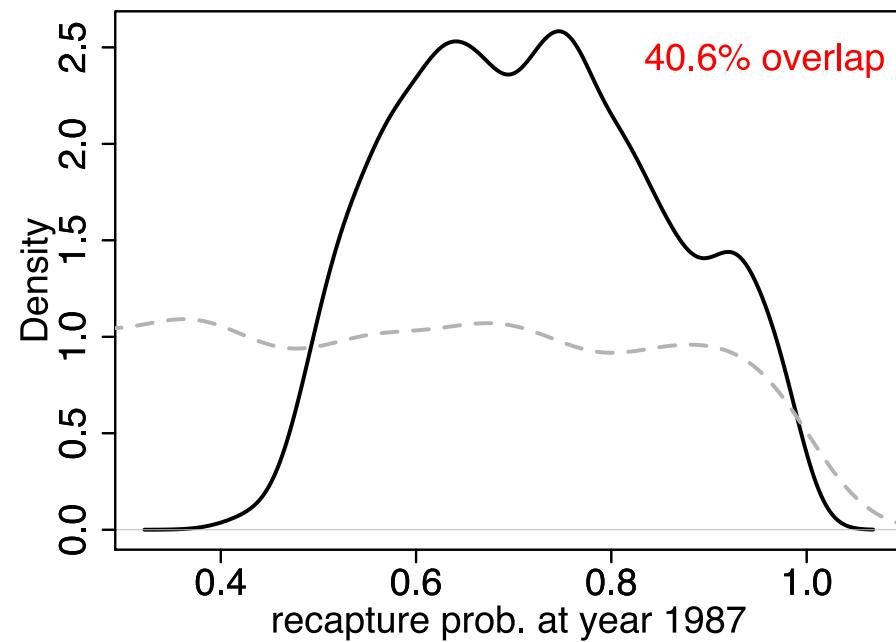
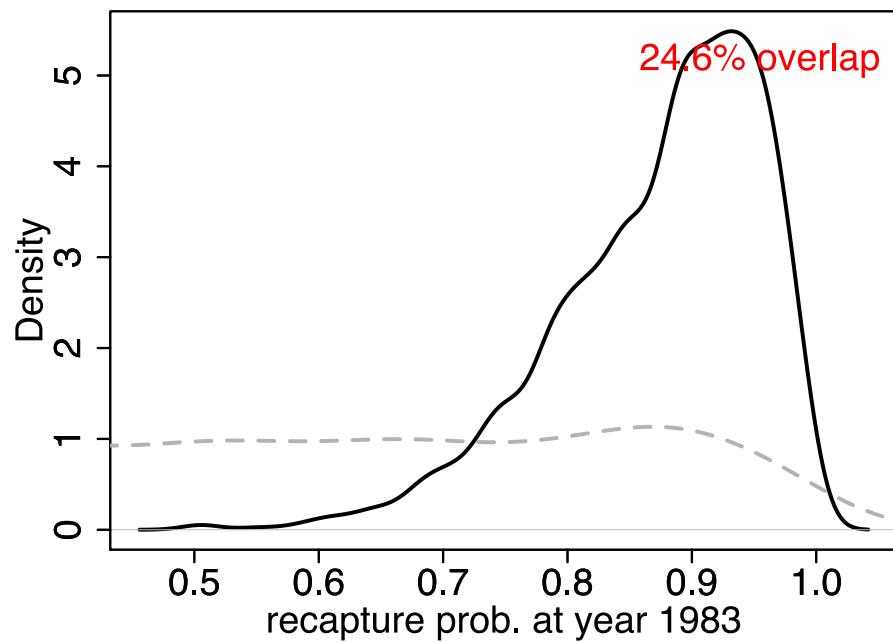


- Last survival and recapture probabilities cannot be estimated separately.
- Poor mixing of the chains.

Prior-posterior overlap for ϕ_4 and ϕ_6



Prior-posterior overlap for p_3 and p_7



What does survival actually mean in capture-recapture ?

- Survival refers to the study area.
- Mortality and permanent emigration are confounded.
- Therefore we estimate apparent survival, not true survival.
- Apparent survival probability = true survival \times study area fidelity.
- Consequently, apparent survival < true survival unless study area fidelity = 1.
- Use caution with interpretation. If possible, combine with ring-recovery data, or go spatial to get closer to true survival.

Further reading

- CJS state-space formulation [Gimenez et al. \(2007\)](#) and [Royle \(2008\)](#).
- Work on missing values by [Bonner et al. \(2006\)](#) and [Langrock and King \(2013\)](#) and [Worthington et al. \(2015\)](#).
- The example on how to incorporate prior information is in [McCarthy and Masters \(2005\)](#).
- Combine live recapture w/ dead recoveries by [Lebreton et al. \(1999\)](#) and go spatial to account for emigration [Gilroy et al. \(2012\)](#) and [Schaub & Royle \(2014\)](#).
- Non-identifiability in a Bayesian framework, see [Gimenez et al. \(2009\)](#) and [book by Cole \(2020\)](#).

On the move: Transition estimation

The team

last updated: 2021-05-18

THE ESTIMATION OF POPULATION SIZE, MIGRATION RATES AND SURVIVAL IN A STRATIFIED POPULATION

A. Neil ARNASON

Computer Science Department, University of Manitoba,
Winnipeg, Canada

INTRODUCTION

CHAPMAN and JUNGE (1956, hereafter referred to as C & J) developed estimates of stratum size and migration rates for a population divided into $n \geq 2$ areas (strata) when animals were free to migrate from area to area. The method was based on data from sampling and marking observations on two occasions. The method was extended by DARROCH (1961) to allow sampling in different numbers of strata at the two sampling times, and to show how to treat some special problems that arise when using the method. These problems arise when a particular data matrix (which must be inverted) is singular or ill-conditioned. The same problems could occur with the estimates which will be given in this paper.

In order to account for deaths or losses from the areas due to permanent emigration out of the areas being sampled, it is necessary to sample on at least three occasions. I developed estimates for the three sample experiment on two areas

BIOMETRICS 49, 177-193
March 1993

Estimating Migration Rates Using Tag-Recovery Data

Carl J. Schwarz

Department of Statistics, University of Manitoba,
Winnipeg, Manitoba R3T 2N2, Canada

Jake F. Schweigert

Biological Sciences Branch, Pacific Biological Station,
Department of Fisheries and Oceans, Nanaimo, British Columbia V9R 5K6, Canada

and

A. Neil Arnason

Department of Computer Science, University of Manitoba,
Winnipeg, Manitoba R3T 2N2, Canada

SUMMARY

Tag-recovery data are used to estimate migration rates among a set of strata. The model formulation is a simple matrix extension of the formulation of a tag-recovery experiment discussed by Brownie et al. (1985, *Statistical Inference from Band-Recovery Data—A Handbook*, 2nd edition, Washington, D.C.: U.S. Department of the Interior). Estimation is more difficult because of the convolution of parameters between release and recovery and this convolution may cause estimates of the survival/

Thank you Canada!



A photograph of Donald Trump speaking at the Republican National Convention in 2016. He is wearing a dark suit, white shirt, and red tie, standing behind a podium with a microphone. The background features a large American flag. The text "BIGGER AND BETTER AND STRONGER" is overlaid in large yellow letters.

**BIGGER AND BETTER
AND STRONGER**

LIVE
RNC
• 2016

ESTIMATING TRANSITION PROBABILITIES FOR STAGE-BASED POPULATION PROJECTION MATRICES USING CAPTURE–RECAPTURE DATA¹

JAMES D. NICHOLS AND JOHN R. SAUER

United States Fish and Wildlife Service, Patuxent Wildlife Research Center, Laurel, Maryland 20708 USA

KENNETH H. POLLOCK

Institute of Statistics, North Carolina State University, Box 8203, Raleigh, North Carolina 27695-8203 USA

JAY B. HESTBECK²

United States Fish and Wildlife Service, Patuxent Wildlife Research Center, Laurel, Maryland 20708 USA

Abstract. In stage-based demography, animals are often categorized into size (or mass) classes, and size-based probabilities of surviving and changing mass classes must be estimated before demographic analyses can be conducted. In this paper, we develop two procedures for the estimation of mass transition probabilities from capture–recapture data. The first approach uses a multistate capture–recapture model that is parameterized directly with the transition probabilities of interest. Maximum likelihood estimates are then obtained numerically using program SURVIV. The second approach involves a modification of Pollock's robust design. Estimation proceeds by conditioning on animals caught in a particular class at time i , and then using closed models to estimate the number of these that are alive in other classes at $i + 1$. Both methods are illustrated by application to meadow vole, *Microtus pennsylvanicus*, capture–recapture data. The two methods produced reasonable estimates that were similar. Advantages of these two approaches include the directness of estimation, the absence of need for restrictive assumptions about the independence of survival and growth, the testability of assumptions, and the testability of related hypotheses of ecological interest (e.g., the hypothesis of temporal variation in transition probabilities).

Key words: capture–recapture models; *Microtus pennsylvanicus*; multistate models; parameter estimation; Pollock's robust design; stage-based population projection matrices; stage transition probabilities.

ESTIMATING BREEDING PROPORTIONS AND TESTING HYPOTHESES ABOUT COSTS OF REPRODUCTION WITH CAPTURE–RECAPTURE DATA¹

JAMES D. NICHOLS AND JAMES E. HINES

National Biological Survey, Patuxent Wildlife Research Center, Laurel, Maryland 20708 USA

KENNETH H. POLLOCK

Institute of Statistics, North Carolina State University, Box 8203, Raleigh, North Carolina 27695-8203 USA

ROBERT L. HINZ AND WILLIAM A. LINK

National Biological Survey, Patuxent Wildlife Research Center, Laurel, Maryland 20708 USA

Abstract. The proportion of animals in a population that breeds is an important determinant of population growth rate. Usual estimates of this quantity from field sampling data assume that the probability of appearing in the capture or count statistic is the same for animals that do and do not breed. A similar assumption is required by most existing methods used to test ecologically interesting hypotheses about reproductive costs using field sampling data. However, in many field sampling situations breeding and nonbreeding animals are likely to exhibit different probabilities of being seen or caught. In this paper, we propose the use of multistate capture–recapture models for these estimation and testing problems. This methodology permits a formal test of the hypothesis of equal capture/sighting probabilities for breeding and nonbreeding individuals. Two estimators of breeding proportion (and associated standard errors) are presented, one for the case of equal capture probabilities and one for the case of unequal capture probabilities. The multistate modeling framework also yields formal tests of hypotheses about reproductive costs to future reproduction or survival or both fitness components. The general methodology is illustrated using capture–recapture data on female meadow voles, *Microtus pennsylvanicus*. Resulting estimates of the proportion of reproductively active females showed strong seasonal variation, as expected, with low breeding proportions in midwinter. We found no evidence of reproductive costs extracted in subsequent survival or reproduction. We believe that this methodological framework has wide application to problems in animal ecology concerning breeding proportions and phenotypic reproductive costs.

Key words: capture/sighting probability; *Microtus pennsylvanicus*; multistate capture–recapture models; proportion of animals breeding; reproductive costs; survival rate.

Wintering site fidelity in Canada Geese



3 sites Carolinas, Chesapeake, Mid-Atlantic, with 21277 banded geese,
data kindly provided by Jay Hestbeck ([Hestbeck et al. 1991](#))

	year_1984	year_1985	year_1986	year_1987	year_1988	year_1989
	0	2	2	0	0	0
	0	0	0	0	0	2
	0	0	0	1	0	0
	0	0	2	0	0	0
	0	3	0	0	3	2
	0	0	0	2	0	0
	2	2	0	2	3	2
	0	0	0	0	2	2

