

# Introduction to Bayesian statistics with R

## 5. Case studies and GLMMs

---

Olivier Gimenez

last updated: 2025-03-10

# Introduction

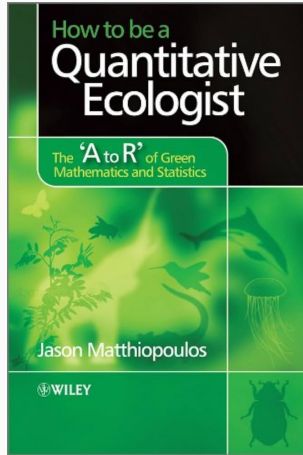
---

- GLMs avec les 3 composantes
- Intuition des GLMMs

## GLMM with Poisson response

---

# From Jason Matthiopoulos' book



## Longitudinal study on coral reef

*A survey of a coral reef uses 10 predefined linear transects covered by divers once every week. The response variable of interest is the abundance of a particular species of anemone as a function of water temperature. Counts of anemones are recorded at 20 regular line segments along the transect. The following piece of code will generate a data set with realistic properties according to the above design. Make sure you understand what it is doing. You might want to explain the script to the colleague next to you. Also, to try and make sense of the code of others, it is always good to plot and/or run small sections of the code.*

# Data generation

```
transects <- 10
data <- NULL
for (tr in 1:transects){
  # random effect (intercept)
  ref <- rnorm(1,0,.5)
  # water temperature gradient
  t <- runif(1, 18,22) + runif(1,-.2,0.2)*1:20
  # Anemone gradient (expected response)
  ans <- exp(ref -14 + 1.8 * t - 0.045 * t^2)
  # actual counts on 20 segments of the current transect
  an <- rpois(20, ans)
  data <- rbind(data, cbind(rep(tr, 20), t, an))
}
```

- Generate a data set using the anemone code.
- Using NIMBLE and brms, fit a GLMM with quadratic effect of temperature and a random intercept.
- Fit the same model to the same data in a Frequentist framework using function `lme4::glmer()`.
- Compare the estimates.



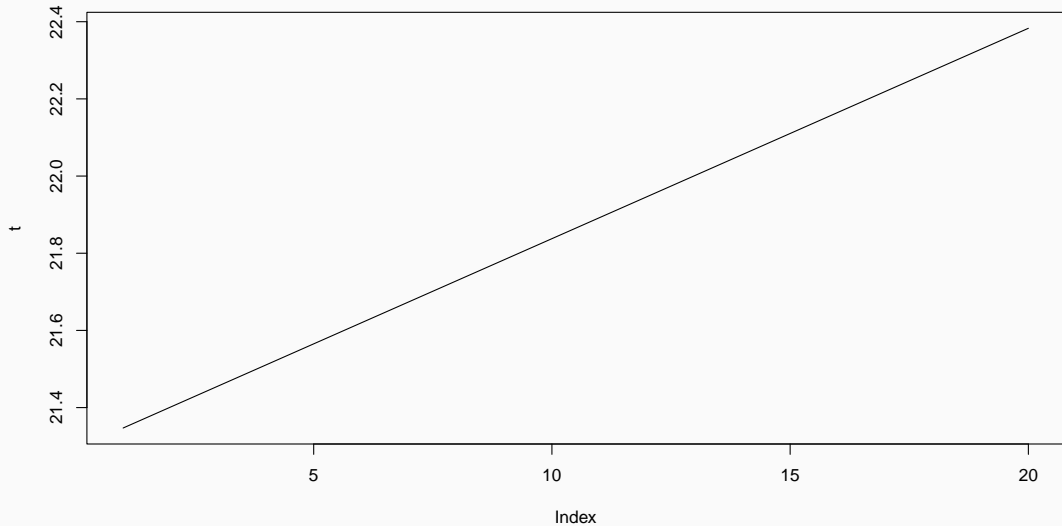
## Make sense of the code

- Always difficult to make sense of the code of others.
- Good to plot and/or run small sections of the code.

## Make sense of the code

```
# random effect (intercept)  
ref <- rnorm(1,0,.5)  
# water temperature gradient  
t <- runif(1, 18,22) + runif(1,-.2,0.2)*1:20  
  
plot(t,type='l')
```

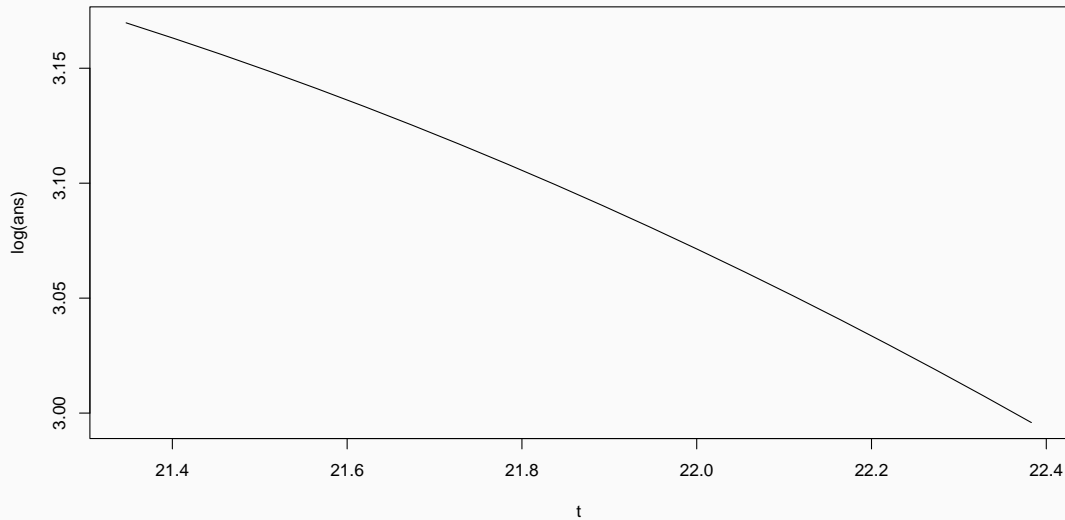
## Make sense of the code



## Make sense of the code

```
# Anemone gradient (expected response)  
ans <- exp(ref -14 + 1.8 * t - 0.045 * t^2)  
plot(t,log(ans),type='l')
```

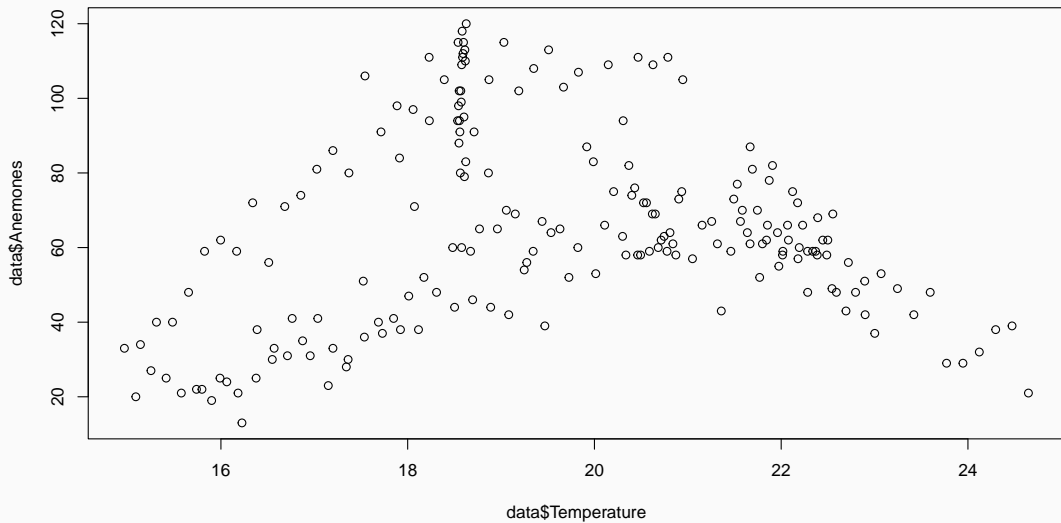
## Make sense of the code



## Make sense of the code

```
data <- data.frame(Transect = data[,1],  
                  Temperature = data[,2],  
                  Anemones = data[,3])  
  
plot(data$Temperature, data$Anemones)
```

## Make sense of the code



## Write down model

$\text{Count}_i \sim \text{Poisson}(\lambda_i)$  [likelihood]

$\log(\lambda_i) = a_{\text{TRANSECT}[i]} + b_1 \text{temp}_i + b_2 \text{temp}_i^2$  [linear model]

$a_j \sim \text{Normal}(\bar{a}, \sigma)$  [prior for varying intercepts]

$\bar{a} \sim \text{Normal}(0, 1.5)$  [prior for population mean]

$\sigma \sim \text{Uniform}(0, 10)$  [prior for standard deviation]

$b_1, b_2 \sim \text{Normal}(0, 1.5)$  [prior for slopes]



**NIMBLE**

---

## Load package

```
library(nimble)
```

# Standardize temperature covariate

```
boo <- data$Temperature
data$Temp <- (boo - mean(boo)) / sd(boo)
head(data)
```

#>	Transect	Temperature	Anemones	Temp
#> 1	1	19.46887	39	-0.05470958
#> 2	1	19.27560	56	-0.13851042
#> 3	1	19.08233	42	-0.22231126
#> 4	1	18.88906	44	-0.30611209
#> 5	1	18.69579	46	-0.38991293
#> 6	1	18.50253	44	-0.47371377

# Model

```
model <- nimbleCode({  
  for (i in 1:n){  
    count[i] ~ dpois(lambda[i])  
    log(lambda[i]) <- intercept[transect[i]] + slope[1] * x[i] + slope[2] * pow(x[i],2)  
  }  
  for (j in 1:nbtransects){  
    intercept[j] ~ dnorm(mu.a, sd = sigma.a)  
  }  
  mu.a ~ dnorm(0, sd = 1.5)  
  sigma.a ~ dunif(0, 10)  
  slope[1] ~ dnorm(0, sd = 1.5)  
  slope[2] ~ dnorm(0, sd = 1.5)  
})
```

```
my.constants <- list(n = nrow(data),  
                    nbtransects = transects)  
my.data <- list(x = data$Temp,  
               count = data$Anemones,  
               transect = data$Transect)
```