Biological inference

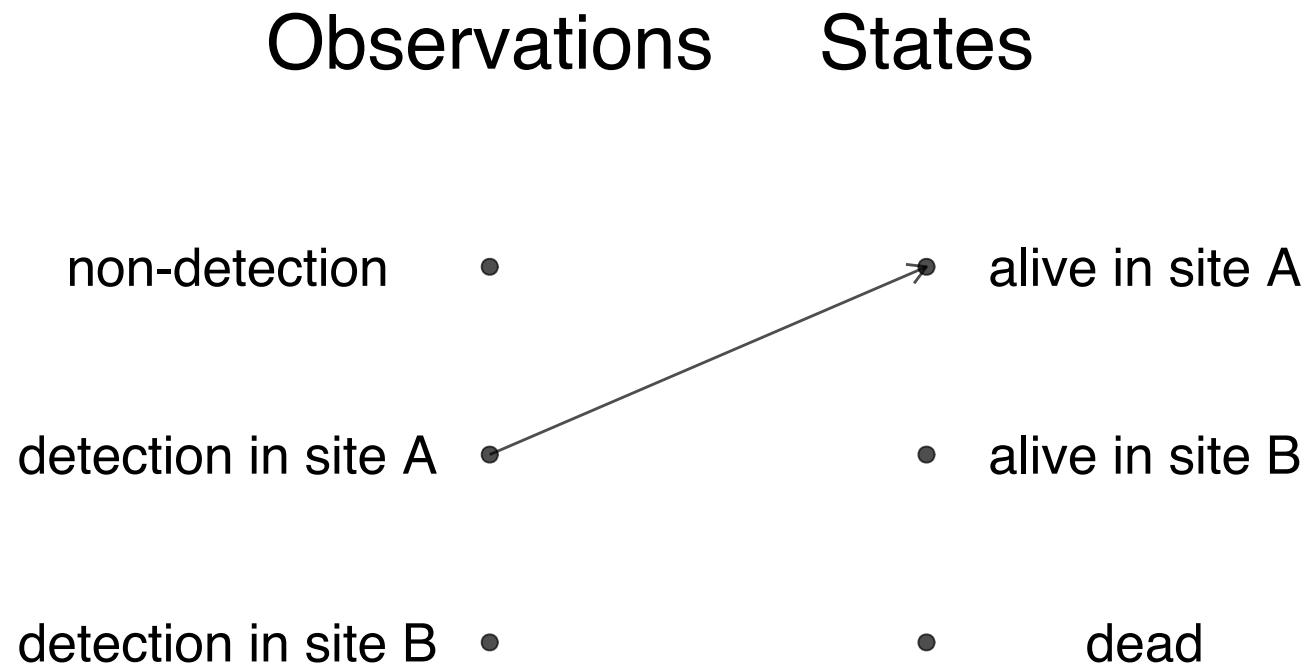
Observations States

non-detection • • alive in site A

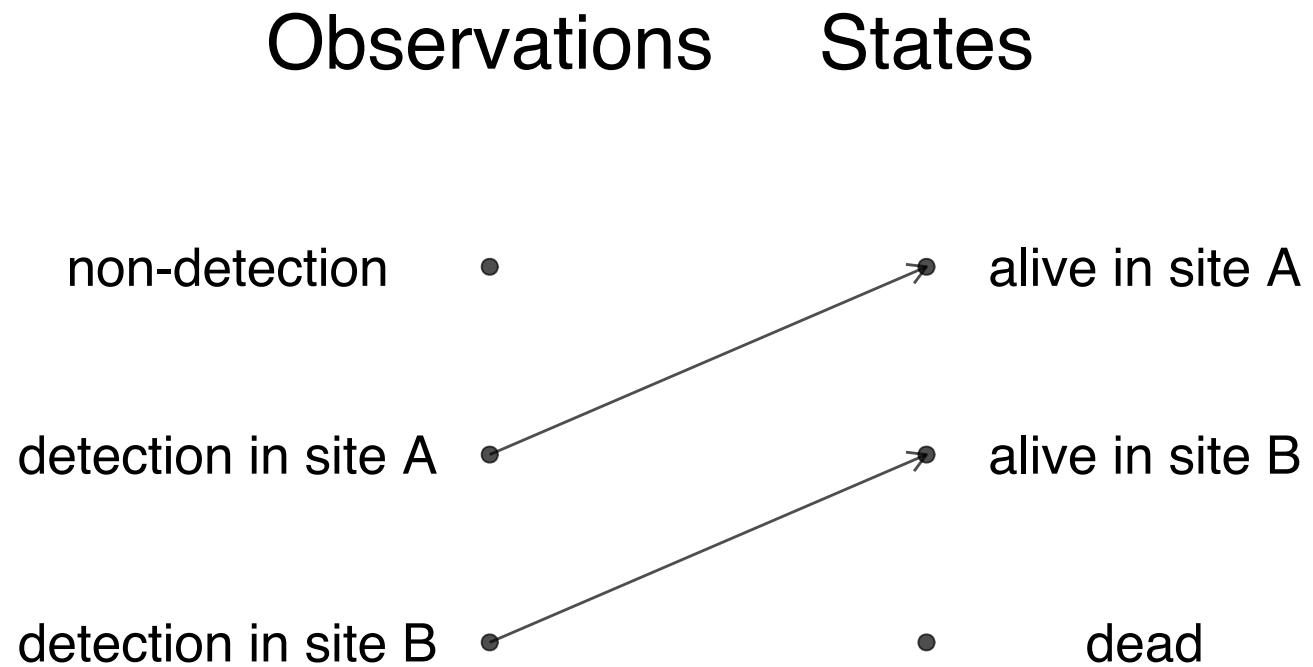
detection in site A • • alive in site B

detection in site B • • dead

Biological inference

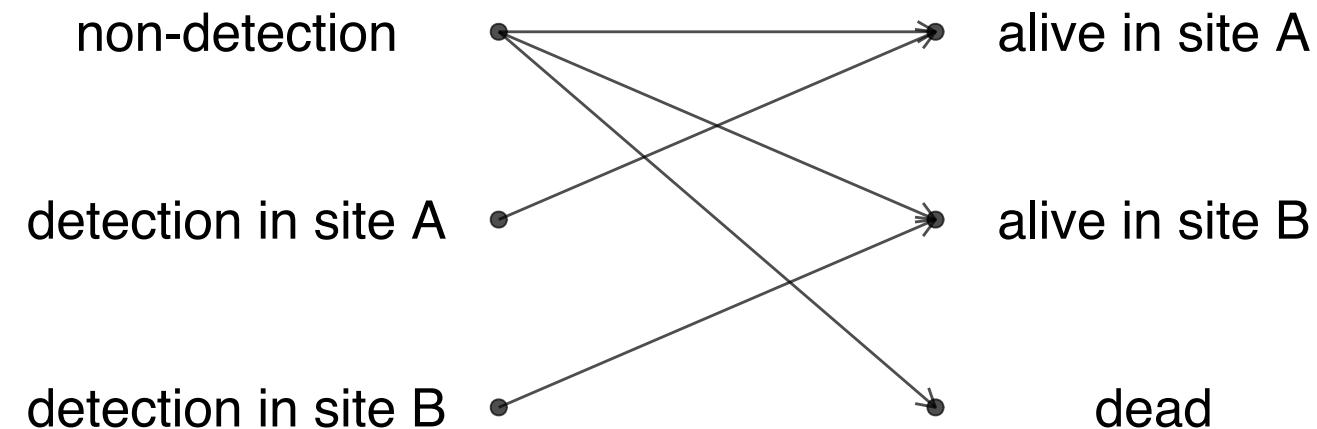


Biological inference



Biological inference

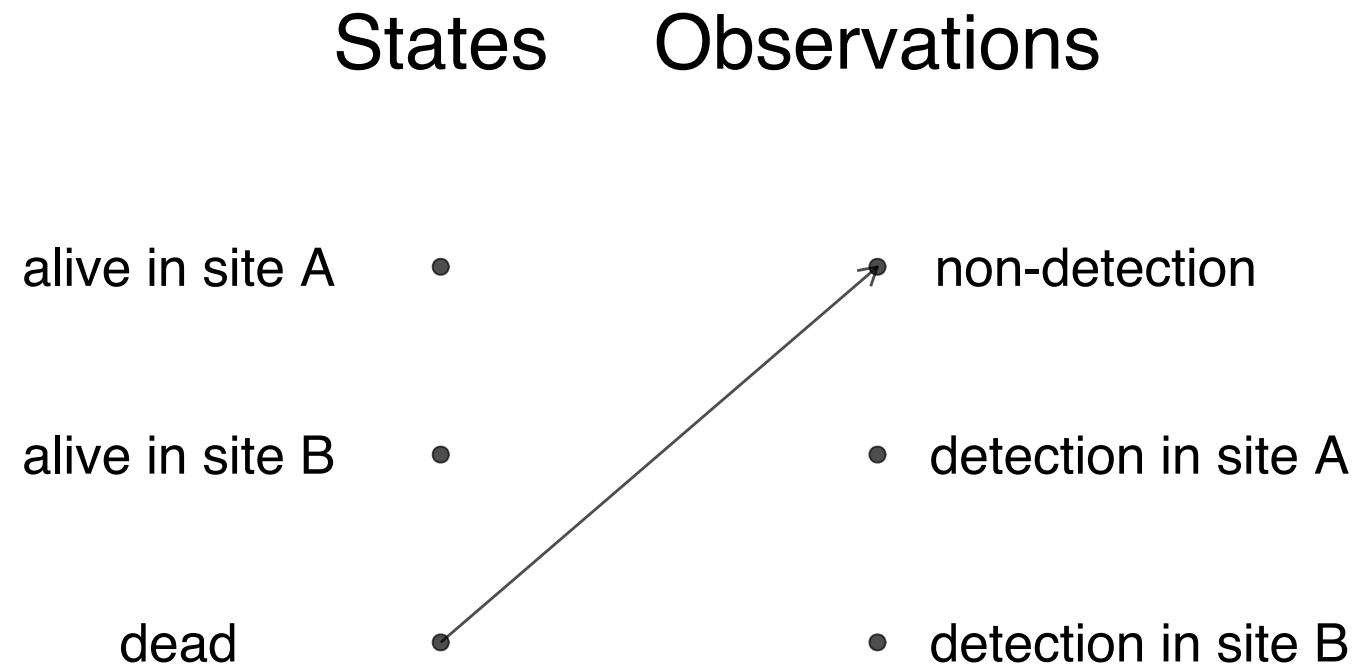
Observations States



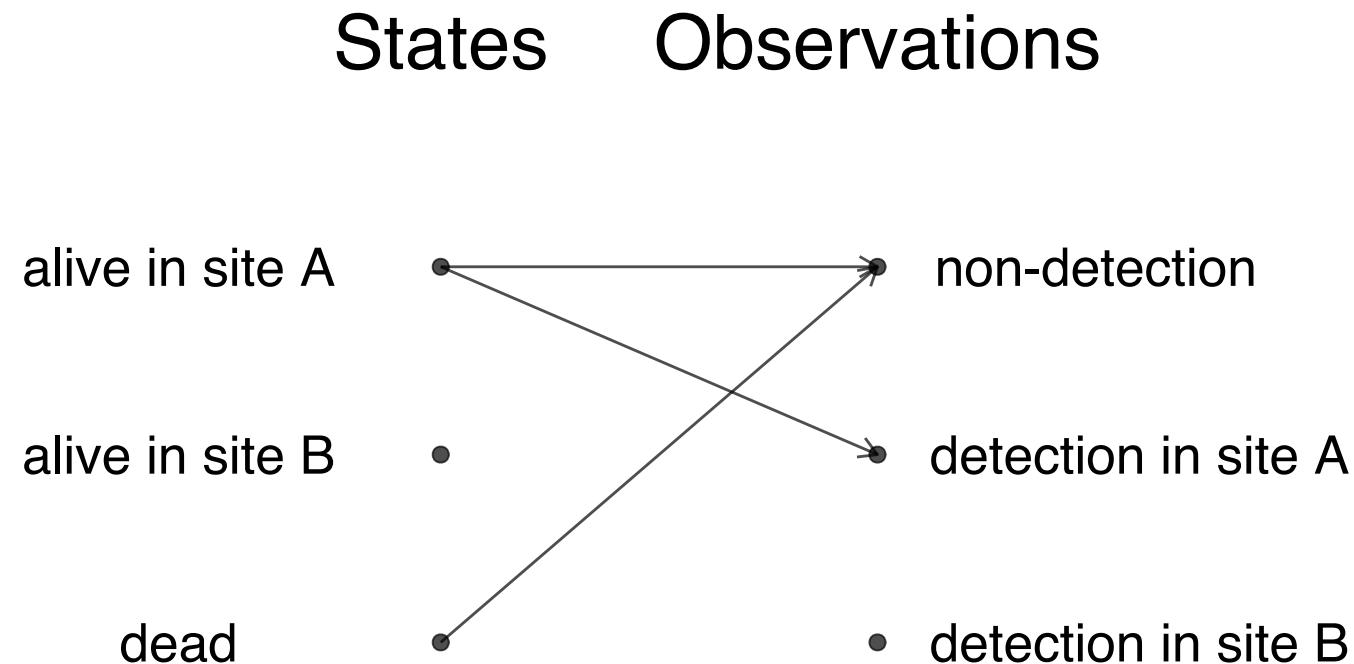
The model construction: How we should think.

States	Observations
alive in site A	<ul style="list-style-type: none">• non-detection
alive in site B	<ul style="list-style-type: none">• detection in site A
dead	<ul style="list-style-type: none">• detection in site B

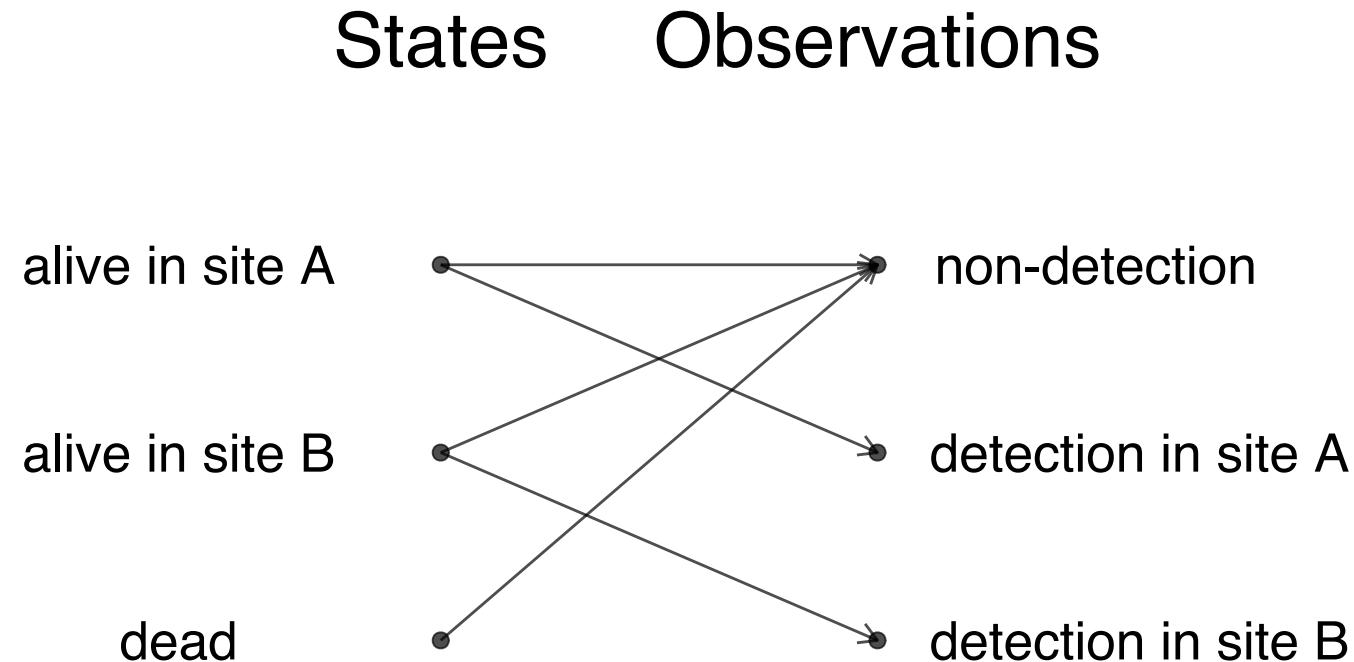
The model construction: How we should think.



The model construction: How we should think.



The model construction: How we should think.



HMM model for dispersal with 2 sites (drop Carolinas)

Transition matrix

$$\boldsymbol{\Gamma} = \left(\begin{array}{ccc} z_t = A & z_t = B & z_t = D \\ \hline \phi_A(1 - \psi_{AB}) & \phi_A\psi_{AB} & 1 - \phi_A \\ \phi_B\psi_{BA} & \phi_B(1 - \psi_{BA}) & 1 - \phi_B \\ 0 & 0 & 1 \end{array} \right) \begin{array}{l} z_{t-1} = A \\ z_{t-1} = B \\ z_{t-1} = D \end{array}$$

HMM model for dispersal with 2 sites (drop Carolinas)

Observation matrix

$$\Omega = \left(\begin{array}{ccc} y_t = 0 & y_t = 1 & y_t = 2 \\ \hline 1 - p_A & p_A & 0 \\ 1 - p_B & 0 & p_B \\ 1 & 0 & 0 \end{array} \right) \begin{array}{l} z_t = A \\ z_t = B \\ z_t = D \end{array}$$

HMM model for dispersal with 2 sites (drop Carolinas)

Observation matrix

$$\Omega = \left(\begin{array}{ccc} y_t = 0 & y_t = 1 & y_t = 2 \\ \hline 1 - p_A & p_A & 0 \\ 1 - p_B & 0 & p_B \\ 1 & 0 & 0 \end{array} \right) \begin{array}{l} z_t = A \\ z_t = B \\ z_t = D \end{array}$$

Note: You may code non-detections as $y_t = 2$, and the first column in the observation matrix should go last.

Our model $(\phi_A, \phi_B, \psi_{AB}, \psi_{BA}, p_A, p_B)$

```
multisite <- nimbleCode({
  # -----
  # Parameters:
  # phiA: survival probability site A
  # phiB: survival probability site B
  # psiAB: movement probability from site A to site B
  # psiBA: movement probability from site B to site A
  # pA: recapture probability site A
  # pB: recapture probability site B
  # -----
  # States (z):
  # 1 alive at A
  # 2 alive at B
  # 3 dead
  # Observations (y):
  # 1 not seen
  # 2 seen at A
  # 3 seen at B
  # -----
  ...
})
```

Our model $(\phi_A, \phi_B, \psi_{AB}, \psi_{BA}, p_A, p_B)$

```
multisite <- nimbleCode( {  
  ...  
  # Priors  
  phiA ~ dunif(0, 1)  
  phiB ~ dunif(0, 1)  
  psiAB ~ dunif(0, 1)  
  psiBA ~ dunif(0, 1)  
  pA ~ dunif(0, 1)  
  pB ~ dunif(0, 1)  
  ...  
}
```

Our model $(\phi_A, \phi_B, \psi_{AB}, \psi_{BA}, p_A, p_B)$

```
multisite <- nimbleCode( {  
  ...  
  # initial state probabilities  
  delta[1] <- piA          # Pr(alive in A t = 1)  
  delta[2] <- 1 - piA       # Pr(alive in B t = 1)  
  delta[3] <- 0             # Pr(dead t = 1) = θ  
  ...  
}
```

- Actually, initial state is known exactly. It is alive at site of initial capture, and π_A is just the proportion of individuals first captured in site A, no need to estimate it.
- Instead of `z[i,first[i]] ~ dcat(delta[1:3])`, use `z[i,first[i]] <- y[i,first[i]]-1` instead in the likelihood.
- Same trick applies to CJS models.

Our model $(\phi_A, \phi_B, \psi_{AB}, \psi_{BA}, p_A, p_B)$

```
multisite <- nimbleCode( {  
  ...  
  # probabilities of state  $z(t+1)$  given  $z(t)$   
  # (read as  $\text{gamma}[z(t), z(t+1)] = \text{gamma}[\text{fromState}, \text{toState}]$ )  
  
  gamma[1,1] <- phiA * (1 - psiAB)  
  gamma[1,2] <- phiA * psiAB  
  gamma[1,3] <- 1 - phiA  
  gamma[2,1] <- phiB * psiBA  
  gamma[2,2] <- phiB * (1 - psiBA)  
  gamma[2,3] <- 1 - phiB  
  gamma[3,1] <- 0  
  gamma[3,2] <- 0  
  gamma[3,3] <- 1  
  ...  
}
```

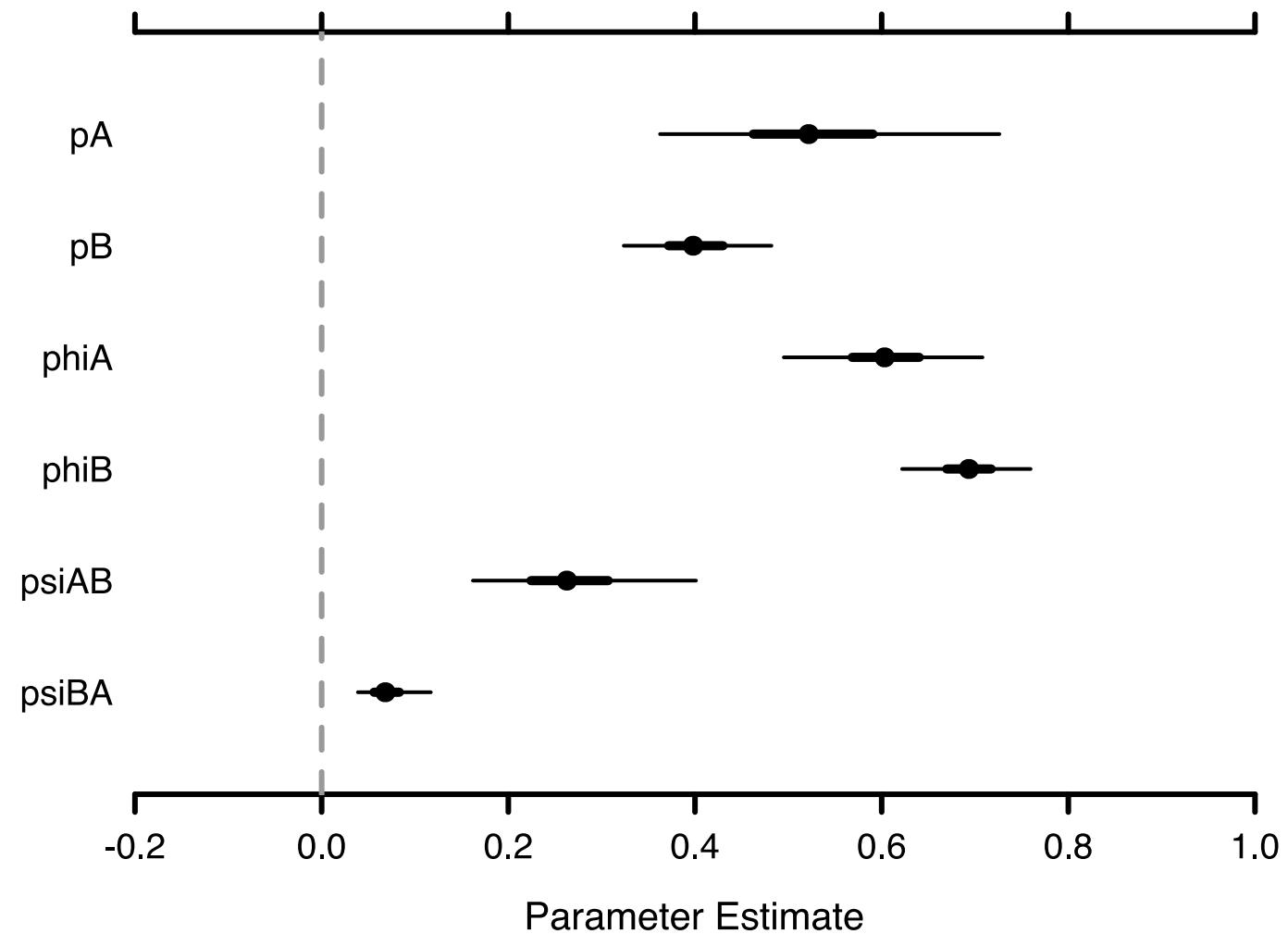
Our model $(\phi_A, \phi_B, \psi_{AB}, \psi_{BA}, p_A, p_B)$

```
multisite <- nimbleCode( {
  ...
  # probabilities of  $y(t)$  given  $z(t)$ 
  # (read as  $\text{omega}[y(t), z(t)] = \text{omega}[\text{Observation}, \text{State}]$ )
  ...
   $\text{omega}[1,1] <- 1 - pA$       #  $\Pr(\text{alive } A \text{ at } t \rightarrow \text{non-detected } t)$ 
   $\text{omega}[1,2] <- pA$           #  $\Pr(\text{alive } A \text{ at } t \rightarrow \text{detected } A \text{ at } t)$ 
   $\text{omega}[1,3] <- 0$            #  $\Pr(\text{alive } A \text{ at } t \rightarrow \text{detected } B \text{ at } t)$ 
   $\text{omega}[2,1] <- 1 - pB$       #  $\Pr(\text{alive } B \text{ at } t \rightarrow \text{non-detected } t)$ 
   $\text{omega}[2,2] <- 0$            #  $\Pr(\text{alive } B \text{ at } t \rightarrow \text{detected } A \text{ at } t)$ 
   $\text{omega}[2,3] <- pB$           #  $\Pr(\text{alive } B \text{ at } t \rightarrow \text{detected } B \text{ at } t)$ 
   $\text{omega}[3,1] <- 1$            #  $\Pr(\text{dead } t \rightarrow \text{non-detected } t)$ 
   $\text{omega}[3,2] <- 0$            #  $\Pr(\text{dead } t \rightarrow \text{detected } A \text{ at } t)$ 
   $\text{omega}[3,3] <- 0$            #  $\Pr(\text{dead } t \rightarrow \text{detected } B \text{ at } t)$ 
  ...
}
```

Our model $(\phi_A, \phi_B, \psi_{AB}, \psi_{BA}, p_A, p_B)$

```
multisite <- nimbleCode( {  
  ...  
  # likelihood  
  for (i in 1:N){  
    # latent state at first capture  
    z[i,first[i]] <- y[i,first[i]] - 1  
    for (t in (first[i]+1):K){  
      # z(t) given z(t-1)  
      z[i,t] ~ dcat(gamma[z[i,t-1],1:3])  
      # y(t) given z(t)  
      y[i,t] ~ dcat(omega[z[i,t],1:3])  
    }  
  }  
})
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
pA	0.53	0.09	0.36	0.52	0.73	1.04	122
pB	0.40	0.04	0.32	0.40	0.48	1.07	165
phiA	0.60	0.05	0.50	0.60	0.71	1.01	195
phiB	0.69	0.04	0.62	0.69	0.76	1.04	199
psiAB	0.27	0.06	0.16	0.26	0.40	1.04	244
psiBA	0.07	0.02	0.04	0.07	0.12	1.03	360

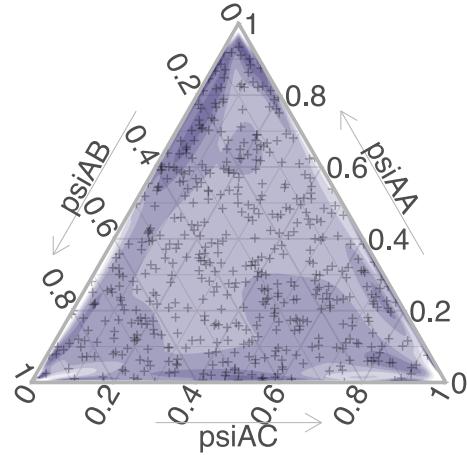


What if there are three sites?

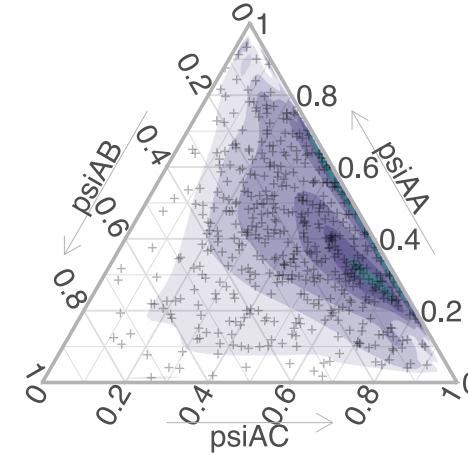
- The transition probabilities still need to be between 0 and 1.
- Another constraint is that the sum of three probabilities of departure from a given site should be one.
- Two methods to fulfill both constraints.
 - Dirichlet prior
 - Multinomial logit link

Dirichlet prior with parameter alpha

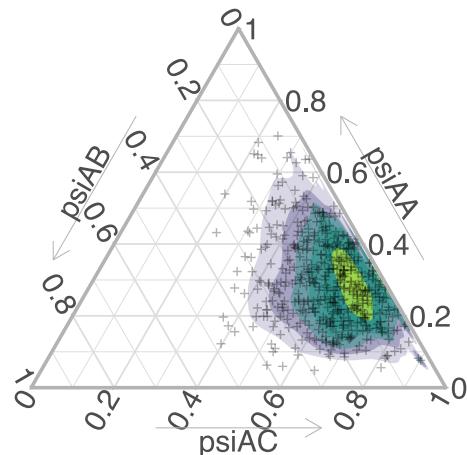
alpha = c(1, 1, 1)



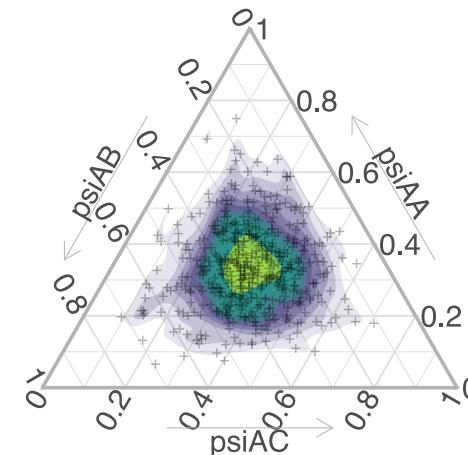
alpha = c(1, 2, 2)



alpha = c(2, 4, 8)



alpha = c(5, 5, 5)



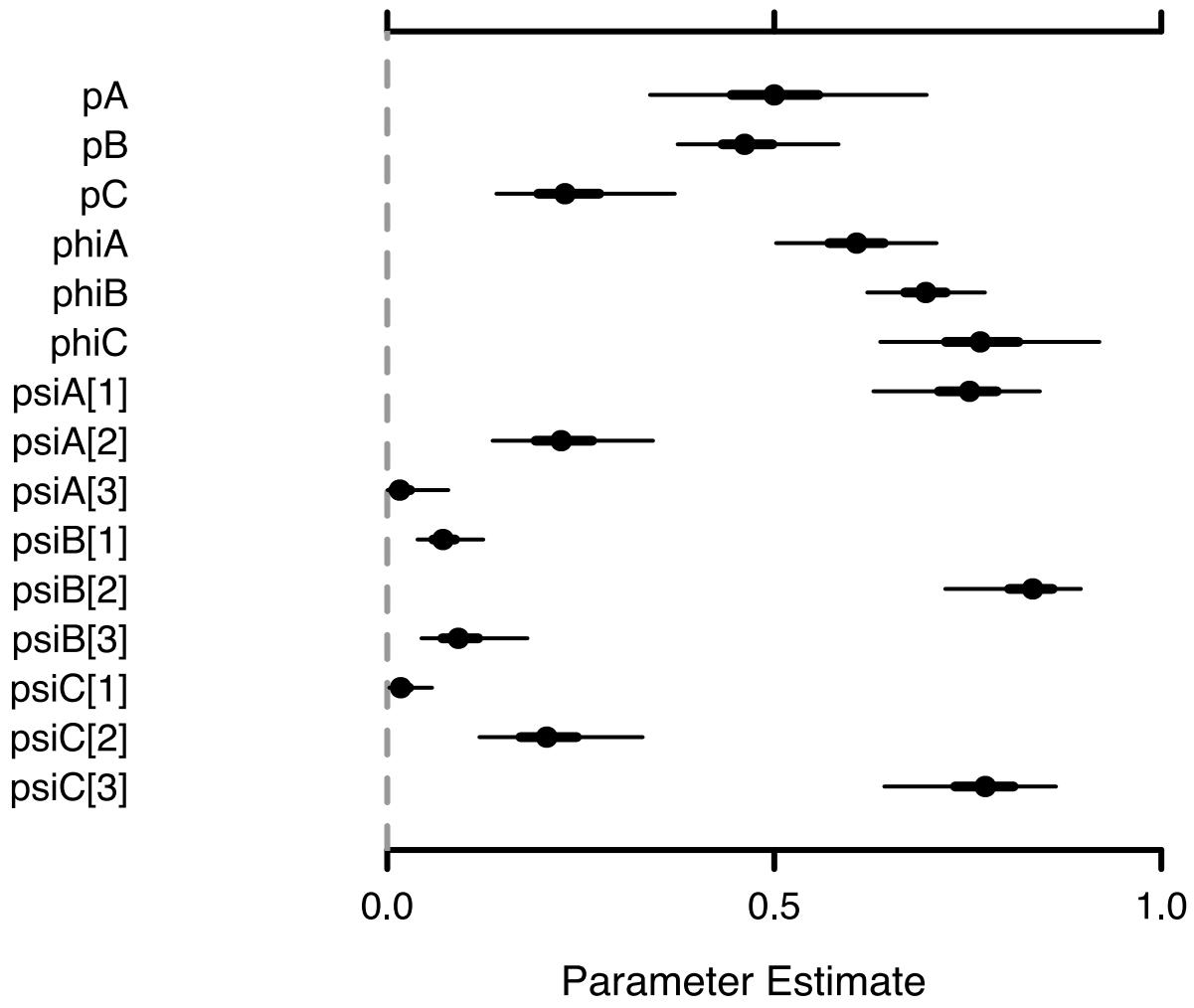
Nimble implementation of the Dirichlet prior