## Initial values

```
init1 <- list(intercept = rnorm(transects), slope = rnorm(2),  
              mu.a = rnorm(1), sigma.a = runif(1))  
init2 <- list(intercept = rnorm(transects), slope = rnorm(2),  
              mu.a = rnorm(1), sigma.a = runif(1))  
initial.values <- list(init1, init2)
```

## Parameters to monitor (and save)

```
parameters.to.save <- c("slope", "mu.a", "sigma.a")
```

## Set up MCMC details

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



## Run model

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

## Post-process results

```
library(MCMCvis)
MCMCsummary(object = mcmc.output, round = 2)
#>           mean    sd  2.5%   50% 97.5% Rhat n.eff
#> mu.a         4.34 0.12  4.10  4.34  4.57 1.00  6789
#> sigma.a      0.36 0.10  0.21  0.34  0.61 1.00  1115
#> slope[1]     0.09 0.03  0.03  0.09  0.16 1.09   131
#> slope[2]    -0.26 0.02 -0.29 -0.26 -0.22 1.00   262
```

## Get regression coefficients

Convert regression coefficients from scaled to non-scaled and compare to values used to generate data (from <https://stats.stackexchange.com/questions/361995/how-to-convert-coefficients-from-quadratic-function-from-scaled-to-not-scaled-co>)

Pull two chains together:

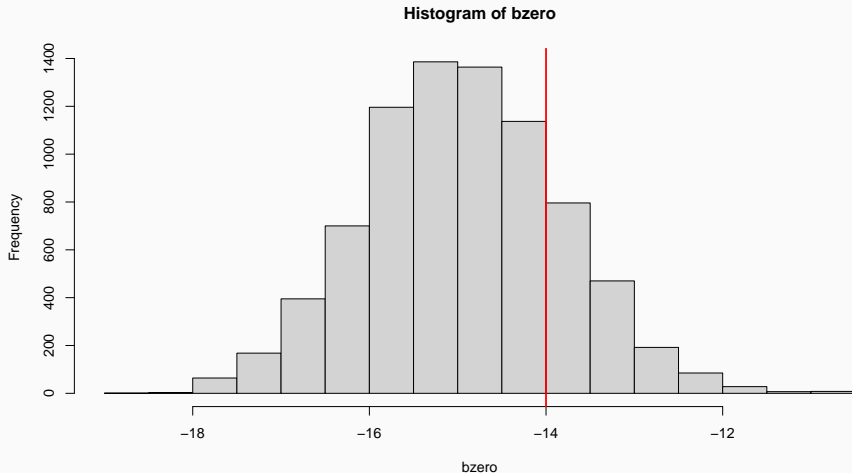
```
samples <- rbind(mcmc.output$chain1,mcmc.output$chain2)
```

## Get regression coefficients

```
sbzero <- samples[, 'mu.a']  
sbun <- samples[, 'slope[1]']  
sbdeux <- samples[, 'slope[2]']  
  
mu <- mean(boo)  
sg <- sd(boo)
```

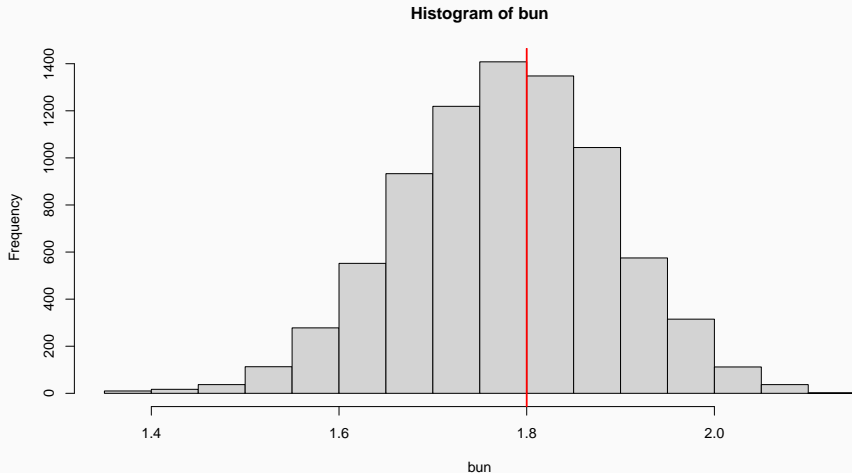
# Get regression coefficients

```
bzero <- sbzero - sbun*mu/sg + sbdeux*mu^2/(sg^2)
hist(bzero)
abline(v = -14, col = "red", lwd = 2)
```



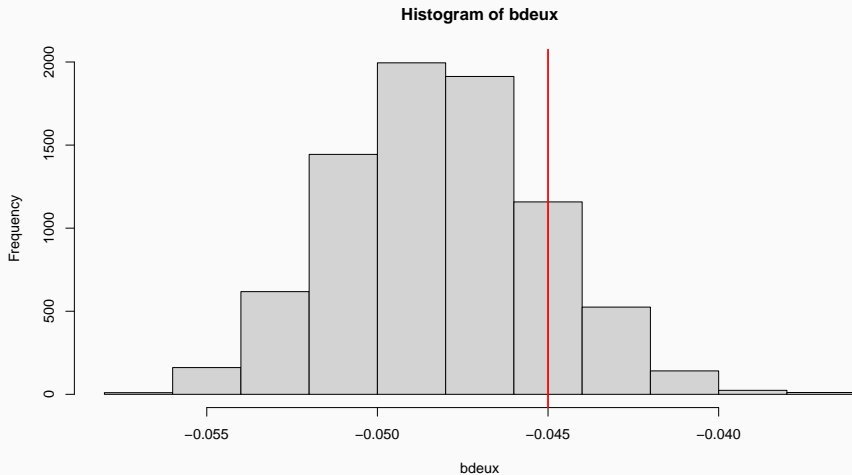
# Get regression coefficients

```
bun <- sbdeux/sg - 2 * sbdeux * mu / (sg^2)
hist(bun)
abline(v = 1.8, col = "red", lwd = 2)
```



# Get regression coefficients

```
bdeux <- sbdeux/(sg^2)
hist(bdeux)
abline(v = - 0.045, col = "red", lwd = 2)
```



## Frequentist approach

---

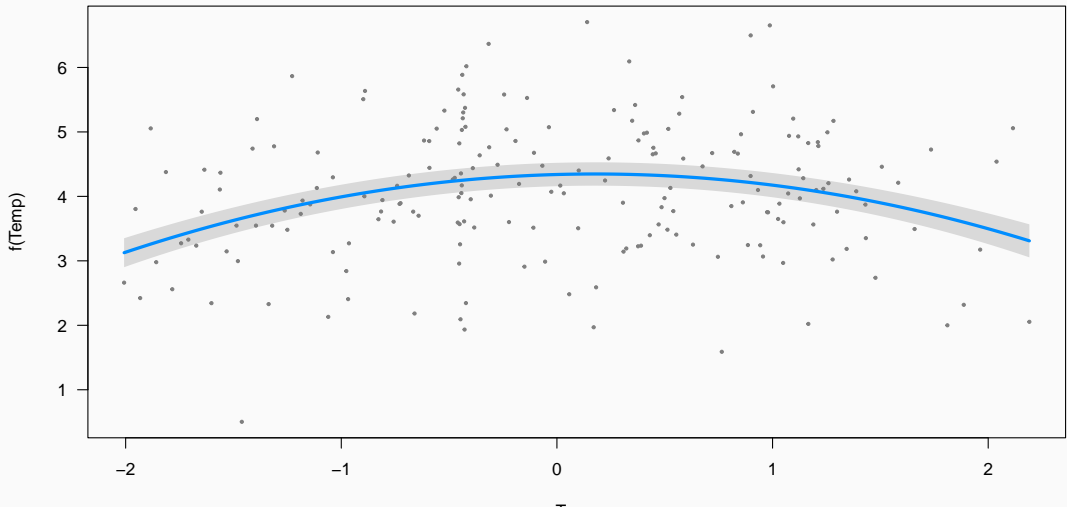


# lme4 fit

```
library(lme4)
fit_lme4 <- glmer(Anemones ~ Temp + I(Temp^2) + (1 | Transect), data = data, family = poisson)
fit_lme4
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#> Approximation) [glmerMod]
#> Family: poisson ( log )
#> Formula: Anemones ~ Temp + I(Temp^2) + (1 | Transect)
#> Data: data
#>      AIC      BIC    logLik deviance df.resid
#> 1443.3781 1456.5714 -717.6891 1435.3781      196
#> Random effects:
#> Groups   Name      Std.Dev.
#> Transect (Intercept) 0.2878
#> Number of obs: 200, groups: Transect, 10
#> Fixed Effects:
#> (Intercept)      Temp    I(Temp^2)
#>    4.33936    0.09095    -0.25586
```

# Visualize

```
visreg::visreg(fit_lme4, xvar = 'Temp')
```



**brms**

---

## Implementation

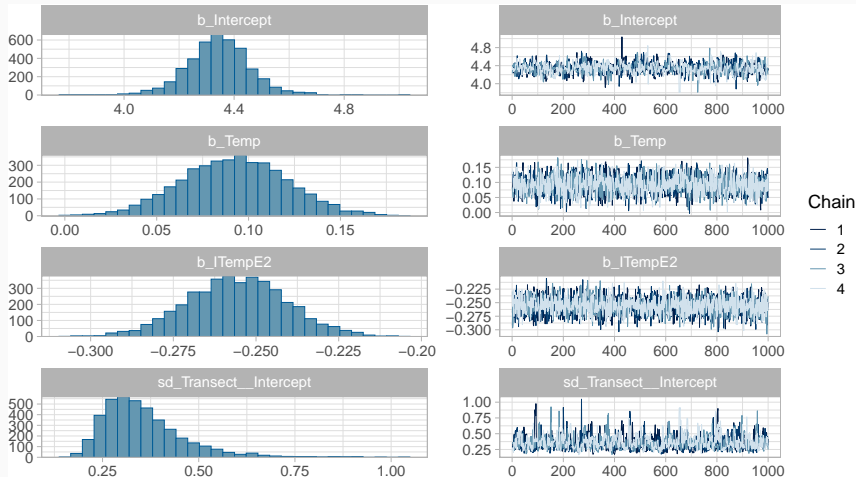
```
library(brms)
bayes.brms <- brm(Anemones ~ Temp + I(Temp^2) + (1 | Transect),
  data = data,
  family = poisson("log"),
  chains = 2, # nb of chains
  iter = 5000, # nb of iterations, including burnin
  warmup = 1000, # burnin
  thin = 1)
```