```
multisite <- nimbleCode( {  
  ...  
  # transitions: Dirichlet priors  
  psiA[1:3] ~ ddirch(alpha[1:3]) # psiAA, psiAB, psiAC  
  psiB[1:3] ~ ddirch(alpha[1:3]) # psiBA, psiBB, psiCC  
  psiC[1:3] ~ ddirch(alpha[1:3]) # psiCA, psiCB, psiCC  
  ...  
}
```

Nimble implementation of the Dirichlet prior

```
multisite <- nimbleCode({  
  ...  
  # probabilities of state z(t+1) given z(t)  
  gamma[1,1] <- phiA * psiA[1]  
  gamma[1,2] <- phiA * psiA[2]  
  gamma[1,3] <- phiA * psiA[3]  
  gamma[1,4] <- 1 - phiA  
  gamma[2,1] <- phiB * psiB[1]  
  gamma[2,2] <- phiB * psiB[2]  
  gamma[2,3] <- phiB * psiB[3]  
  gamma[2,4] <- 1 - phiB  
  gamma[3,1] <- phiC * psiC[1]  
  gamma[3,2] <- phiC * psiC[2]  
  gamma[3,3] <- phiC * psiC[3]  
  gamma[3,4] <- 1 - phiC  
  gamma[4,1] <- 0  
  gamma[4,2] <- 0  
  gamma[4,3] <- 0  
  gamma[4,4] <- 1  
  ...
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
pA	0.50	0.09	0.34	0.50	0.70	1.00	153
pB	0.47	0.05	0.38	0.46	0.58	1.01	152
pC	0.24	0.06	0.14	0.23	0.37	1.01	117
phiA	0.61	0.05	0.50	0.61	0.71	1.00	230
phiB	0.70	0.04	0.62	0.70	0.77	1.04	183
phiC	0.77	0.07	0.64	0.77	0.92	1.07	104
psiA[1]	0.75	0.05	0.63	0.75	0.84	1.01	463
psiA[2]	0.23	0.05	0.14	0.22	0.34	1.01	441
psiA[3]	0.02	0.02	0.00	0.02	0.08	1.03	201
psiB[1]	0.07	0.02	0.04	0.07	0.12	1.00	275
psiB[2]	0.83	0.04	0.72	0.83	0.90	1.04	129
psiB[3]	0.10	0.04	0.04	0.09	0.18	1.06	129
psiC[1]	0.02	0.01	0.00	0.02	0.06	1.00	624
psiC[2]	0.21	0.05	0.12	0.21	0.33	1.02	420
psiC[3]	0.77	0.06	0.64	0.77	0.86	1.02	419



Multinomial logit

- Say we have P sites or states.
- Specify a normal prior distribution for $P - 1$ transition parameters α_j . These probabilities are on the multinomial logit scale, possibly function of covariates.
- To back-transform these parameters, we use:

$$\beta_j = \frac{\exp(\alpha_j)}{1 + \sum_{p=1}^P \exp(\alpha_p)}, j = 1, \dots, P - 1$$

- This ensures that all β_j are between 0 and 1, and their sum is 1.
- Last parameter is calculated as the complement $\beta_P = 1 - \sum_{j=1}^{P-1} \exp(\beta_j)$

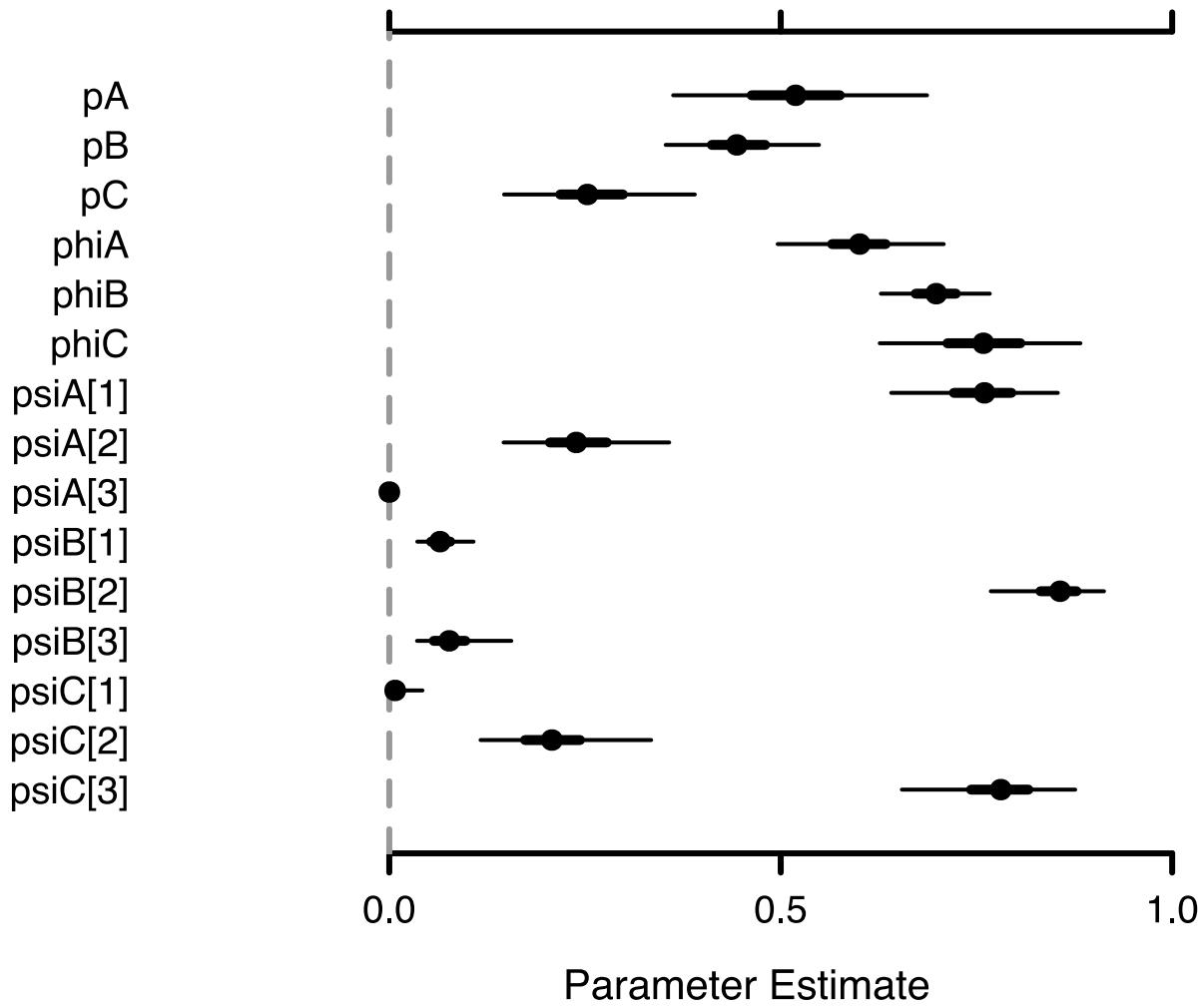
Nimble implementation of the Dirichlet prior

```
multisite <- nimbleCode({  
  ...  
  # transitions: multinomial logit  
  # normal priors on logit of all but one transition probs  
  for (i in 1:2){  
    lpsiA[i] ~ dnorm(0, sd = 1000)  
    lpsiB[i] ~ dnorm(0, sd = 1000)  
    lpsiC[i] ~ dnorm(0, sd = 1000)  
  }  
  # constrain the transitions such that their sum is < 1  
  for (i in 1:2){  
    psiA[i] <- exp(lpsiA[i]) / (1 + exp(lpsiA[1]) + exp(lpsiA[2]))  
    psiB[i] <- exp(lpsiB[i]) / (1 + exp(lpsiB[1]) + exp(lpsiB[2]))  
    psiC[i] <- exp(lpsiC[i]) / (1 + exp(lpsiC[1]) + exp(lpsiC[2]))  
  }  
  # last transition probability  
  psiA[3] <- 1 - psiA[1] - psiA[2]  
  psiB[3] <- 1 - psiB[1] - psiB[2]  
  psiC[3] <- 1 - psiC[1] - psiC[2]  
  ...
```

Nimble implementation of the Dirichlet prior

```
multisite <- nimbleCode({  
  ...  
  # probabilities of state z(t+1) given z(t)  
  gamma[1,1] <- phiA * psiA[1]  
  gamma[1,2] <- phiA * psiA[2]  
  gamma[1,3] <- phiA * psiA[3]  
  gamma[1,4] <- 1 - phiA  
  gamma[2,1] <- phiB * psiB[1]  
  gamma[2,2] <- phiB * psiB[2]  
  gamma[2,3] <- phiB * psiB[3]  
  gamma[2,4] <- 1 - phiB  
  gamma[3,1] <- phiC * psiC[1]  
  gamma[3,2] <- phiC * psiC[2]  
  gamma[3,3] <- phiC * psiC[3]  
  gamma[3,4] <- 1 - phiC  
  gamma[4,1] <- 0  
  gamma[4,2] <- 0  
  gamma[4,3] <- 0  
  gamma[4,4] <- 1  
  ...
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
pA	0.52	0.08	0.36	0.52	0.69	1.02	154
pB	0.45	0.05	0.35	0.44	0.55	1.10	129
pC	0.26	0.06	0.15	0.25	0.39	1.01	94
phiA	0.60	0.05	0.50	0.60	0.71	1.01	244
phiB	0.70	0.04	0.63	0.70	0.77	1.11	168
phiC	0.76	0.07	0.63	0.76	0.88	1.03	126
psiA[1]	0.76	0.05	0.64	0.76	0.85	1.02	477
psiA[2]	0.24	0.05	0.15	0.24	0.36	1.01	486
psiA[3]	0.00	0.00	0.00	0.00	0.00	1.35	47
psiB[1]	0.07	0.02	0.04	0.06	0.11	1.03	394
psiB[2]	0.85	0.04	0.77	0.86	0.91	1.04	133
psiB[3]	0.08	0.03	0.04	0.08	0.16	1.01	79
psiC[1]	0.01	0.01	0.00	0.01	0.04	1.00	514
psiC[2]	0.21	0.05	0.12	0.21	0.33	1.00	299
psiC[3]	0.78	0.06	0.65	0.78	0.88	1.00	270



Live demo



Sites may be states.

Examples of multistate models

- *Epidemiological or disease states:* sick/healthy, uninfected/infected/recovered.
- *Morphological states:* small/medium/big, light/medium/heavy.
- *Breeding states:* e.g. breeder/non-breeder, failed breeder, first-time breeder.
- *Developmental or life-history states:* e.g. juvenile/subadult/adult.
- *Social states:* e.g. solitary/group-living, subordinate/dominant.
- *Death states:* e.g. alive, dead from harvest, dead from natural causes.

States = individual, time-specific categorical covariates.

Sooty shearwater (David Boyle)



Sooty shearwaters and life-history tradeoffs

- We consider data collected between 1940 and 1957 by Lance Richdale on Sooty shearwaters (aka titis).
- These data were reanalyzed with multistate models by [Scofield et al. \(2001\)](#) who kindly provided us with the data.
- Following the way the data were collected, four states were originally considered:
 - Alive breeder;
 - Accompanied by another bird in a burrow;
 - Alone in a burrow;
 - On the surface;
 - Dead.

Sooty shearwaters and life-history tradeoffs

- Because of numerical issues, we pooled all alive states but breeder together in a non-breeder state (NB) that includes:
 - failed breeders (birds that had bred previously – skip reproduction or divorce) and pre-breeders (birds that had yet to breed).
 - Note that because burrows were not checked before hatching, some birds in the category NB might have already failed.
 - We therefore regard those birds in the B state as successful breeders, and those in the NB state as nonbreeders plus prebreeders and failed breeders.
- Observations are non-detections, and detections as breeder and non-breeder
- Does breeding affect survival? Does breeding in current year affect breeding next year?

HMM model for transition between states

Transition matrix

$$\boldsymbol{\Gamma} = \begin{array}{ccc|c} & z_t = B & z_t = NB & z_t = D \\ \hline \begin{matrix} z_{t-1} = B \\ z_{t-1} = NB \\ z_{t-1} = D \end{matrix} & \phi_B(1 - \psi_{BNB}) & \phi_B\psi_{BNB} & 1 - \phi_B \\ & \phi_{NB}\psi_{NBB} & \phi_{NB}(1 - \psi_{NBB}) & 1 - \phi_{NB} \\ & 0 & 0 & 1 \end{array}$$

- Costs or reproduction would reflect in future reproduction
 $\psi_{BB} = 1 - \psi_{BNB} < \psi_{NBB}$ or survival $\phi_B < \phi_{NB}$.

HMM model for transition between states

Observation matrix

$$\Omega = \begin{array}{ccc} & y_t = 0 & y_t = 1 & y_t = 2 \\ \hline & 1 - p_B & p_B & 0 \\ & 1 - p_{NB} & 0 & p_{NB} \\ & 1 & 0 & 0 \end{array} \quad \begin{array}{l} z_t = B \\ z_t = NB \\ z_t = D \end{array}$$

Our model $(\phi_{NB}, \phi_B, \psi_{NBB}, \psi_{BNB}, p_{NB}, p_B)$

```
multistate <- nimbleCode({
  # -----
  # Parameters:
  # phiB: survival probability state B
  # phiNB: survival probability state NB
  # psiBNB: transition probability from B to NB
  # psiNBB: transition probability from NB to B
  # pB: recapture probability B
  # pNB: recapture probability NB
  # -----
  # States (z):
  # 1 alive B
  # 2 alive NB
  # 3 dead
  # Observations (y):
  # 1 not seen
  # 2 seen as B
  # 3 seen as NB
  # -----
  ...
})
```

Our model $(\phi_{NB}, \phi_B, \psi_{NBB}, \psi_{BNB}, p_{NB}, p_B)$

```
multistate <- nimbleCode( {  
  ...  
  # Priors  
  phiB ~ dunif(0, 1)  
  phiNB ~ dunif(0, 1)  
  psiBNB ~ dunif(0, 1)  
  psiNBB ~ dunif(0, 1)  
  pB ~ dunif(0, 1)  
  pNB ~ dunif(0, 1)  
  ...  
}
```

Our model $(\phi_{NB}, \phi_B, \psi_{NBB}, \psi_{BNB}, p_{NB}, p_B)$

```
multistate <- nimbleCode({  
  ...  
  # probabilities of state  $z(t+1)$  given  $z(t)$   
  gamma[1,1] <- phiB * (1 - psiBNB)  
  gamma[1,2] <- phiB * psiBNB  
  gamma[1,3] <- 1 - phiB  
  gamma[2,1] <- phiNB * psiNBB  
  gamma[2,2] <- phiNB * (1 - psiNBB)  
  gamma[2,3] <- 1 - phiNB  
  gamma[3,1] <- 0  
  gamma[3,2] <- 0  
  gamma[3,3] <- 1  
  ...
```

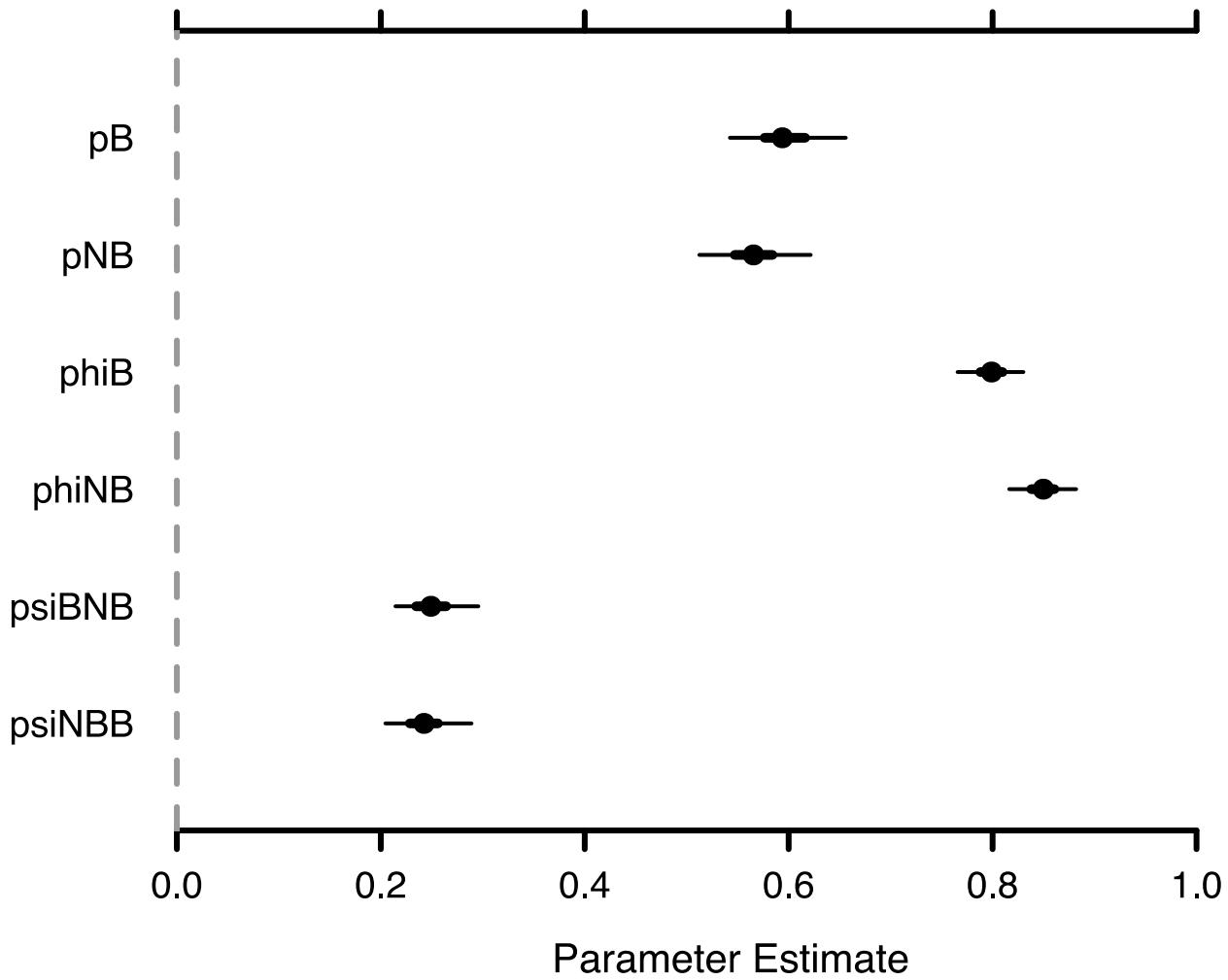
Our model $(\phi_{NB}, \phi_B, \psi_{NBB}, \psi_{BNB}, p_{NB}, p_B)$

```
multistate <- nimbleCode({  
  ...  
  # probabilities of  $y(t)$  given  $z(t)$   
  omega[1,1] <- 1 - pB      #  $Pr(\text{alive } B \text{ } t \rightarrow \text{non-detected } t)$   
  omega[1,2] <- pB          #  $Pr(\text{alive } B \text{ } t \rightarrow \text{detected } B \text{ } t)$   
  omega[1,3] <- 0            #  $Pr(\text{alive } B \text{ } t \rightarrow \text{detected NB } t)$   
  omega[2,1] <- 1 - pNB     #  $Pr(\text{alive } NB \text{ } t \rightarrow \text{non-detected } t)$   
  omega[2,2] <- 0            #  $Pr(\text{alive } NB \text{ } t \rightarrow \text{detected } B \text{ } t)$   
  omega[2,3] <- pNB          #  $Pr(\text{alive } NB \text{ } t \rightarrow \text{detected NB } t)$   
  omega[3,1] <- 1            #  $Pr(\text{dead } t \rightarrow \text{non-detected } t)$   
  omega[3,2] <- 0            #  $Pr(\text{dead } t \rightarrow \text{detected } N \text{ } t)$   
  omega[3,3] <- 0            #  $Pr(\text{dead } t \rightarrow \text{detected NB } t)$   
  ...  
})
```

Our model $(\phi_{NB}, \phi_B, \psi_{NBB}, \psi_{BNB}, p_{NB}, p_B)$

```
multistate <- nimbleCode( {
  ...
  # likelihood
  for (i in 1:N){
    # latent state at first capture
    z[i,first[i]] <- y[i,first[i]] - 1
    for (t in (first[i]+1):K){
      # z(t) given z(t-1)
      z[i,t] ~ dcat(gamma[z[i,t-1],1:3])
      # y(t) given z(t)
      y[i,t] ~ dcat(omega[z[i,t],1:3])
    }
  }
})
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
pB	0.60	0.03	0.54	0.59	0.66	1.00	202
pNB	0.57	0.03	0.51	0.57	0.62	1.01	281
phiB	0.80	0.02	0.77	0.80	0.83	1.01	313
phiNB	0.85	0.02	0.82	0.85	0.88	1.00	404
psiBNB	0.25	0.02	0.21	0.25	0.30	1.00	434
psiNBB	0.24	0.02	0.20	0.24	0.29	1.03	478



Multistate models are very flexible

- Access to reproduction
- Temporary emigration
- Combination of life and dead encounters

Access to reproduction

Transition matrix:

$$\Gamma = \begin{pmatrix} z_t = J & z_t = 1yNB & z_t = 2yNB & z_t = B & z_t = D \\ \hline 0 & \phi_1(1 - \alpha_1) & 0 & \phi_1\alpha_1 & 1 - \phi_1 \\ 0 & 0 & \phi_2(1 - \alpha_2) & \phi_2\alpha_2 & 1 - \phi_2 \\ 0 & 0 & 0 & \phi_3 & 1 - \phi_3 \\ 0 & 0 & 0 & \phi_B & 1 - \phi_B \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{array}{l} z_{t-1} = J \\ z_{t-1} = 1yNB \\ z_{t-1} = 2yNB \\ z_{t-1} = B \\ z_{t-1} = D \end{array}$$

- First-year and second-year individuals breed with probabilities α_1 and α_2 .
- Then, everybody breeds from age 3.