# Numerical summaries

```
summary(bayes.brms)
#> Family: poisson
#> Links: mu = log
#> Formula: Anemones ~ Temp + I(Temp^2) + (1 | Transect)
#> Data: data (Number of observations: 200)
#> Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
#> total post-warmup draws = 4000
#>
#> Multilevel Hyperparameters:
#> ~Transect (Number of levels: 10)
#>      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> sd(Intercept)    0.36      0.11    0.21    0.63 1.00      697    1083
#>
#> Regression Coefficients:
#>      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> Intercept      4.34      0.12    4.11    4.58 1.01      757     981
#> Temp           0.09      0.03    0.04    0.15 1.00     2110    2162
#> ITempE2       -0.26      0.02   -0.29   -0.23 1.00     2324    2210
#>
#> Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
#> and Tail_ESS are effective sample size measures, and Rhat is the potential
#> scale reduction factor on split chains (at convergence, Rhat = 1).
```

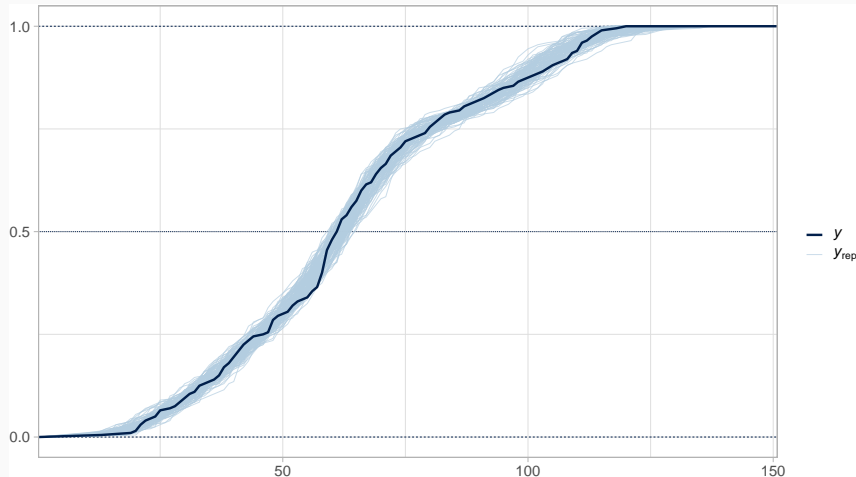
# Convergence

```
plot(bayes.brms)
```



## Quality of fit assessment

```
pp_check(bayes.brms, ndraws = 100, type = 'ecdf_overlay')
```



## Model comparison

- What if we'd like to test the effect of temperature using WAIC?
- We fit a model with no effect of temperature:

```
bayes.brms2 <- brm(Anemones ~ 1 + (1 | Transect),  
  data = data,  
  family = poisson("log"),  
  chains = 2, # nb of chains  
  iter = 5000, # nb of iterations, including burnin  
  warmup = 1000, # burnin  
  thin = 1)
```



## Model comparison

Then we compare both models, by ranking them with their WAIC:

```
waic1 <- waic(bayes.brms) # waic model w/ temperature
waic2 <- waic(bayes.brms2) # waic model wo/ temperature
data.frame(with_temp = waic1$waic,
            without_temp = waic2$waic)
#>   with_temp without_temp
#> 1  1403.737    1755.672
```

## GLMM with binomial response

---

Received: 5 July 2021 | Revised: 10 August 2021 | Accepted: 30 September 2021

DOI: 10.1111/ele.13915

## LETTER

ECOLOGY LETTERS  WILEY

### Sampling bias exaggerates a textbook example of a trophic cascade



Elaine M. Brice<sup>1</sup>  | Eric J. Larsen<sup>2</sup> | Daniel R. MacNulty<sup>1</sup> 

<sup>1</sup>Department of Wildland Resources and Ecology Center, Utah State University, Logan, Utah, USA

<sup>2</sup>Department of Geography and Geology, University of Wisconsin – Stevens Point, Stevens Point, Wisconsin, USA

#### Correspondence

Elaine M. Brice, Department of Wildland Resources and Ecology Center, Utah State University, Logan, UT 84322, USA.  
Email: elbrice92@gmail.com

#### Funding information

National Science Foundation, Grant/Award Number: DGE-1633756; Yellowstone National Park; University of Wyoming-National Park Service Small Grant Program, Grant/Award Number: 1003867-USU; Utah State University

Editor: Tim Coulson

#### Abstract

Understanding trophic cascades in terrestrial wildlife communities is a major challenge because these systems are difficult to sample properly. We show how a tradition of non-random sampling has confounded this understanding in a textbook system (Yellowstone National Park) where carnivore [*Canis lupus* (wolf)] recovery is associated with a trophic cascade involving changes in herbivore [*Cervus canadensis* (elk)] behaviour and density that promote plant regeneration. Long-term data indicate a practice of sampling only the tallest young plants overestimated regeneration of overstory aspen (*Populus tremuloides*) by a factor of 4–7 compared to random sampling because it favoured plants taller than the preferred browsing height of elk and overlooked non-regenerating aspen stands. Random sampling described a trophic cascade, but it was weaker than the one that non-random sampling described. Our findings highlight the critical importance of basic sampling principles (e.g. randomisation) for achieving an accurate understanding of trophic cascades in terrestrial wildlife systems.

#### KEYWORDS

aspen, carnivore, elk, non-random sampling, predator indirect effects, preferred browsing height, sampling bias, trophic cascade, ungulate, wolf

## Read in data

```
rawdat <- read_csv("dat/Aspen_Data.csv")
head(rawdat)
#> # A tibble: 6 x 6
#>   Plot Year Tree Browse Height Type
#>   <dbl> <dbl> <dbl>  <dbl>  <dbl> <chr>
#> 1     1   2008     1      1    115 5T
#> 2     1   2008     2      1    130 5T
#> 3     1   2008     3      1    113 5T
#> 4     1   2008     4      1    110 5T
#> 5     1   2008     5      0    132 5T
#> 6     1   2009     1      1    200 5T
```

## Data collection (1)

*We measured browsing and height of young aspen in 113 plots distributed randomly across the study area. Each plot was a 1 x 20 m belt transect located randomly within an aspen stand that was itself randomly selected from an inventory of stands with respect to high and low wolf-use areas. The inventory was a list of 992 grid cells (240 × 360 m) that contained at least one stand. A 'stand' was a group of tree-size aspen (more than 10 cm diameter at breast height) in which each tree was less than 30 m from every other tree. One hundred and thirteen grid cells were randomly selected from the inventory, one stand was randomly selected from each cell, and one plot was randomly established in each stand.*

## Data collection (2)

*We measured aspen at the end of the growing season (late July–September), focusing on plants less than 600 cm tall and more than 1 year old, which we termed ‘young aspen’. For each stand, we measured every young aspen within a plot (‘random stems’) and each of the five tallest young aspen within the stand (‘5T stems’). For all young aspen, we measured browsing status (browsed or unbrowsed) and height of the leader (tallest) stem. A leader was ‘browsed’ if its growth from the previous growing season had been eaten, which we identified by a sharp, pruned edge at the base of the current year’s growth. Most plots were measured nearly every year since 1999 and our analysis focused on data from 10 years (2007–2014, 2016–2017) in which sampled stands included measurements of random and 5T stems.*

*We combined measurements of 5T and random stems into one data set of all stems ( $N = 18,623$ ) across all years ( $N = 10$  years).*

# Our job

- Read in the data and get familiar with it.
- Using NIMBLE and brms, fit a GLMM with 'Browsed' as the response variable. Consider year and type of stems (random or 5T stems) as explanatory variables. Use stand identity as a random effect on the intercept.
- Fit the same model to the same data in a Frequentist framework using function `lme4::glmer()`.
- Compare the estimates.



## Read in data

```
str(rawdat)
#> spc_tbl_ [18,792 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
#> $ Plot   : num [1:18792] 1 1 1 1 1 1 1 1 1 1 ...
#> $ Year   : num [1:18792] 2008 2008 2008 2008 2008 ...
#> $ Tree   : num [1:18792] 1 2 3 4 5 1 2 3 4 5 ...
#> $ Browse: num [1:18792] 1 1 1 1 0 1 0 0 0 0 ...
#> $ Height: num [1:18792] 115 130 113 110 132 200 208 218 240 260 ...
#> $ Type   : chr [1:18792] "5T" "5T" "5T" "5T" ...
#> - attr(*, "spec")=
#> .. cols(
#> ..   Plot = col_double(),
#> ..   Year = col_double(),
#> ..   Tree = col_double(),
#> ..   Browse = col_double(),
#> ..   Height = col_double(),
#> ..   Type = col_character()
#> .. )
#> - attr(*, "problems")=<externalptr>
```

# Metadata

- Plot: individual identifier for each of 113 plots distributed randomly across the study area. Each plot was a  $1 \times 20$  m belt transect located randomly within an aspen stand
- Year: year in which aspen was sampled
- Tree: individual identifier for each stem within a plot
- Browse: denotes the browsing status (browsed = 1, unbrowsed = 0) of the leader (tallest) stem. A leader was 'browsed' if its growth from the previous growing season had been eaten
- Height: height (cm) of the leader stem of each individual aspen
- Type: sampling method. Every young aspen within a plot is a "random" stem, and each of the five tallest young aspen within the stand is a "5T" stem.