Access to reproduction

Observation matrix:

$$\boldsymbol{\Omega} = \left(\begin{array}{cccc} y_t = 0 & y_t = 1 & y_t = 2 & y_t = 3 \\ \hline 1 & 0 & 0 & 0 \\ 1 - p_1 & p_1 & 0 & 0 \\ 1 - p_2 & 0 & p_2 & 0 \\ 1 - p_3 & 0 & 0 & p_3 \\ 1 & 0 & 0 & 0 \end{array} \right) \quad \begin{array}{l} z_t = J \\ z_t = 1yNB \\ z_t = 2yNB \\ z_t = B \\ z_t = D \end{array}$$

- Juveniles are never detected.

Temporary emigration

Transition matrix:

$$\boldsymbol{\Gamma} = \begin{array}{ccc|c} & z_t = \text{in} & z_t = \text{out} & z_t = \text{D} \\ \hline \phi(1 - \psi_{\text{in} \rightarrow \text{out}}) & \phi\psi_{\text{in} \rightarrow \text{out}} & 1 - \phi & \\ \phi\psi_{\text{out} \rightarrow \text{in}} & \phi(1 - \psi_{\text{out} \rightarrow \text{in}}) & 1 - \phi & \\ 0 & 0 & 1 & \end{array} \begin{array}{l} z_{t-1} = \text{in} \\ z_{t-1} = \text{out} \\ z_{t-1} = \text{D} \end{array}$$

Observation matrix:

$$\boldsymbol{\Omega} = \begin{array}{cc|c} y_t = 0 & y_t = 1 & \\ \hline 1 - p & p & \\ 1 & 0 & \\ 1 & 0 & \end{array} \begin{array}{l} z_t = \text{in} \\ z_t = \text{out} \\ z_t = \text{D} \end{array}$$

Combination of life and dead encounters

Transition matrix

$$\Gamma = \begin{array}{ccc} z_t = A & z_t = JD & z_t = D \\ \hline s & 1-s & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{array} \quad \begin{array}{l} z_{t-1} = \text{alive} \\ z_{t-1} = \text{just dead} \\ z_{t-1} = \text{dead for good} \end{array}$$

Observation matrix

$$\Omega = \begin{array}{ccc} y_t = 0 & y_t = 1 & y_t = 2 \\ \hline 1-p & 0 & p \\ 1-r & r & 0 \\ 1 & 0 & 0 \end{array} \quad \begin{array}{l} z_t = A \\ z_t = JD \\ z_t = D \end{array}$$

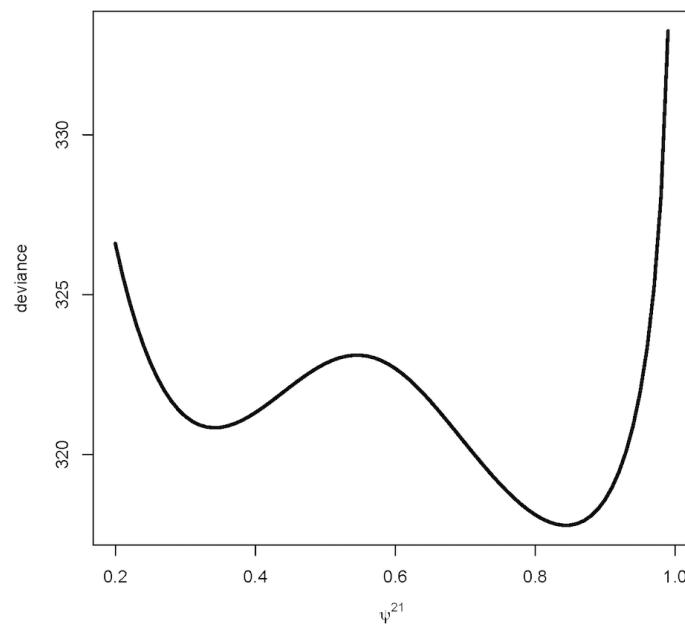
Issue of local minima

- Simulated data
 - 2 sites or states, and 7 occasions
 - Survival $\phi = 1$, detection $p = 0.6$
 - Transition $\psi_{12} = 0.6$
 - Transition $\psi_{21} = 0.85$
- Courtesy of Jérôme Dupuis, used in Gimenez et al. (2005).

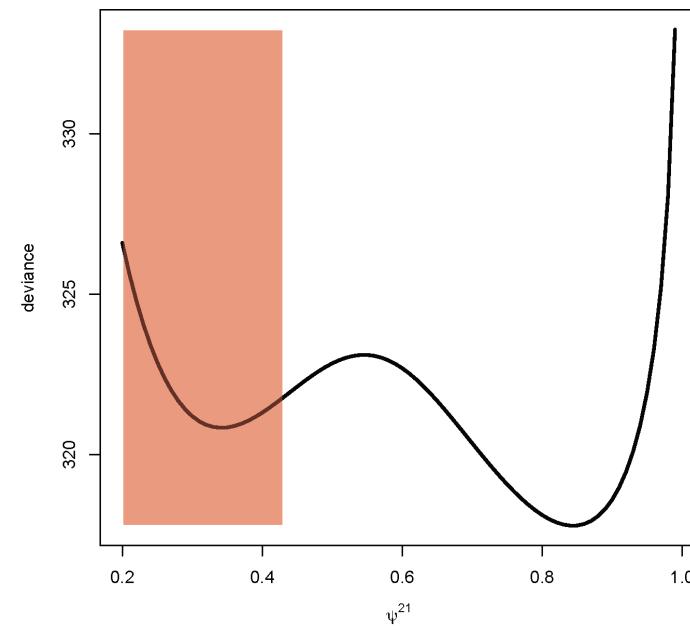
Data

V1	V2	V3	V4	V5	V6	V7
2	0	2	1	2	0	2
2	0	2	1	2	0	2
2	0	2	1	2	0	2
2	0	2	1	2	0	2
1	1	1	0	1	0	1
1	1	1	0	1	0	1
1	1	1	0	1	0	1

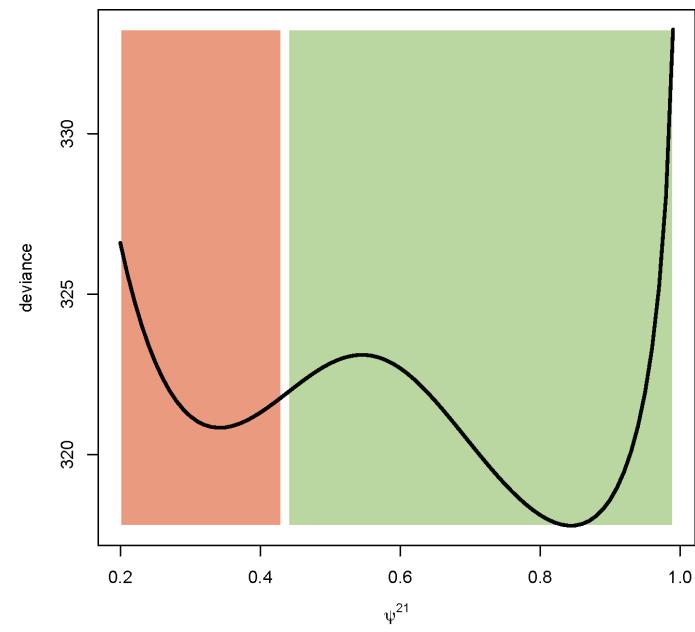
Deviance as a function of transition 2->1

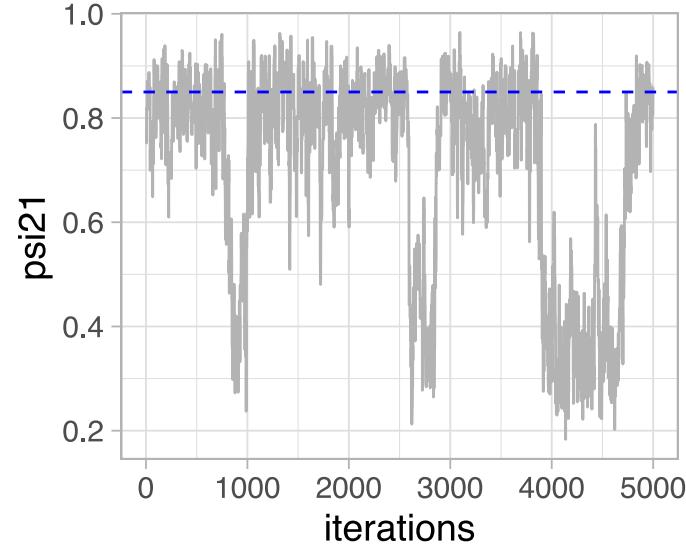
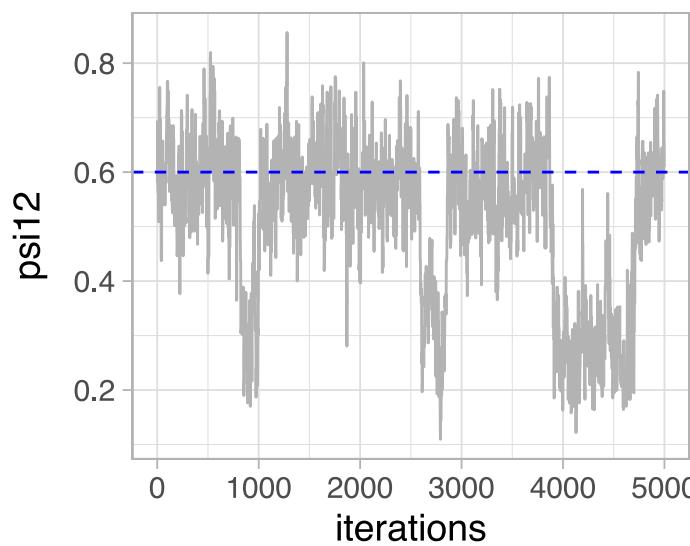
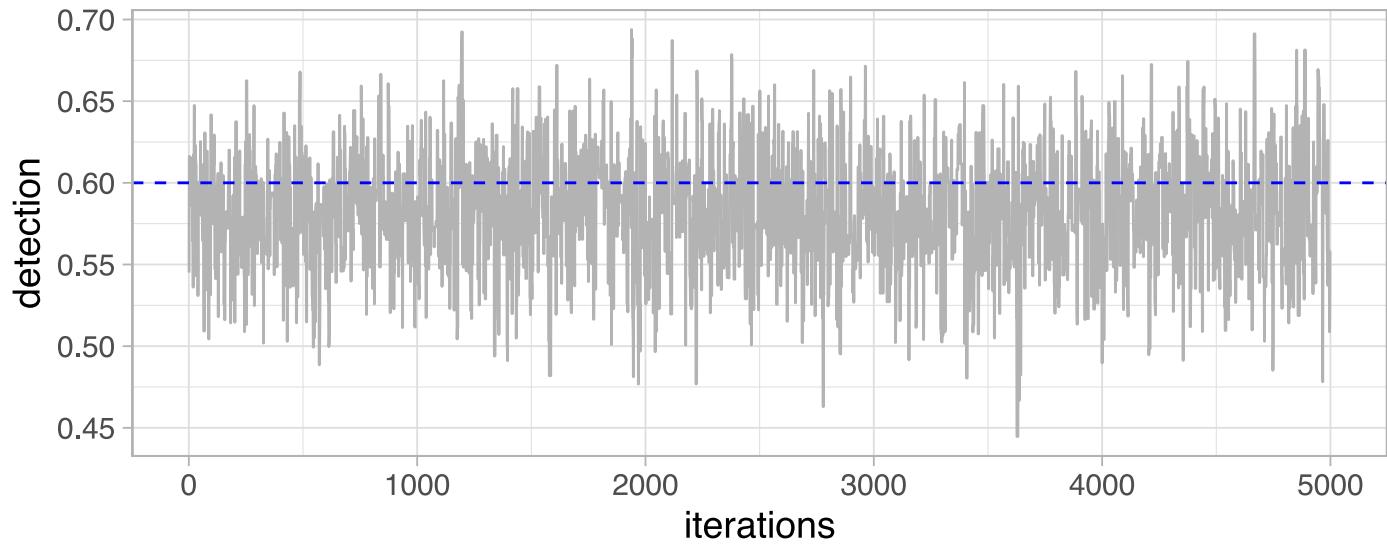


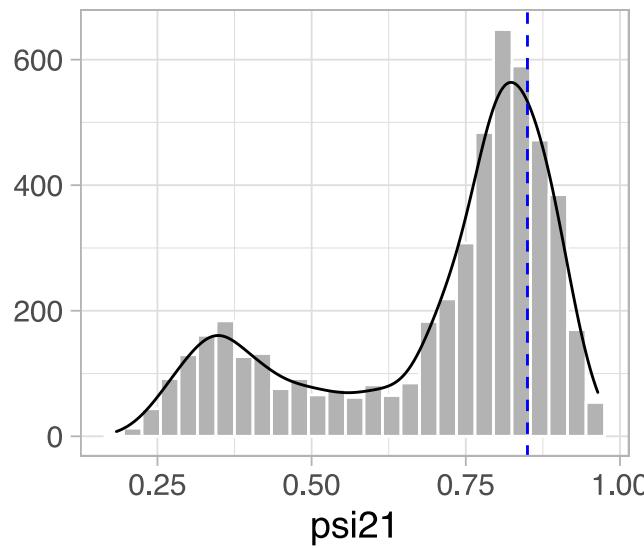
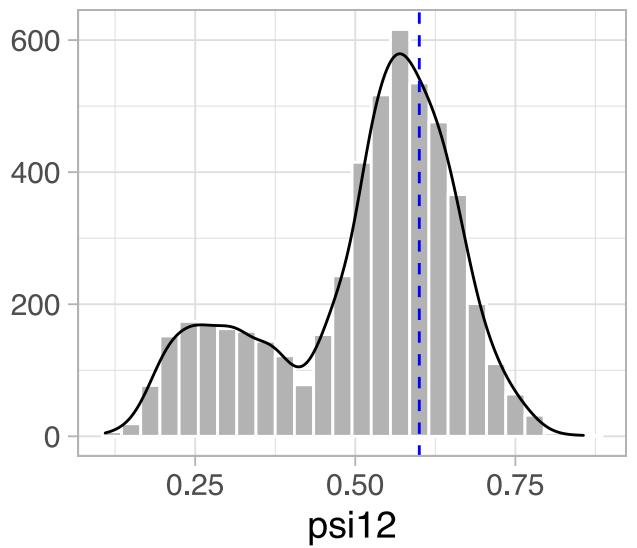
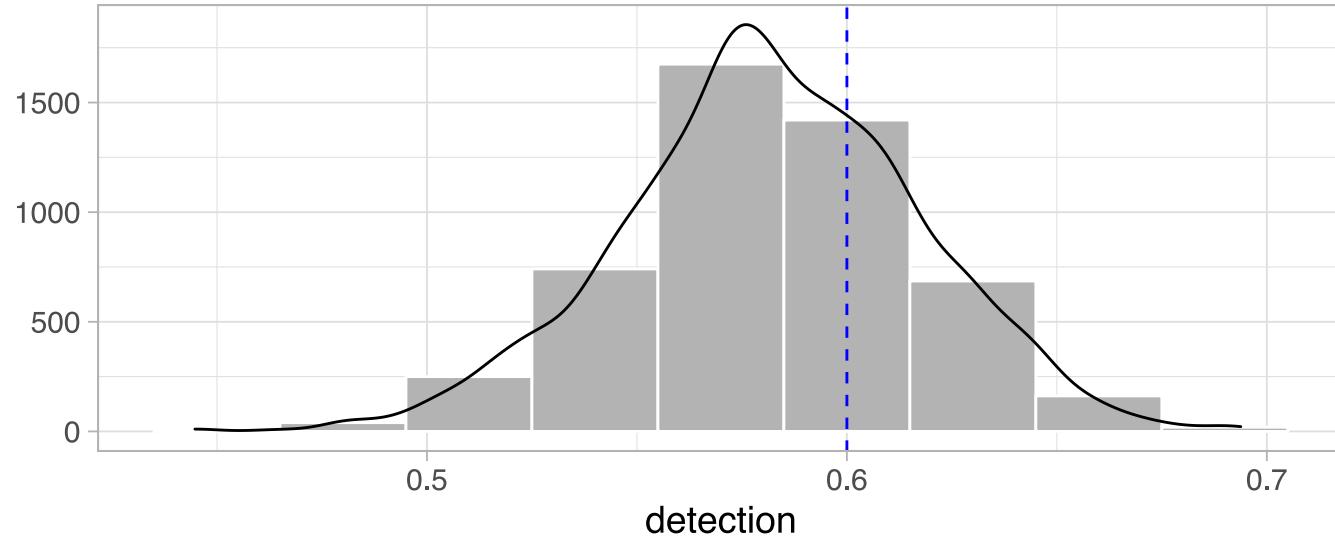
Initial values lead to *local* minimum



Initial values lead to *global* minimum







Further reading

- Lebreton, J.-D., J. D. Nichols, R. J. Barker, R. Pradel and J. A. Spendelow (2009).
Modeling Individual Animal Histories with Multistate Capture–Recapture Models.
Advances in Ecological Research, 41:87–173.

Live demo



Known knowns, unknown knowns and unknowns: Uncertainty in state assignment

The team

last updated: 2021-05-18

Uncertainty in state assignment

Multievent models extend multistate models with uncertainty in state assignment

- Breeding status in female roe deer is ascertained based on fawn detection
- Sex status is ascertained based on morphological criteria in Audouin's gulls
- Disease status in house finches is ascertained based on birds' eyes examination
- Hybrid status in wolves is ascertained based on genetics
- Dominance status in wolves is ascertained based on heterogeneity in detection

We need to explicitly consider state assignment in a model

HMMs to the rescue!