## Format data

```
dat <- rawdat %>%  
  transmute(y = Browse,  
            type = if_else(Type == "5T", 1, 0), # numeric  
            year = as.integer(Year) - 2006,    # year starts at 1  
            id = as_factor(Plot))              # id is a factor
```

## Write down model

$\text{Browsed}_i \sim \text{Bernoulli}(p_i)$  [likelihood]

$\text{logit}(p_i) = a_{\text{ID}[i]} + b_1 \text{ year}_i + b_2 \text{ type}_i$  [linear model]

$a_j \sim \text{Normal}(\bar{a}, \sigma)$  [prior for varying intercepts]

$\bar{a} \sim \text{Normal}(0, 1.5)$  [prior for population mean]

$\sigma \sim \text{Uniform}(0, 10)$  [prior for standard deviation]

$b_1, b_2 \sim \text{Normal}(0, 1.5)$  [prior for slopes]

**NIMBLE**

---

## Load package

```
library(nimble)
```

# Model

```
model <- nimbleCode({  
  for (i in 1:n){  
    y[i] ~ dbern(p[i])  
    logit(p[i]) <- a[id[i]] + beta[1] * type[i] + beta[2] * year[i] + beta[3] * year[i] * type[i]  
  }  
  for (j in 1:nlevels){  
    a[j] ~ dnorm(mu.a, sd = sigma.a)  
  }  
  for (k in 1:3){  
    beta[k] ~ dnorm(0, sd = 1.5)  
  }  
  mu.a ~ dnorm(0, sd = 1.5)  
  sigma.a ~ dunif(0,10)  
})
```

```
my.constants <- list(id = as.numeric(as_factor(dat$id)),  
                    n = length(dat$y),  
                    nlevels = length(levels(as_factor(dat$id))))  
my.data <- list(y = dat$y,  
              type = dat$type,  
              year = dat$year)
```



## Initial values

```
init1 <- list(mu.a = -0.5, sigma.a = 0.1, beta = rnorm(3))  
init2 <- list(mu.a = 0.5, sigma.a = 0.5, beta = rnorm(3))  
initial.values <- list(init1, init2)
```

## Parameters to monitor (and save)

```
parameters.to.save <- c("mu.a", "sigma.a", "beta")
```

## Set up MCMC details

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

## Run model

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

## Post-process results

```
library(MCMCvis)
MCMCsummary(object = mcmc.output, round = 2)
#>           mean    sd  2.5%   50% 97.5% Rhat n.eff
#> beta[1] -0.42 0.09 -0.60 -0.43 -0.26 1.00   190
#> beta[2] -0.14 0.01 -0.15 -0.14 -0.12 1.03   192
#> beta[3] -0.08 0.01 -0.11 -0.08 -0.06 1.00   203
#> mu.a      1.57 0.13  1.31  1.57  1.82 1.00  1841
#> sigma.a   1.26 0.09  1.09  1.25  1.45 1.00  1228
```

# Visualize (1)

```
# pool two chains together
samples <- rbind(mcmc.output$chain1, mcmc.output$chain2)
nsim <- nrow(samples)

# get values sampled in posterior distribution of regression parameters
beta1 <- samples[, 'beta[1]']
beta2 <- samples[, 'beta[2]']
beta3 <- samples[, 'beta[3]']
intercept <- samples[, 'mu.a']

# predict
logitp_5T <- matrix(NA, nrow = nsim, ncol = 11)
logitp_random <- matrix(NA, nrow = nsim, ncol = 11)
for (i in 1:nsim){ # loop over simulations
  for (j in 1:11){ # loop over years
    logitp_5T[i,j] <- intercept[i] + beta1[i] * 1 + beta2[i] * j + beta3[i] * 1 * j
    logitp_random[i,j] <- intercept[i] + beta1[i] * 0 + beta2[i] * j + beta3[i] * 0 * j
  }
}

# back-transform
p_5T <- plogis(logitp_5T)
p_random <- plogis(logitp_random)
```

## Visualize (2)

```
# means and credible intervals
```

```
p_5T_mean <- apply(p_5T, 2, mean)
```

```
p_5T_inf <- apply(p_5T, 2, quantile, probs = 2.5/100)
```

```
p_5T_sup <- apply(p_5T, 2, quantile, probs = 97.5/100)
```

```
p_random_mean <- apply(p_random, 2, mean)
```

```
p_random_inf <- apply(p_random, 2, quantile, probs = 2.5/100)
```