Examples

- Testing life-history trade-offs while accounting for uncertainty in breeding status
- Quantifying disease dynamics while accounting for uncertainty in disease status
- Estimating survival while accounting for individual heterogeneity in detection

Examples

- **Testing life-history trade-offs while accounting for uncertainty in breeding status**
- Quantifying disease dynamics while accounting for uncertainty in disease status
- Estimating survival while accounting for individual heterogeneity in detection

Sooty shearwater (David Boyle)



Uncertainty in breeding status

- 3 states
 - breeding (B)
 - non-breeding (NB)
 - dead (D)
- 4 observations
 - not encountered (0)
 - found, ascertained as breeder (1)
 - found, ascertained as non-breeder (2)
 - found, status unknown (3)

How states generate observations

States

Observations

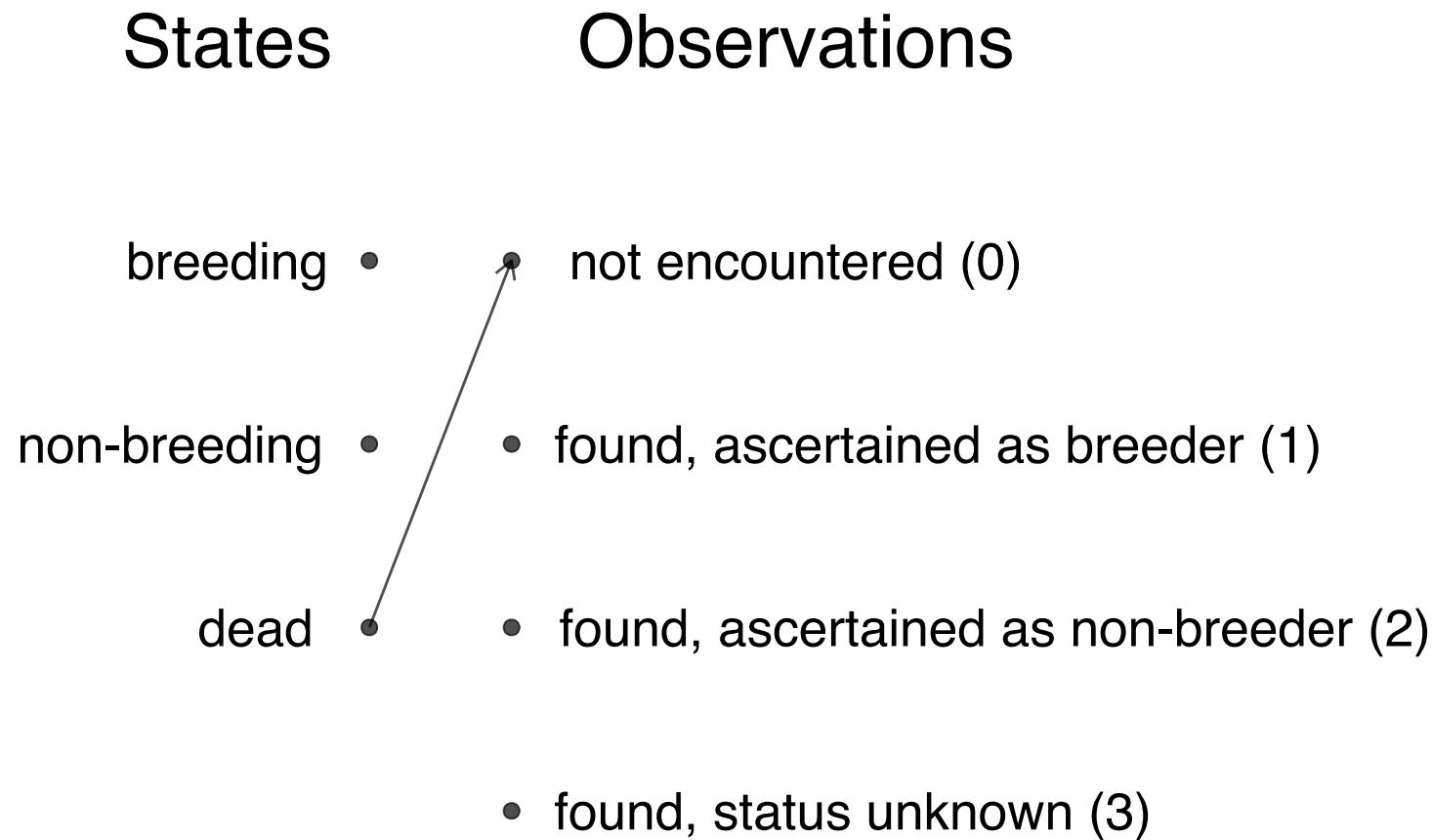
breeding • • not encountered (0)

non-breeding • • found, ascertained as breeder (1)

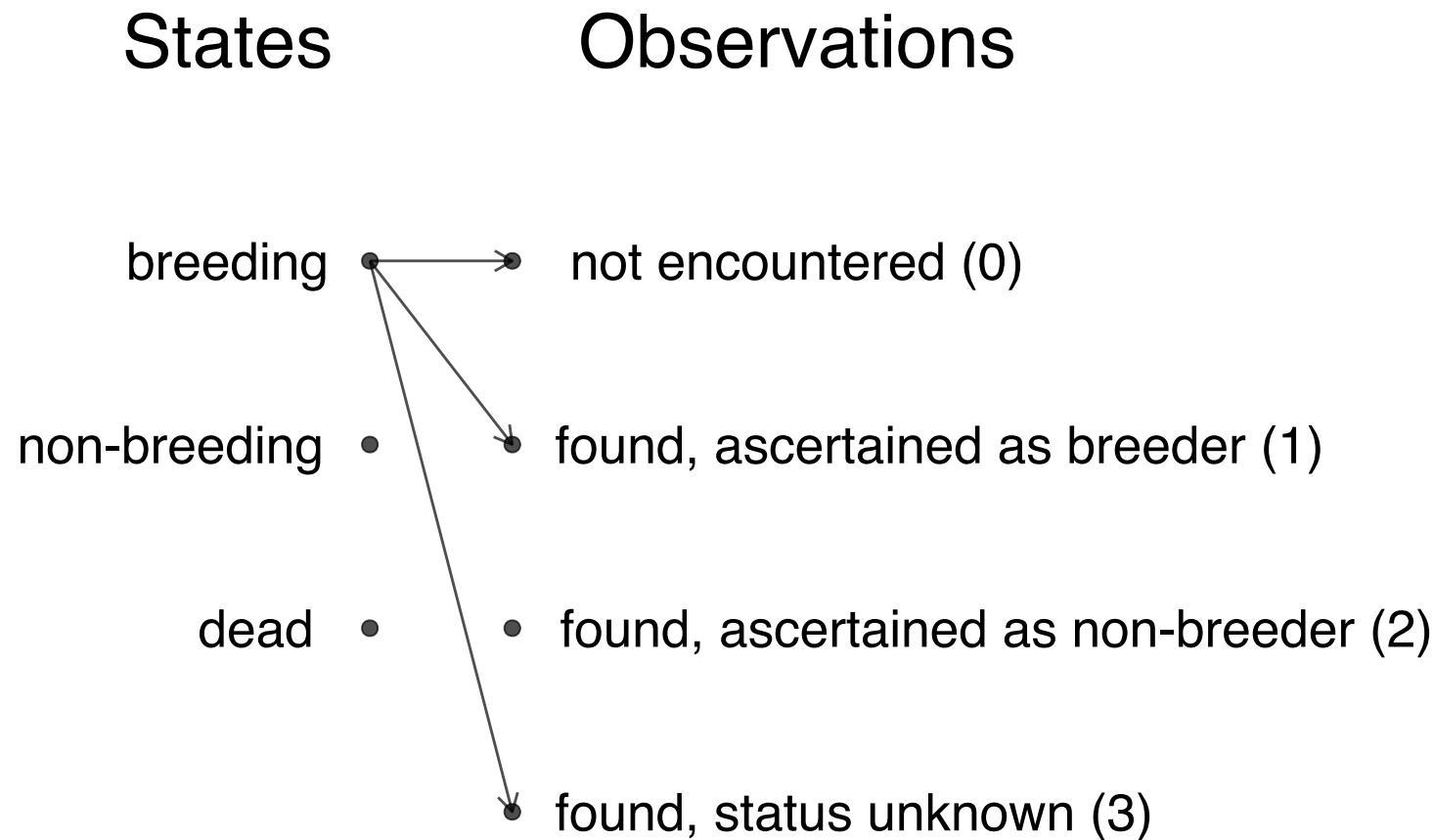
dead • • found, ascertained as non-breeder (2)

 • found, status unknown (3)

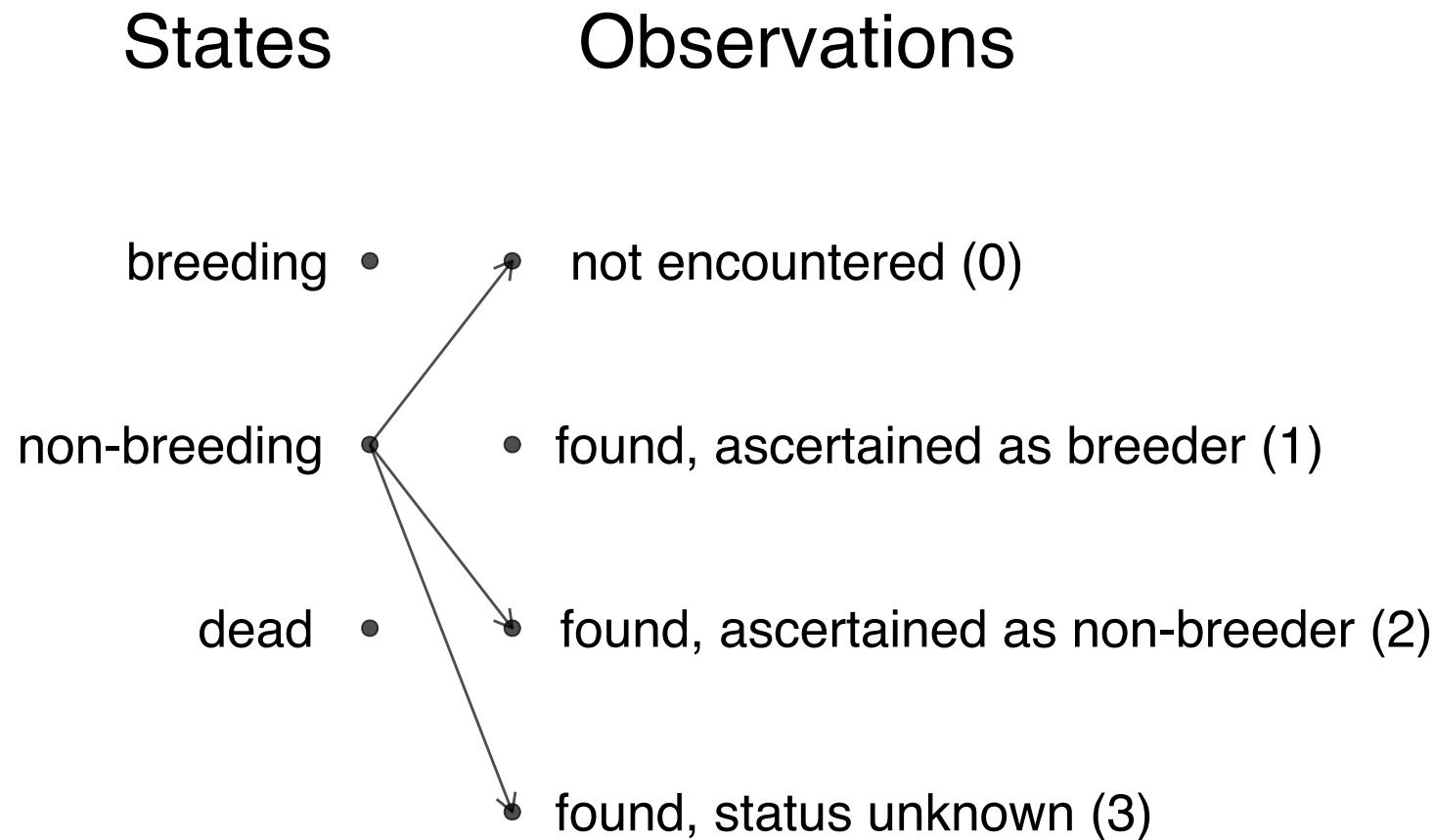
How states generate observations



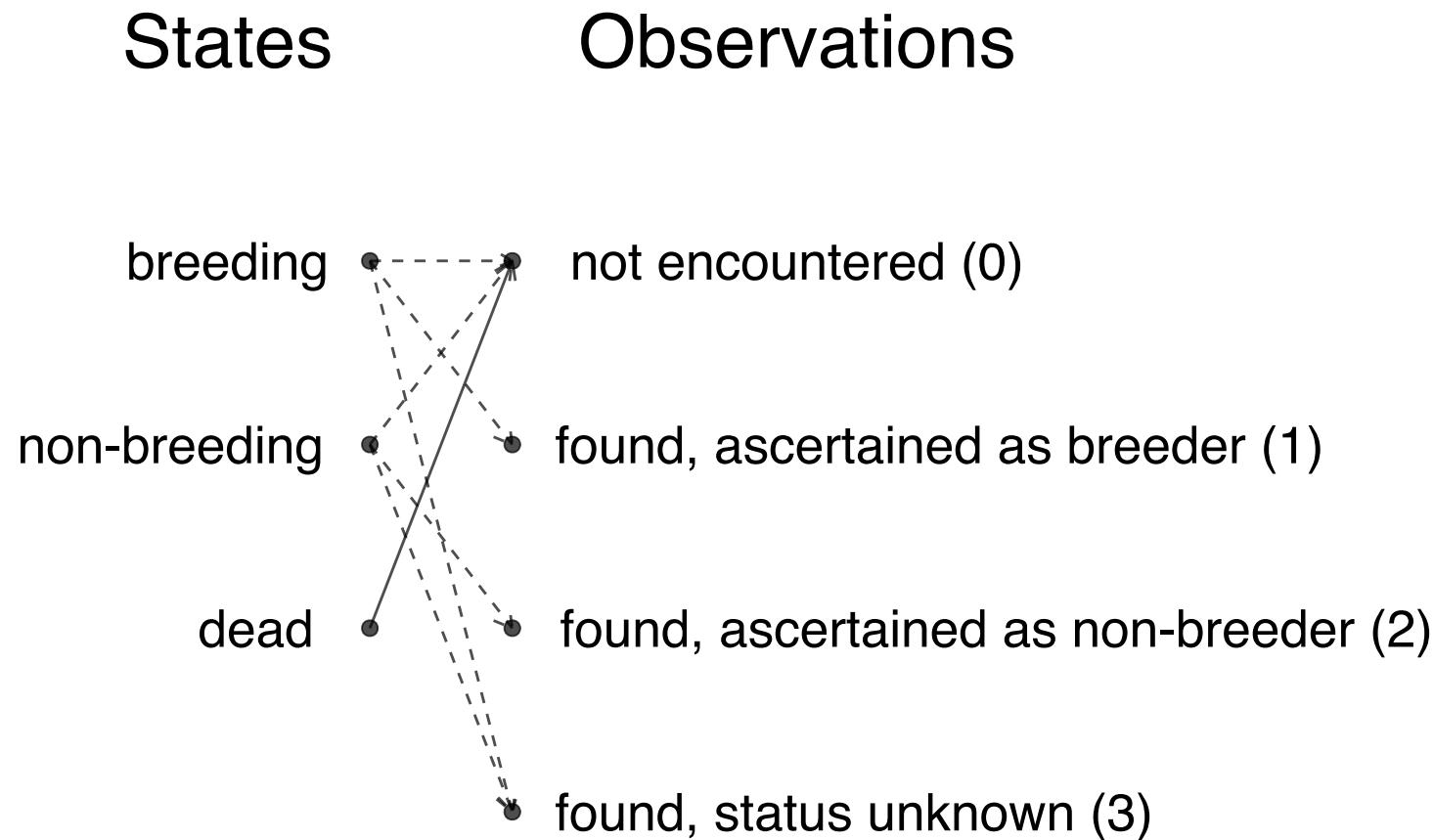
How states generate observations



How states generate observations



How states generate observations



HMM model for breeding states with uncertainty

Vector of initial state probabilities

$$\delta = \left(\begin{array}{ccc} z_t = B & z_t = NB & z_t = D \\ \hline \pi_B & 1 - \pi_B & 0 \end{array} \right)$$

- π_B is the probability that a newly encountered individual is a breeder
- $\pi_{NB} = 1 - \pi_B$ is the probability that a newly encountered individual is a non-breeder

HMM model for breeding states with uncertainty

Transition matrix

$$\boldsymbol{\Gamma} = \begin{array}{ccc|c} & z_t = B & z_t = NB & z_t = D \\ \hline \begin{matrix} z_{t-1} = B \\ z_{t-1} = NB \\ z_{t-1} = D \end{matrix} & \phi_B(1 - \psi_{BNB}) & \phi_B\psi_{BNB} & 1 - \phi_B \\ & \phi_{NB}\psi_{NBB} & \phi_{NB}(1 - \psi_{NBB}) & 1 - \phi_{NB} \\ & 0 & 0 & 1 \end{array}$$

- ϕ_B is breeder survival, ϕ_{NB} that of non-breeders.
- ψ_{BNB} is the probability for an individual breeding a year to be a non-breeder the next year.
- ψ_{NBB} is the probability for an non-breeder individual to breeder the next year.

HMM model for breeding states with uncertainty

Observation matrix

$$\Omega = \begin{array}{cccc} & y_t = 0 & y_t = 1 & y_t = 2 & y_t = 3 \\ \hline & 1 - p_B & p_B \beta_B & 0 & p_B(1 - \beta_B) \\ & 1 - p_{NB} & 0 & p_{NB} \beta_{NB} & p_{NB}(1 - \beta_{NB}) \\ & 1 & 0 & 0 & 0 \end{array} \quad \begin{array}{l} z_t = B \\ z_t = NB \\ z_t = D \end{array}$$

- β_B is the probability to assign an individual in state B to state B.
- β_{NB} is the probability to assign an individual in state NB to state NB.
- p_B is the detection probability of breeders, p_{NB} that of non-breeders.

HMM model for breeding states with uncertainty

Because animals are all captured, $p_B = p_{NB} = 1$ at first encounter:

$$\begin{array}{cccc} y_t = 0 & y_t = 1 & y_t = 2 & y_t = 3 \\ \hline 0 & \beta_B & 0 & (1 - \beta_B) \\ 0 & 0 & \beta_{NB} & (1 - \beta_{NB}) \\ 1 & 0 & 0 & 0 \end{array} \quad \begin{array}{l} z_t = B \\ z_t = NB \\ z_t = D \end{array}$$

Note: Breeding assessment is unaffected.

Our model $(\phi_B, \phi_{NB}, \psi_{BNB}, \psi_{NBB}, p_B, p_{NB}, \beta_B, \beta_{NB}, \pi)$

```
multievent <- nimbleCode({
  # -----
  # Parameters:
  # phiB: survival probability state B
  # phiNB: survival probability state NB
  # psiBNB: transition probability from B to NB
  # psiNBB: transition probability from NB to B
  # pB: recapture probability B
  # pNB: recapture probability NB
  # piB prob. of being in initial state breeder
  # betaNB prob to ascertain the breeding status of an individual encountered as non-breeder
  # betaB prob to ascertain the breeding status of an individual encountered as breeder
  # -----
  # States (z):
  # 1 alive B
  # 2 alive NB
  # 3 dead
  # Observations (y):
  # 1 = non-detected
  # 2 = seen and ascertained as breeder
  # 3 = seen and ascertained as non-breeder
  # 4 = not ascertained
  # -----
```

Our model $(\phi_B, \phi_{NB}, \psi_{B|NB}, \psi_{N|BB}, p_B, p_{NB}, \beta_B, \beta_{NB}, \pi)$

```
multievent <- nimbleCode( {  
  ...  
  # Priors  
  phiB ~ dunif(0, 1)  
  phiNB ~ dunif(0, 1)  
  psiBNB ~ dunif(0, 1)  
  psiNBB ~ dunif(0, 1)  
  pB ~ dunif(0, 1)  
  pNB ~ dunif(0, 1)  
  piB ~ dunif(0, 1)  
  betaNB ~ dunif(0, 1)  
  betaB ~ dunif(0, 1)  
  ...  
}
```

Our model $(\phi_B, \phi_{NB}, \psi_{B|NB}, \psi_{NBB}, p_B, p_{NB}, \beta_B, \beta_{NB}, \pi)$

```
multievent <- nimbleCode( {  
  ...  
  # vector of initial stats probs  
  delta[1] <- piB # prob. of being in initial state B  
  delta[2] <- 1 - piB # prob. of being in initial state NB  
  delta[3] <- 0 # prob. of being in initial state dead  
  ...  
}
```

Our model $(\phi_B, \phi_{NB}, \psi_{BNB}, \psi_{NBB}, p_B, p_{NB}, \beta_B, \beta_{NB}, \pi)$

```
multievent <- nimbleCode({  
  ...  
  # probabilities of state  $z(t+1)$  given  $z(t)$   
  gamma[1,1] <- phiB * (1 - psiBNB)  
  gamma[1,2] <- phiB * psiBNB  
  gamma[1,3] <- 1 - phiB  
  gamma[2,1] <- phiNB * psiNBB  
  gamma[2,2] <- phiNB * (1 - psiNBB)  
  gamma[2,3] <- 1 - phiNB  
  gamma[3,1] <- 0  
  gamma[3,2] <- 0  
  gamma[3,3] <- 1  
  ...
```

Our model $(\phi_B, \phi_{NB}, \psi_{B|NB}, \psi_{NBB}, p_B, p_{NB}, \beta_B, \beta_{NB}, \pi)$

```
multievent <- nimbleCode({  
  ...  
  # probabilities of  $y(t)$  given  $z(t)$   
  omega[1,1] <- 1 - pB          #  $Pr(\text{alive } B \text{ } t \rightarrow \text{non-detected } t)$   
  omega[1,2] <- pB * betaB      #  $Pr(\text{alive } B \text{ } t \rightarrow \text{detected } B \text{ } t)$   
  omega[1,3] <- 0                #  $Pr(\text{alive } B \text{ } t \rightarrow \text{detected } NB \text{ } t)$   
  omega[1,4] <- pB * (1 - betaB) #  $Pr(\text{alive } B \text{ } t \rightarrow \text{detected } U \text{ } t)$   
  omega[2,1] <- 1 - pNB         #  $Pr(\text{alive } NB \text{ } t \rightarrow \text{non-detected } t)$   
  omega[2,2] <- 0                #  $Pr(\text{alive } NB \text{ } t \rightarrow \text{detected } B \text{ } t)$   
  omega[2,3] <- pNB * betaNB    #  $Pr(\text{alive } NB \text{ } t \rightarrow \text{detected } NB \text{ } t)$   
  omega[2,4] <- pNB * (1 - betaNB) #  $Pr(\text{alive } NB \text{ } t \rightarrow \text{detected } U \text{ } t)$   
  omega[3,1] <- 1                #  $Pr(\text{dead } t \rightarrow \text{non-detected } t)$   
  omega[3,2] <- 0                #  $Pr(\text{dead } t \rightarrow \text{detected } N \text{ } t)$   
  omega[3,3] <- 0                #  $Pr(\text{dead } t \rightarrow \text{detected } NB \text{ } t)$   
  omega[3,4] <- 0                #  $Pr(\text{dead } t \rightarrow \text{detected } U \text{ } t)$   
  ...
```

Our model $(\phi_B, \phi_{NB}, \psi_{BNB}, \psi_{NBB}, p_B, p_{NB}, \beta_B, \beta_{NB}, \pi)$

```
multievent <- nimbleCode({  
  ...  
  # probabilities of y(first) given z(first)  
  omega.init[1,1] <- 0          # Pr(alive B t = 1 -> non-detected t = 1)  
  omega.init[1,2] <- betaB      # Pr(alive B t = 1 -> detected B t = 1)  
  omega.init[1,3] <- 0          # Pr(alive B t = 1 -> detected NB t = 1)  
  omega.init[1,4] <- 1 - betaB  # Pr(alive B t = 1 -> detected U t = 1)  
  omega.init[2,1] <- 0          # Pr(alive NB t = 1 -> non-detected t = 1)  
  omega.init[2,2] <- 0          # Pr(alive NB t = 1 -> detected B t = 1)  
  omega.init[2,3] <- betaNB    # Pr(alive NB t = 1 -> detected NB t = 1)  
  omega.init[2,4] <- 1 - betaNB # Pr(alive NB t = 1 -> detected U t = 1)  
  omega.init[3,1] <- 1          # Pr(dead t = 1 -> non-detected t = 1)  
  omega.init[3,2] <- 0          # Pr(dead t = 1 -> detected N t = 1)  
  omega.init[3,3] <- 0          # Pr(dead t = 1 -> detected NB t = 1)  
  omega.init[3,4] <- 0          # Pr(dead t = 1 -> detected U t = 1)  
  ...  
})
```

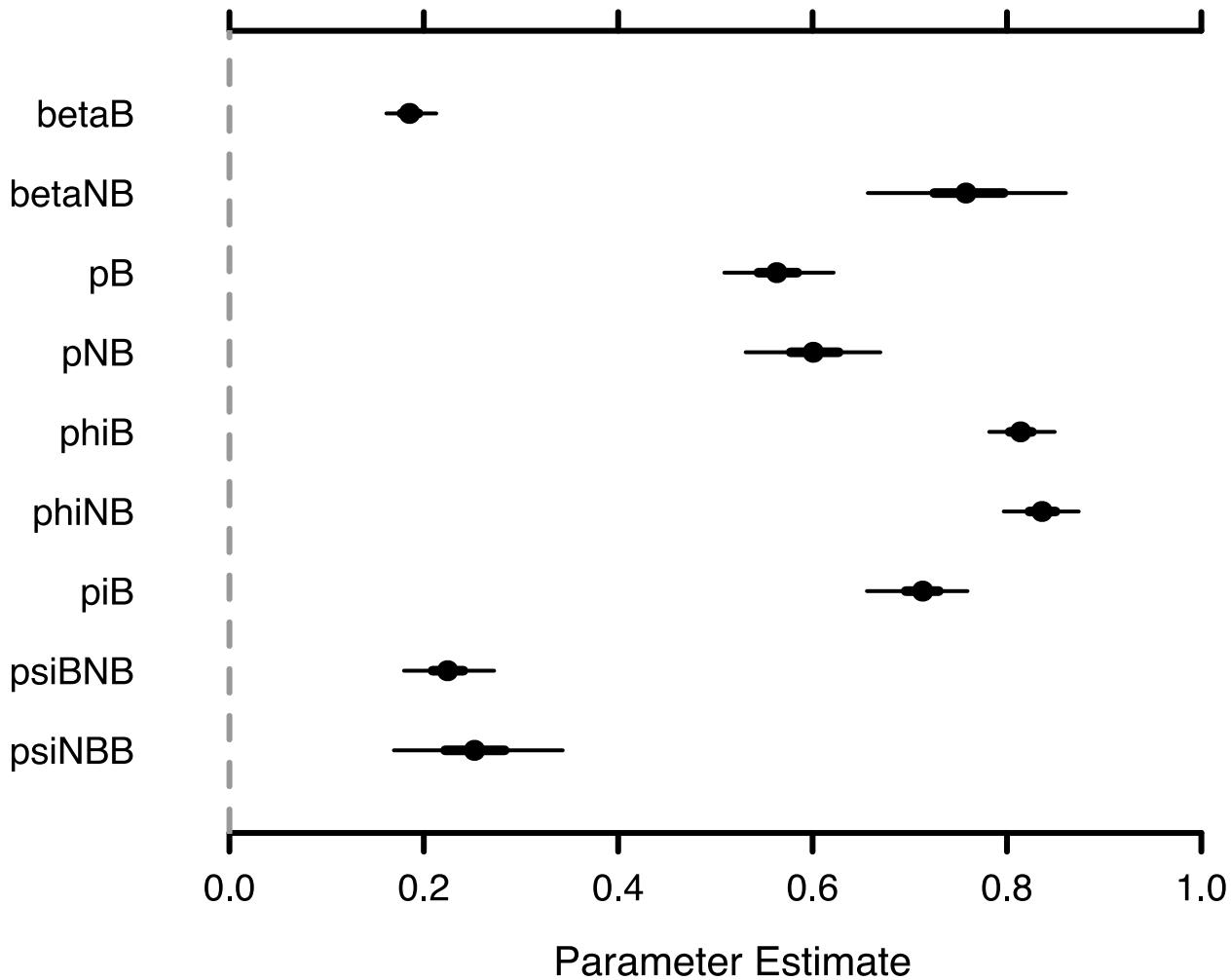
Our model $(\phi_B, \phi_{NB}, \psi_{B|NB}, \psi_{NBB}, p_B, p_{NB}, \beta_B, \beta_{NB}, \pi)$

```
multievent <- nimbleCode( {
  ...
  # likelihood
  for (i in 1:N){
    # latent state at first capture
    z[i,first[i]] ~ dcat(delta[1:3])
    y[i,first[i]] ~ dcat(omega.init[z[i,first[i]],1:4])
    for (t in (first[i]+1):K){
      # z(t) given z(t-1)
      z[i,t] ~ dcat(gamma[z[i,t-1],1:3])
      # y(t) given z(t)
      y[i,t] ~ dcat(omega[z[i,t],1:4])
    }
  }
})
```

Results

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
betaB	0.19	0.01	0.16	0.19	0.21	1.01	332
betaNB	0.76	0.05	0.66	0.76	0.86	1.01	65
pB	0.56	0.03	0.51	0.56	0.62	1.06	229
pNB	0.60	0.04	0.53	0.60	0.67	1.03	142
phiB	0.81	0.02	0.78	0.81	0.85	1.01	312
phiNB	0.84	0.02	0.80	0.84	0.87	1.00	354
piB	0.71	0.03	0.66	0.71	0.76	1.02	115
psiBNB	0.23	0.02	0.18	0.22	0.27	1.00	214
psiNBB	0.25	0.04	0.17	0.25	0.34	1.00	95

- Breeders are difficult to assigned to the correct state.
- Non-breeders are relatively well classified as non-breeders.
- No cost of breeding, neither on survival, nor on future reproduction.



Live demo



Examples

- Testing life-history trade-offs while accounting for uncertainty in breeding status
- **Quantifying disease dynamics while accounting for uncertainty in disease status**
- Estimating survival while accounting for individual heterogeneity in detection

Animal epidemiology with uncertain disease states

- We consider a system of an emerging pathogen *Mycoplasma gallisepticum* Edward and Kanarek and its host the house finch, *Carpodacus mexicanus* Müller.

A house finch with a heavy infection (Jim Mondok).



Animal epidemiology with uncertain disease states

- We consider a system of an emerging pathogen *Mycoplasma gallisepticum* Edward and Kanarek and its host the house finch, *Carpodacus mexicanus* Müller.
- Faustino et al. (2004) and Conn & Cooch (2009) studied impact of pathogen on host demographic rates.
- Problem is true disease state for some encountered individuals is ambiguous because seen at distance.
- In this context, how to study the dynamics of the disease?

States and observations

- 3 states
 - healthy (H)
 - ill (I)
 - dead (D)
- 4 observations
 - not seen (0)
 - captured healthy (1)
 - captured ill (2)
 - health status unknown, i.e. seen at distance (3)

How states generate observations.

States

healthy •

ill •

dead •

Observations

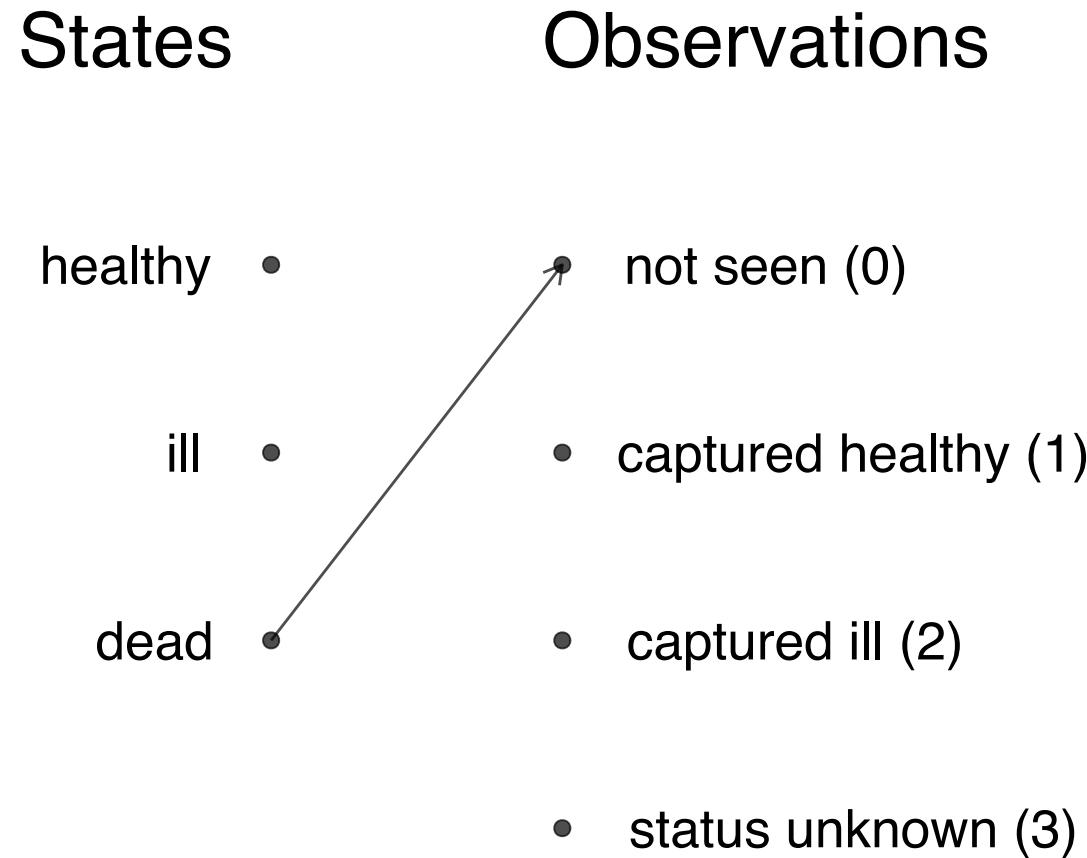
- not seen (0)

- captured healthy (1)

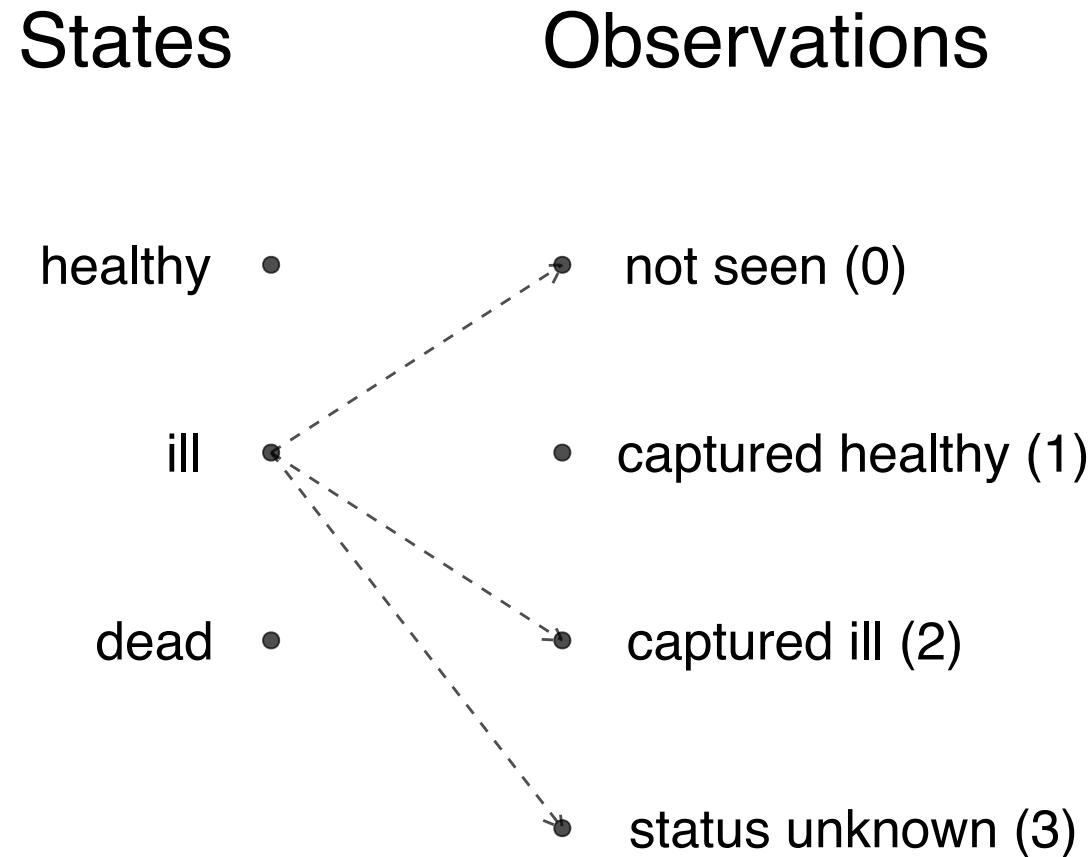
- captured ill (2)

- status unknown (3)

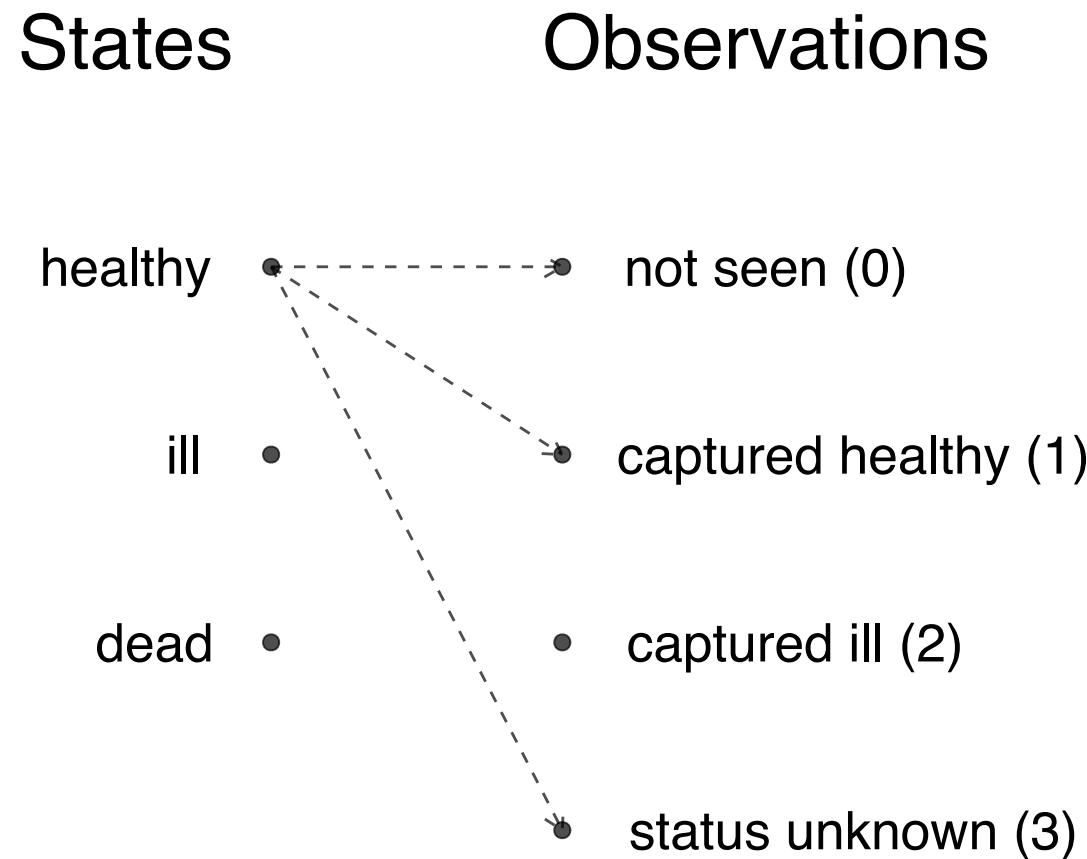
How states generate observations.



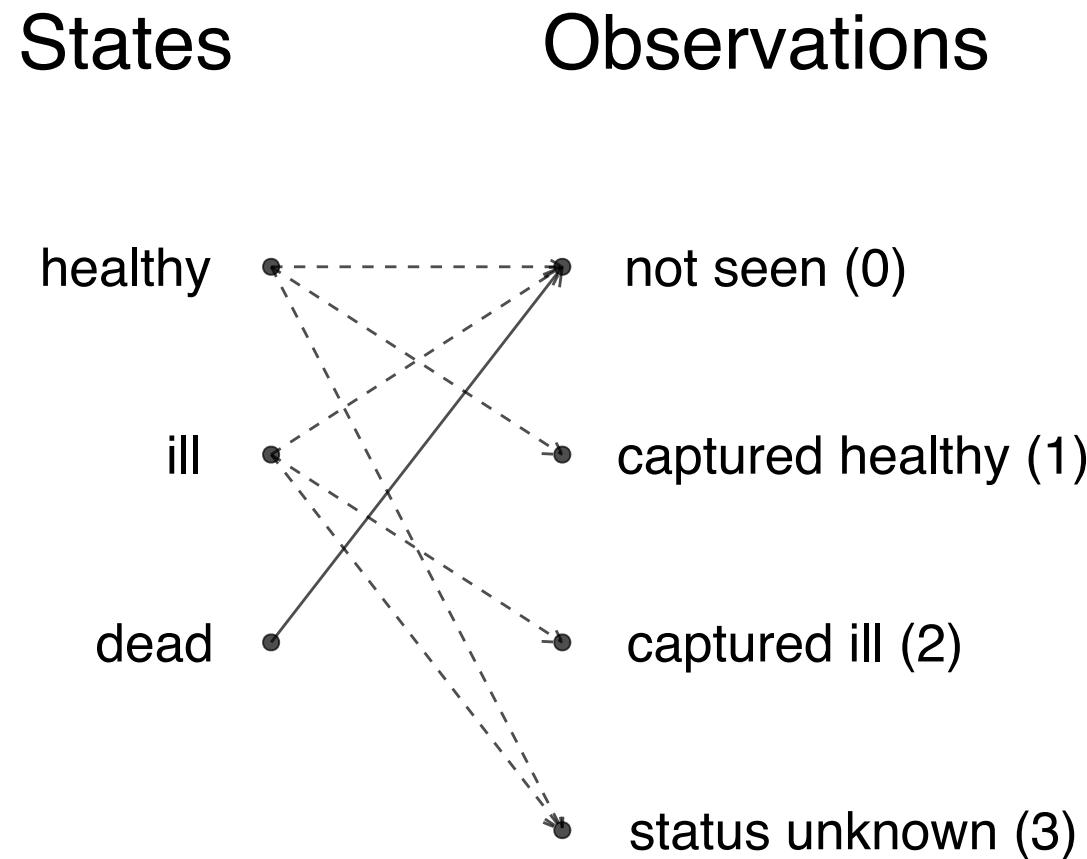
How states generate observations.



How states generate observations.



How states generate observations.



HMM model for disease states with uncertainty

Vector of initial state probabilities

$$\delta = \left(\begin{array}{ccc} z_t = H & z_t = I & z_t = D \\ \hline \pi_H & 1 - \pi_H & 0 \end{array} \right)$$

- π_H is the probability that a newly encountered individual is healthy.
- $\pi_I = 1 - \pi_H$ is the probability that a newly encountered individual is ill.

HMM model for disease states with uncertainty

Transition matrix

$$\boldsymbol{\Gamma} = \begin{array}{ccc} & z_t = H & z_t = I & z_t = D \\ \hline & \phi_H(1 - \psi_{HI}) & \phi_H\psi_{HI} & 1 - \phi_H \\ & \phi_I\psi_{IH} & \phi_I(1 - \psi_{IH}) & 1 - \phi_I \\ & 0 & 0 & 1 \end{array} \quad \begin{array}{l} z_{t-1} = H \\ z_{t-1} = I \\ z_{t-1} = D \end{array}$$

- ϕ_H is the survival probability of healthy individuals, ϕ_I that of ill individuals.
- ψ_{HI} is the probability of getting sick, ψ_{IH} that of recovering from the disease.

HMM model for disease states with uncertainty

Transition matrix, incurable disease

$$\boldsymbol{\Gamma} = \begin{array}{ccc} & z_t = H & z_t = I & z_t = D \\ \hline & \phi_H(1 - \psi_{HI}) & \phi_H\psi_{HI} & 1 - \phi_H \\ & 0 & \phi_I & 1 - \phi_I \\ & 0 & 0 & 1 \end{array} \quad \begin{array}{l} z_{t-1} = H \\ z_{t-1} = I \\ z_{t-1} = D \end{array}$$

- No possibility of recovering from the disease, that is $\psi_{IH} = 0$. Once you get sick, you remain sick $\psi_{II} = 1 - \psi_{IH} = 1$.
- For analysing the house finch data, we allow recovering from the disease, and we will use transition matrix from previous slide.

HMM model for disease states with uncertainty

Observation matrix

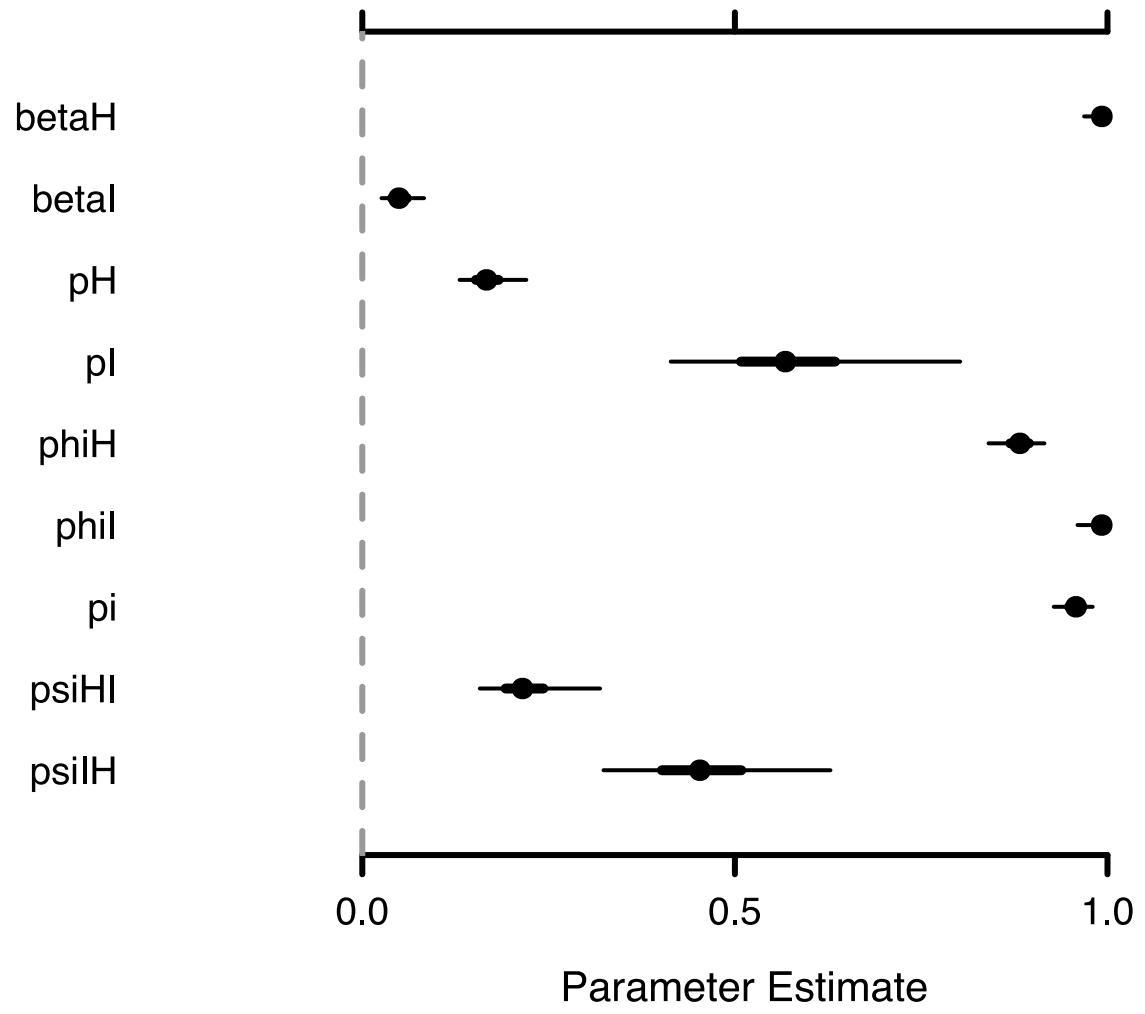
$$\Omega = \begin{array}{cccc|c} & y_t = 0 & y_t = 1 & y_t = 2 & y_t = 3 & \\ \hline & 1 - p_H & p_H \beta_H & 0 & p_H(1 - \beta_H) & z_t = H \\ & 1 - p_I & 0 & p_I \beta_I & p_I(1 - \beta_I) & z_t = I \\ & 1 & 0 & 0 & 0 & z_t = D \end{array}$$

- β_H is the probability to assign a healthy individual to state H.
- β_I is the probability to assign a sick individual to state I.
- p_H is the detection probability of healthy individuals, p_I that of sick individuals.

Results

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
betaH	0.99	0.01	0.97	0.99	1.00	1.01	1421
betaI	0.05	0.01	0.03	0.05	0.08	1.00	6477
pH	0.17	0.02	0.13	0.17	0.22	1.01	331
pI	0.58	0.10	0.41	0.57	0.80	1.04	220
phiH	0.88	0.02	0.84	0.88	0.92	1.01	360
phiI	0.99	0.01	0.96	0.99	1.00	1.00	1004
pi	0.96	0.01	0.93	0.96	0.98	1.00	4190
psiHI	0.22	0.04	0.16	0.22	0.32	1.02	311
psiIH	0.46	0.08	0.32	0.45	0.63	1.02	392

- Healthy individuals are correctly assigned, while infected individuals are difficult to ascertain.
- Sounds like being infected has an effect on detection and survival. Run models without effects and compare with WAIC for formal testing.
- Infection rate is 22%, recovery rate is 46%.



Live demo



Examples

- Testing life-history trade-offs while accounting for uncertainty in breeding status
- Quantifying disease dynamics while accounting for uncertainty in disease status
- **Estimating survival while accounting for individual heterogeneity in detection**

Individual heterogeneity with finite mixtures.

- Gray wolf is a social species with hierarchy in packs which may reflect in species demography.



Individual heterogeneity with finite mixtures.

- Gray wolf is a social species with hierarchy in packs which may reflect in demography.
- Shirley Pledger in a series of papers developed heterogeneity models in which individuals are assigned in two or more classes with class-specific survival/detection probabilities.
- Cubaynes et al. (2010) used HMMs to account for heterogeneity in the detection process due to social status, see also Pradel et al. (2009).

Individual heterogeneity

- 3 states
 - alive in class 1 (A1)
 - alive in class 2 (A2)
 - dead (D)
- 4 observations
 - not captured (0)
 - captured (1)

HMM model for individual heterogeneity

Vector of initial state probabilities

$$\delta = \left(\begin{array}{ccc} z_t = A1 & z_t = A2 & z_t = D \\ \hline \pi & 1 - \pi & 0 \end{array} \right)$$

- π is the probability of being alive in class 1.
- $1 - \pi$ is the probability of being in class 2.

HMM model for individual heterogeneity

Transition matrix

$$\boldsymbol{\Gamma} = \left(\begin{array}{ccc} z_t = A1 & z_t = A2 & z_t = D \\ \hline \phi & 0 & 1 - \phi \\ 0 & \phi & 1 - \phi \\ 0 & 0 & 1 \end{array} \right) \begin{array}{l} z_{t-1} = A1 \\ z_{t-1} = A2 \\ z_{t-1} = D \end{array}$$

- ϕ is the survival probability, which could be made heterogeneous.

HMM model for individual heterogeneity

Transition matrix, with change in heterogeneity class

$$\boldsymbol{\Gamma} = \begin{array}{ccc} z_t = A1 & z_t = A2 & z_t = D \\ \hline \phi(1 - \psi_{12}) & \phi\psi_{12} & 1 - \phi \\ \phi\psi_{21} & \phi(1 - \psi_{21}) & 1 - \phi \\ 0 & 0 & 1 \end{array} \begin{array}{l} z_{t-1} = A1 \\ z_{t-1} = A2 \\ z_{t-1} = D \end{array}$$

- ψ_{12} is the probability for an individual to change class of heterogeneity, from 1 to 2.
- ψ_{21} is the probability for an individual to change class of heterogeneity, from 2 to 1.

HMM model for individual heterogeneity

Observation matrix

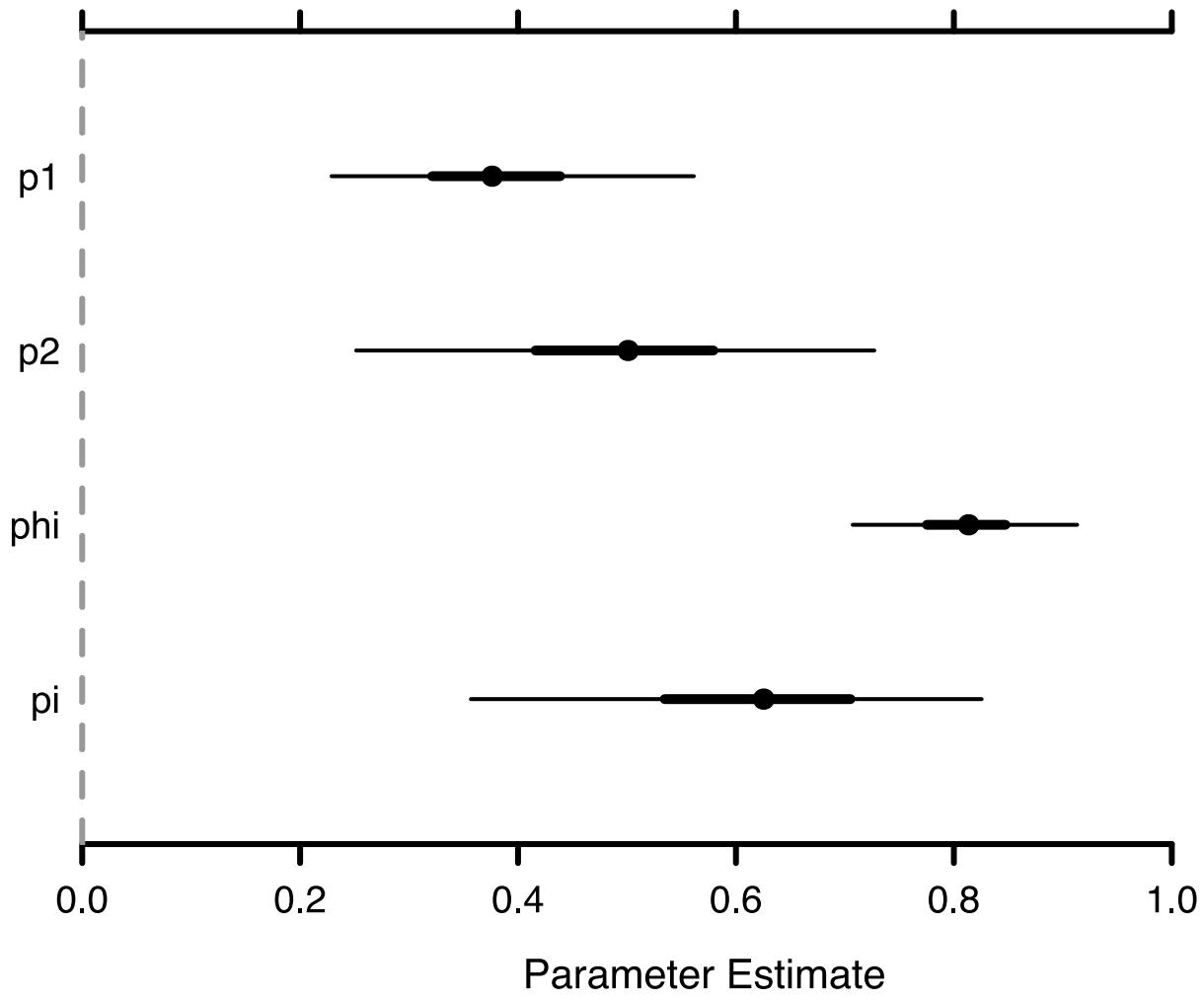
$$\Omega = \begin{array}{cc} & \begin{matrix} y_t = 0 & y_t = 1 \end{matrix} \\ \hline & \begin{matrix} 1 - p_1 & p_1 \\ 1 - p_2 & p_2 \\ 1 & 0 \end{matrix} \end{array} \begin{array}{l} z_t = A1 \\ z_t = A2 \\ z_t = D \end{array}$$

- p_1 is detection for individuals in class 1, and p_2 that of individuals in class 2.

Results

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
p1	0.38	0.09	0.23	0.38	0.56	1.04	210
p2	0.50	0.12	0.25	0.50	0.73	1.01	229
phi	0.81	0.05	0.71	0.81	0.91	1.04	317
pi	0.62	0.12	0.36	0.63	0.83	1.02	164

- We have lowly detectable individuals (class A1 with p_1) in proportion 62%.
- And highly (or so) detectable individuals (class A2 with p_2) in proportion 38%.
- Note that interpretation of classes is made a posteriori.
- Survival is 81%.



HMM model for individual heterogeneity

- You may consider more classes, and select among models, see [Cubaynes et al. \(2012\)](#).
- You may also go for a non-parametric approach and let the data tell you how many classes you need. This is relatively easy to do in Nimble, see [Turek et al. \(2021\)](#).
- More about individual heterogeneity in [Gimenez et al. \(2018\)](#).

HMMs to analyse capture-recapture data

With the same data, ask further questions, just consider different states.



THE ONLY LIMIT IS YOUR IMAGINATION



PEW PEW!

How to make our models remember?

- So far, the dynamics of the states are first-order Makovian.
- The site where you will be depends only on the site where you are, and not on the sites you were previously.
- How to relax this assumption, and go second-order Markovian?
- Memory models were initially proposed by Hestbeck et al. (1991) and Brownie et al. (1993), then formulated as HMMs in Rouan et al. (2009). See also Cole et al. (2014).

Remember HMM model for dispersal between 2 sites

Transition matrix

$$\boldsymbol{\Gamma} = \begin{array}{ccc} z_t = A & z_t = B & z_t = D \\ \hline \phi_A(1 - \psi_{AB}) & \phi_A\psi_{AB} & 1 - \phi_A \\ \phi_B\psi_{BA} & \phi_B(1 - \psi_{BA}) & 1 - \phi_B \\ 0 & 0 & 1 \end{array} \quad \begin{array}{l} z_{t-1} = A \\ z_{t-1} = B \\ z_{t-1} = D \end{array}$$

Observation matrix

$$\boldsymbol{\Omega} = \begin{array}{ccc} y_t = 0 & y_t = 1 & y_t = 2 \\ \hline 1 - p_A & p_A & 0 \\ 1 - p_B & 0 & p_B \\ 1 & 0 & 0 \end{array} \quad \begin{array}{l} z_t = A \\ z_t = B \\ z_t = D \end{array}$$

HMM formulation of the memory model

- To keep track of the sites previously visited, the trick is to consider states as being pairs of sites occupied
- States
 - AA is for alive in site A at t and alive in site A at $t - 1$
 - AB is for alive in site A at t and alive in site B at $t - 1$
 - BA is for alive in site B at t and alive in site A at $t - 1$
 - BB is for alive in site B at t and alive in site B at $t - 1$
 - D is for dead
- Observations
 - 0 not captured
 - 1 captured at site A
 - 2 captured at site B

HMM formulation of the memory model

Vector of initial state probabilities

$$\delta = \left(\begin{array}{ccccc} z_t = AA & z_t = AB & z_t = BA & z_t = BB & z_t = D \\ \hline \pi_{AA} & \pi_{AB} & \pi_{BA} & \pi_{BB} & 0 \end{array} \right)$$

- where $\pi_{BB} = 1 - (\pi_{AA} + \pi_{AB} + \pi_{BA})$,
- and π_{ij} at site j when first captured at t and site i at $t - 1$.

HMM formulation of the memory model

Transition matrix

$$\Gamma = \begin{pmatrix} z_t = AA & z_t = AB & z_t = BA & z_t = BB & z_t = D \\ \hline \phi_{AAA} & \phi_{AAB} & 0 & 0 & 1 - \phi_{AAA} - \phi_{AAB} \\ 0 & 0 & \phi_{ABA} & \phi_{ABB} & 1 - \phi_{ABA} - \phi_{ABB} \\ \phi_{BAA} & \phi_{BAB} & 0 & 0 & 1 - \phi_{BAA} - \phi_{BAB} \\ 0 & 0 & \phi_{BBA} & \phi_{BBB} & 1 - \phi_{BBA} - \phi_{BBB} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{array}{l} z_t = AA \\ z_t = AB \\ z_t = BA \\ z_t = BB \\ z_t = D \end{array}$$

- ϕ_{ijk} is probability to be in site k at time $t + 1$ for an individual present in site j at t and in site i at $t - 1$

HMM formulation of the memory model

Transition matrix, alternate parameterization

$$\Gamma = \begin{pmatrix} z_t = AA & z_t = AB & z_t = BA & z_t = BB & z_t = D \\ \hline \phi\psi_{AAA} & \phi(1 - \psi_{AAA}) & 0 & 0 & 1 - \phi \\ 0 & 0 & \phi(1 - \psi_{ABB}) & \phi\psi_{ABB} & 1 - \phi \\ \phi\psi_{BAA} & \phi(1 - \psi_{BAA}) & 0 & 0 & 1 - \phi \\ 0 & 0 & \phi(1 - \psi_{BBB}) & \phi\psi_{BBB} & 1 - \phi \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{array}{l} z_t = AA \\ z_t = AB \\ z_t = BA \\ z_t = BB \\ z_t = D \end{array}$$

- ϕ is the probability of surviving from one occasion to the next.
- ψ_{ijj} is the probability an animal stays at the same site j given that it was at site i on the previous occasion.

HMM formulation of the memory model

Observation matrix

$$\Omega = \begin{array}{c|ccc} & y_t = 0 & y_t = 1 & y_t = 2 \\ \hline & 1 - p_A & p_A & 0 \\ & 1 - p_B & 0 & p_B \\ \Omega = & 1 - p_A & p_A & 0 \\ & 1 - p_B & 0 & p_B \\ & 1 & 0 & 0 \end{array} \quad \begin{array}{l} z_t = AA \\ z_t = AB \\ z_t = BA \\ z_t = BB \\ z_t = D \end{array}$$

Further reading

- Seminal paper by Pradel (2005) **Multievent: An Extension of Multistate Capture-Recapture Models to Uncertain States**. *Biometrics*, 61: 442-447.
- Dupuis (1995) had a similar idea for the Arnason-Schwarz model: Dupuis, J. (1995) **Bayesian estimation of movement and survival probabilities from capture-recapture data**. *Biometrika*. Vol. 82, pp 761-772.
- See also for a review Gimenez et al. (2012) **Estimating demographic parameters using hidden process dynamic models**. *Theoretical Population Biology* 82: 307-316.

Live demo

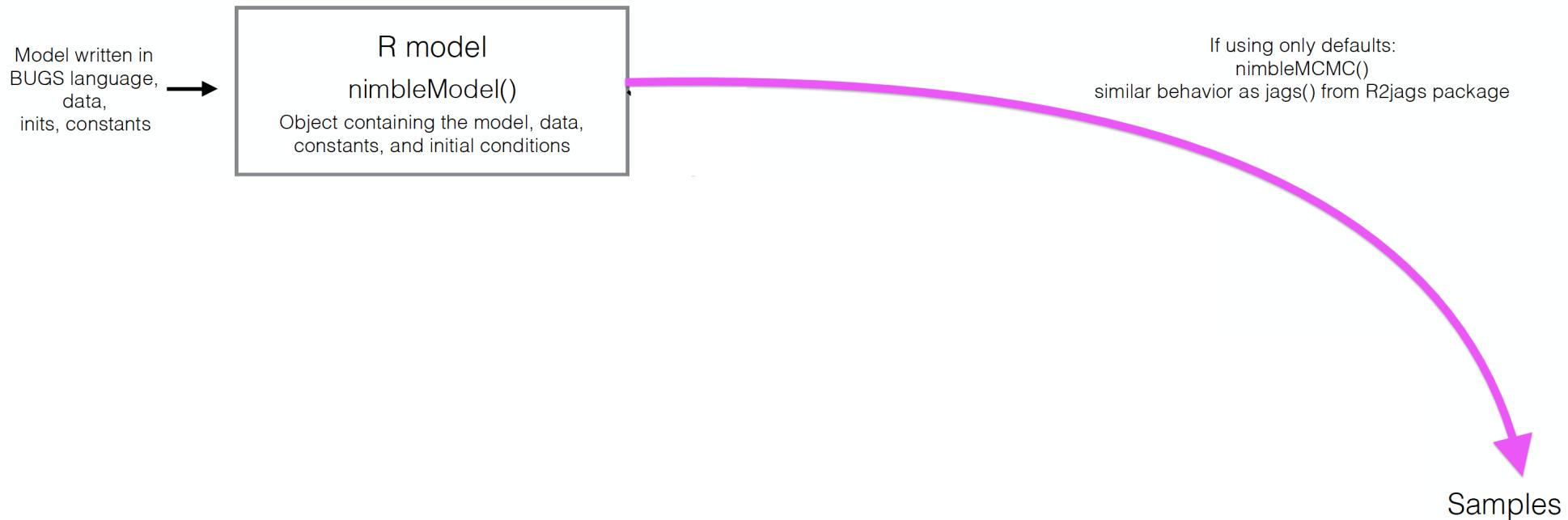


Skip your coffee break: Speed up MCMC convergence

The team

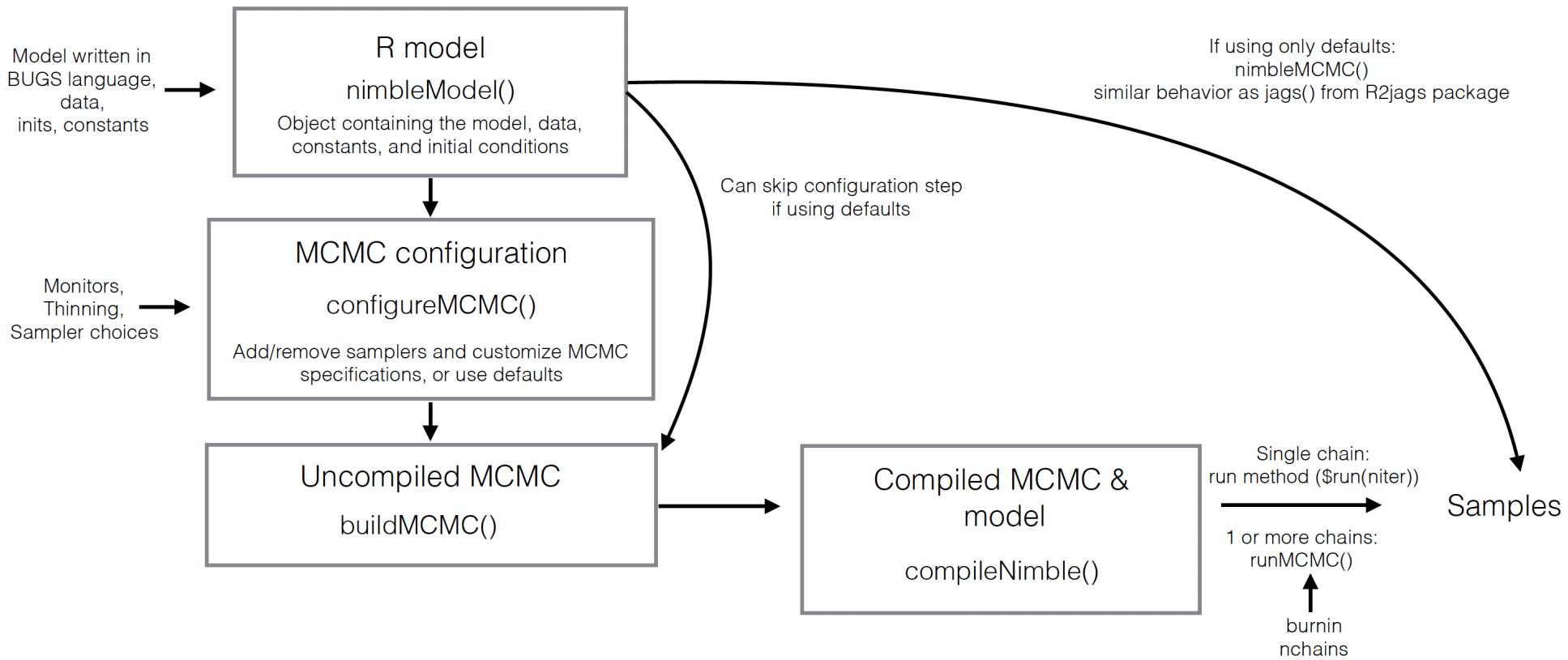
last updated: 2021-05-12

Our nimble workflow so far



Adapted from L. Ponisio

But nimble gives full access to the MCMC engine



Credit: L. Ponisio

Steps to use NIMBLE at full capacity

1. Build the model. It is an R object.
 2. Build the MCMC.
 3. Compile the model and MCMC.
 4. Run the MCMC.
 5. Extract the samples.
- `nimbleMCMC` does all of this at once.

Back to CJS models with Dipper data.

Define model

```
hmm.phip <- nimbleCode({
  delta[1] <- 1                      #  $Pr(\text{alive } t = 1) = 1$ 
  delta[2] <- 0                      #  $Pr(\text{dead } t = 1) = 0$ 
  phi ~ dunif(0, 1)      # prior survival
  gamma[1,1] <- phi                  #  $Pr(\text{alive } t \rightarrow \text{alive } t+1)$ 
  gamma[1,2] <- 1 - phi              #  $Pr(\text{alive } t \rightarrow \text{dead } t+1)$ 
  gamma[2,1] <- 0                  #  $Pr(\text{dead } t \rightarrow \text{alive } t+1)$ 
  gamma[2,2] <- 1                  #  $Pr(\text{dead } t \rightarrow \text{dead } t+1)$ 
  p ~ dunif(0, 1)      # prior detection
  omega[1,1] <- 1 - p                #  $Pr(\text{alive } t \rightarrow \text{non-detected } t)$ 
  omega[1,2] <- p                  #  $Pr(\text{alive } t \rightarrow \text{detected } t)$ 
  omega[2,1] <- 1                  #  $Pr(\text{dead } t \rightarrow \text{non-detected } t)$ 
  omega[2,2] <- 0                  #  $Pr(\text{dead } t \rightarrow \text{detected } t)$ 

  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2])
      y[i,j] ~ dcat(omega[z[i,j], 1:2])
    }
  }
})
```

Run and summarise

```
mcmc.phip <- nimbleMCMC(code = hmm.phip,
                           constants = my.constants,
                           data = my.data,
                           inits = initial.values,
                           monitors = parameters.to.save,
                           niter = n.iter,
                           nburnin = n.burnin,
                           nchains = n.chains)
```

```
|-----|-----|-----|-----|
|-----|-----|-----|-----|
|-----|-----|-----|-----|
|-----|-----|-----|-----|
```

```
MCMCsummary(object = mcmc.phip, round = 2)
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
p	0.90	0.03	0.83	0.90	0.95	1.00	286
phi	0.56	0.02	0.51	0.56	0.61	1.02	541

Detailed Nimble workflow

1. Build the model (R object)

```
hmm.phip <- nimbleModel(code = hmm.phip,  
                         constants = my.constants,  
                         data = my.data,  
                         inits = initial.values())
```

defining model...

building model...

setting data and initial values...

running calculate on model (any error reports that follow may simply reflect missing values in model variables)

checking model sizes and dimensions...

model building finished.

2. Build the MCMC

```
phip.mcmc.configuration <- configureMCMC(hmm.phip)
```

```
===== Monitors =====
thin = 1: phi, p, z
===== Samplers =====
RW sampler (2)
  - phi
  - p
posterior_predictive sampler (39)
  - z[] (39 elements)
categorical sampler (1103)
  - z[] (1103 elements)
```

```
phip.mcmc <- buildMCMC(phip.mcmc.configuration)
```

3. Compile the model and MCMC

```
phip.model <- compileNimble(hmm.phip)
```

```
compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compilation finished.
```

```
c.phip.mcmc <- compileNimble(phip.mcmc, project = phip.model)
```

```
compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compilation finished.
```

4. Run the MCMC

```
samples <- runMCMC(c.phip.mcmc, niter = 1000)
```

running chain 1...

```
|-----|-----|-----|-----|  
|-----|-----|-----|-----|
```

```
# Alternative:  
# c.phip.mcmc$run(1000)  
# samples <- as.matrix(c.phip.mcmc$mvSamples)
```

5. Look at results

```
summary(samples[, "phi"])
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
phi	0.4236	0.5472	0.5687	0.5698	0.5795	0.7440

```
summary(samples[, "p"])
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
p	0.5468	0.8803	0.8961	0.8771	0.9128	0.9686

Why is it useful?

Use and debug model in R

- Makes your life easier when it comes to debugging
- Inspect variables

```
hmm.phip$gamma
```

```
      [,1]      [,2]
[1, ] 0.4235601 0.5764399
[2, ] 0.0000000 1.0000000
```

- Calculate likelihood

```
hmm.phip$calculate()
```

```
[1] -1406.893
```

Example of debugging a model in R

- Pretend an impossible state was given in inits, making a dead bird alive again.

```
phip.model$calculate("z") # We can see there is a problem in z (states).
```

```
[1] -Inf
```

```
c(phip.model$calculate("z[5,"), # Bird 5 is valid.  
  phip.model$calculate("z[6,")) # Bird 6 isn't.
```

```
[1] -3.883928      -Inf
```

```
phip.model$z[6,]          # We have found the problem
```

```
[1] 1 1 2 1 2 2 2
```

Open the hood, and change/modify/write samplers

- Slice samplers instead of Metropolis-Hastings.
- Samplers on a log scale, especially for a variance, standard deviation, or precision parameter.
- Blocking correlated parameters.
- To know all samplers available in Nimble, type in `help(samplers)`.
- Source code for samplers and distributions is **in R** and can be copied and modified.
- Use `compareMCMCs` package to compare options (including Stan and Jags!).

Consider a model with wing length and individual random effect on survival.

```

hmm.phiwlrep <- nimbleCode({
  p ~ dunif(0, 1) # prior detection
  omega[1,1] <- 1 - p      # Pr(alive t -> non-detected t)
  omega[1,2] <- p          # Pr(alive t -> detected t)
  omega[2,1] <- 1          # Pr(dead t -> non-detected t)
  omega[2,2] <- 0          # Pr(dead t -> detected t)
  for (i in 1:N){
    logit(phi[i]) <- beta[1] + beta[2] * winglength[i] + eps[i]
    eps[i] ~ dnorm(mean = 0, sd = sdeps)
    gamma[1,1,i] <- phi[i]      # Pr(alive t -> alive t+1)
    gamma[1,2,i] <- 1 - phi[i]  # Pr(alive t -> dead t+1)
    gamma[2,1,i] <- 0          # Pr(dead t -> alive t+1)
    gamma[2,2,i] <- 1          # Pr(dead t -> dead t+1)
  }
  beta[1] ~ dnorm(mean = 0, sd = 1.5)
  beta[2] ~ dnorm(mean = 0, sd = 1.5)
  sdeps ~ dunif(0, 10)
  delta[1] <- 1              # Pr(alive t = 1) = 1
  delta[2] <- 0              # Pr(dead t = 1) = 0
  # likelihood
  for (i in 1:N){
    z[i,first[i]] ~ dcat(delta[1:2])
    for (j in (first[i]+1):T){
      z[i,j] ~ dcat(gamma[z[i,j-1], 1:2, i])
      y[i,j] ~ dcat(omega[z[i,j], 1:2])
    }
  }
}

```

Trace plot for standard deviation of the random effect
(default sampler)

Change samplers

- Good sampling strategies depend on the model and data. What are the samplers used by default?

```
mcmcConf <- configureMCMC(hmm.phirlrep.m)
```

```
===== Monitors =====
thin = 1: p, beta, sdeps, z
===== Samplers =====
RW sampler (259)
- p
- beta[] (2 elements)
- sdeps
- eps[] (255 elements)
posterior_predictive sampler (78)
- eps[] (39 elements)
- z[] (39 elements)
categorical sampler (1103)
- z[] (1103 elements)
```

Remove default sampler, and use slice sampler

```
mcmcConf$removeSamplers('sdeps')
mcmcConf$addSampler(target = 'sdeps',
                     type = "slice")
mcmcConf
```

```
===== Monitors =====
thin = 1: p, beta, sdeps, z
===== Samplers =====
slice sampler (1)
 - sdeps
RW sampler (258)
 - p
 - beta[] (2 elements)
 - eps[] (255 elements)
posterior_predictive sampler (78)
 - eps[] (39 elements)
 - z[] (39 elements)
categorical sampler (1103)
 - z[] (1103 elements)
```

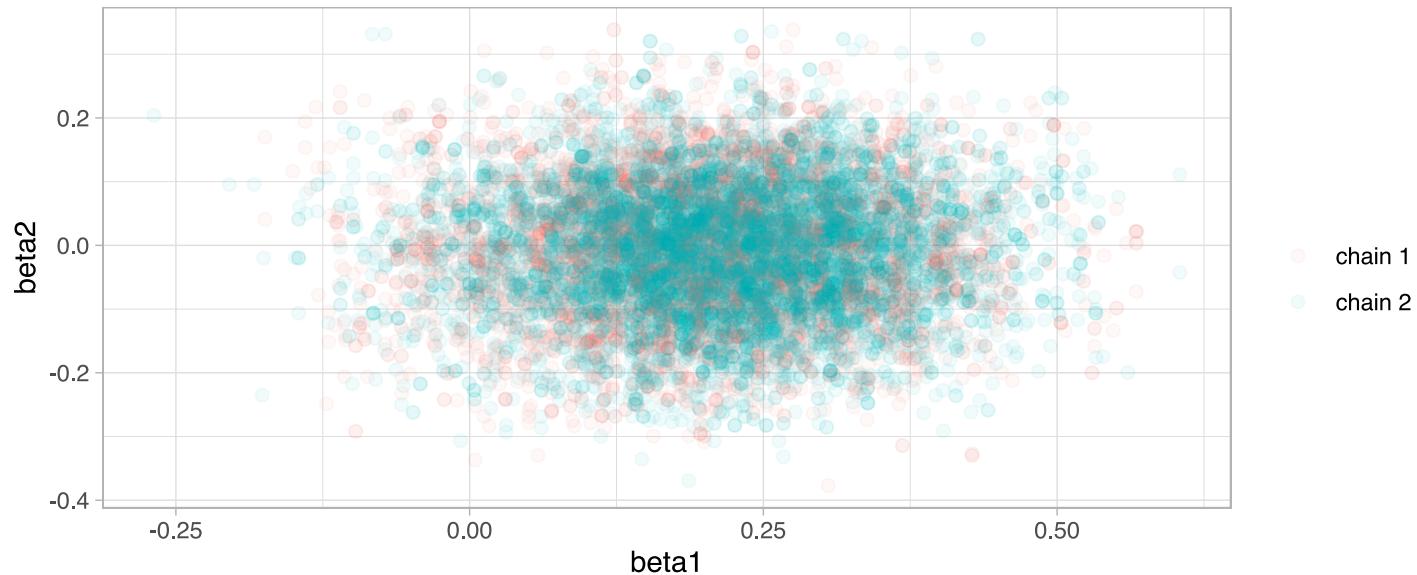
Trace plot for standard deviation of the random effect
(slice sampler)

Which is better?

- MCMC efficiency depends on both mixing and computation time.
- MCMC efficiency = Effective Sample Size (ESS) / computation time.
- MCMC efficiency is the number of effectively independent posterior samples generated per second.
- ESS is different for each parameter. (Computation time is the same for each parameter.)
- ESS can be estimated from packages `coda` or `mcmcse`. These give statistical estimates, so different runs will give different estimates.
- Efficiency with default sampler = $25.7 / 21.53 = 1.19$.
- Efficiency with slice sampler = $19.24 / 19.39 = 0.99$.

Block sampling

- High correlation in (regression) parameters may make independent samplers inefficient.



- Block sampling (propose candidate values from multivariate distribution) might help.

Block sampling

- Remove and replace independent RW samples by block sampling. Then proceed as usual.

```
mcmcConf$removeSamplers(c('beta[1]', 'beta[2]'))
mcmcConf$addSampler(target = c('beta[1]', 'beta[2]'),
                     type = "RW_block")
```

Block sampling

```
mcmcConf
```

```
===== Monitors =====
thin = 1: p, beta, sdeps, z
===== Samplers =====
slice sampler (1)
 - sdeps
RW_block sampler (1)
 - beta[1], beta[2]
RW sampler (256)
 - p
 - eps[]  (255 elements)
posterior_predictive sampler (78)
 - eps[]  (39 elements)
 - z[]   (39 elements)
categorical sampler (1103)
 - z[]  (1103 elements)
```

Summary of strategies for improving MCMC

- Choose better initial values.
- Customize sampler choice (more in [Chapter 7 of the User's manual](#)).
- Reparameterize, e.g. standardize covariates, deal with parameter redundancy.
- Rewrite the model.
 - Vectorize to improve computational efficiency (not covered).
 - Avoid long chains of deterministic dependencies.
 - Marginalize to remove parameters
 - Use new functions and new distributions written as nimbleFunctions.
- Write new samplers that take advantage of particular model structures (not covered).
- Using multiple cores with parallelization: see how-to at https://r-nimble.org/nimbleExamples/parallelizing_NIMBLE.html

Marginalization

- User-defined distributions is another neat feature of Nimble.
- Integrate over latent states if those are not the focus of ecological inference (marginalization).
- Marginalization often (but not always) improves MCMC. See [Ponisio et al. 2020](#) for examples.
- The [nimbleEcology](#) package implements capture-recapture models and HMMs with marginalization.

Our model ($\phi_A, \phi_B, \psi_{AB}, \psi_{BA}, p_A, p_B$)

```
multisite <- nimbleCode( {
  ...
  # Likelihood
  for (i in 1:N){
    # Define latent state at first capture
    z[i,first[i]] <- y[i,first[i]] - 1
    for (t in (first[i]+1):K){
      # State process: draw S(t) given S(t-1)
      z[i,t] ~ dcat(gamma[z[i,t-1],1:3])
      # Observation process: draw O(t) given S(t)
      y[i,t] ~ dcat(omega[z[i,t],1:3])
    }
  }
})
```

Same model with nimbleEcology

```
multisite <- nimbleCode({  
  ...  
  # initial state probs  
  for(i in 1:N) {  
    init[i, 1:4] <- gamma[ y[i, first[i] ] - 1, 1:4 ] # first state propagation  
  }  
  
  # likelihood  
  for (i in 1:N){  
    y[i,(first[i]+1):K] ~ dHMM(init = init[i,1:4],           # count data from first[i] + 1  
                                prob0bs = omega[1:4,1:4],       # observation matrix  
                                probTrans = gamma[1:4,1:4],     # transition matrix  
                                len = K - first[i],          # nb of occasions  
                                checkRowSums = 0)            # do not check whether elements in a row sum to 1  
  }  
  ...  
}
```

- This runs twice as fast as the standard formulation with explicit latent states.
- Marginalizing typically gives better mixing.

Reducing redundant calculations

- So far, a row of the dataset is an individual. However, several individuals may share the same encounter history.
- The contribution of M individuals with the same encounter history is the likelihood of this particular encounter history raised to the power M .
- Using this so-called **weighted likelihood** greatly decreases the computational burden.
- This idea is used in most computer programs that implement maximum likelihood. In the Bayesian framework, the same idea was proposed in [Turek et al. \(2016\)](#).
- Cannot be done in Jags. Can be done in nimble thanks to nimble functions!
- The run is *much* faster. Also allows fitting models to big datasets. More details in dedicated Worksheet.

No live demo, but there is a worksheet.



Future directions for NIMBLE

- NIMBLE is under active development. Contributors are welcome, including those who want to get involved but don't know where.
- Faster building of models and algorithms. Ability to save and re-load compiled work.
- Automatic differentiation of model calculations, enabling Hamiltonian Monte Carlo, other sampling strategies, and Laplace approximation.
- Tools for building packages that use NIMBLE "under the hood".

Further reading

- Turek, D., de Valpine, P. & Paciorek, C.J. Efficient Markov chain Monte Carlo sampling for hierarchical hidden Markov models *Environ Ecol Stat* 23: 549–564 (2016).
- Ponisio, L.C., de Valpine, P., Michaud, N., and Turek, D. One size does not fit all: Customizing MCMC methods for hierarchical models using NIMBLE *Ecol Evol*. 10: 2385–2416 (2020).
- Nimble workshop to come 26-28 May, check out [here](#).
- Nimble workshop material online available [here](#).
- Nimble [manual](#) and [cheatsheet](#).

Conclusions

The team

last updated: 2021-05-15

Take-home messages and recommendations

Make the best of your data with HMMs

- Here is a [searchable list](#) of HMM analyses of capture-recapture data.
- This list is not exhaustive, please get in touch with us if you'd like to add a reference.

Bayesian capture-recapture analysis with HMMs

- Make your ecological question explicit.
- Think of observations and states first.
- Then write down the observation and transition matrices on paper.
- Start simple, all parameters constant for example. Make sure convergence is reached.
- Add complexity one step at a time.

Bayesian capture-recapture analysis with HMMs

- Use simulations to better understand your model.
- Nimble models can be used to simulate data, check out [this tutorial](#).
- Do not try to optimize your code. Make it work first, then think of optimization.

"Premature optimization is the root of all evil" - Donald Knuth (creator of TeX and author of "The Art of Computer Programming")

- Read [Bayesian workflow](#) by Gelman et al. (2021).

Till next time

- The Slack space will remain for some time. Happy to answer questions you might have related to the workshop.
- Website will be updated with
 - video recordings
 - your feedbacks
 - a FAQ section based on your questions
- A book is on its way. More in 2022 hopefully.