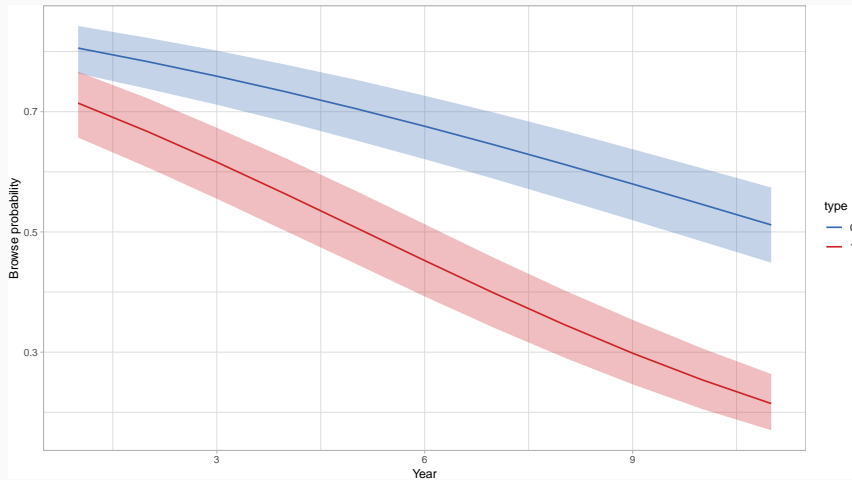
```
p_random_sup <- apply(p_random, 2, quantile, probs = 97.5/100)
```

## Visualize (3)

```
data.frame(year = c(1:11, 1:11),
            type = factor(c(rep(1, 11), rep(0, 11))), # Convert type to a factor
            p_mean = c(p_5T_mean, p_random_mean),
            p_inf = c(p_5T_inf, p_random_inf),
            p_sup = c(p_5T_sup, p_random_sup)) %>%
ggplot() +
geom_ribbon(aes(x = year,
               ymin = p_inf,
               ymax = p_sup,
               group = type,
               fill = type),
           alpha = 0.3,
           show.legend = F) +
geom_line(aes(x = year,
              y = p_mean,
              group = type,
              color = type),
          size=0.7) +
scale_color_manual(values = c("1" = "#CC2529", "0" = "#396AB1")) +
scale_fill_manual(values = c("1" = "#CC2529", "0" = "#396AB1")) +
xlab("Year")+
ylab("Browse probability")+
theme_light()
```



## Visualize (3)



## Frequentist approach

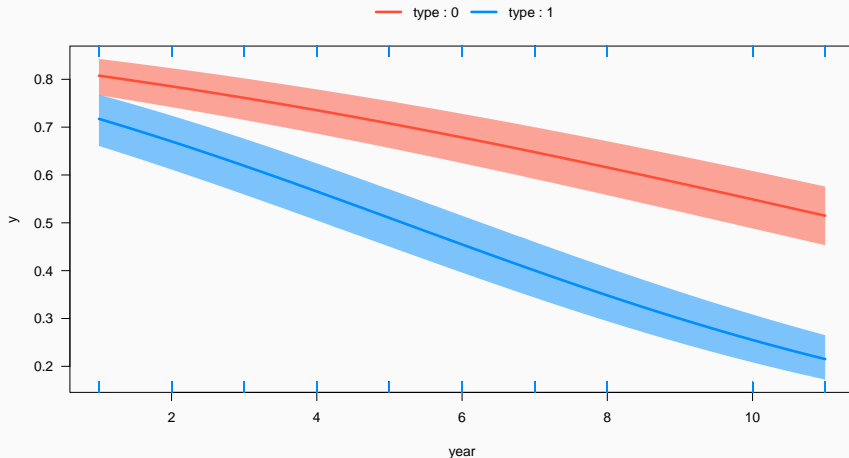
---

# lme4 fit

```
library(lme4)
fit_lme4 <- glmer(y ~ type * year + (1|id), data = dat, family = "binomial")
fit_lme4
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#> Approximation) [glmerMod]
#> Family: binomial (logit)
#> Formula: y ~ type * year + (1 | id)
#> Data: dat
#>      AIC      BIC    logLik deviance df.resid
#> 21428.86 21468.02 -10709.43  21418.86    18618
#> Random effects:
#> Groups Name      Std.Dev.
#> id      (Intercept) 1.233
#> Number of obs: 18623, groups: id, 113
#> Fixed Effects:
#> (Intercept)      type      year  type:year
#>    1.57173    -0.41777    -0.13750    -0.08506
```

# Visualize

```
visreg::visreg(fit = fit_lme4, xvar = "year",  
               by = "type", overlay = TRUE, scale = "response")
```



**brms**

---

## Implementation

```
library(brms)
bayes.brms <- brm(y ~ type * year + (1|id),
  data = dat,
  family = bernoulli("logit"),
  chains = 2, # nb of chains
  iter = 5000, # nb of iterations, including burnin
  warmup = 1000, # burnin
  thin = 1)
```

# Numerical summaries

```
summary(bayes.brms)
#> Family: bernoulli
#> Links: mu = logit
#> Formula: y ~ type * year + (1 | id)
#> Data: dat (Number of observations: 18623)
#> Draws: 2 chains, each with iter = 5000; warmup = 1000; thin = 1;
#> total post-warmup draws = 8000
#>
#> Multilevel Hyperparameters:
#> ~id (Number of levels: 113)
#>      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> sd(Intercept)    1.26      0.10    1.08    1.47 1.00      825    1570
#>
#> Regression Coefficients:
#>      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> Intercept      1.57      0.13    1.31    1.82 1.01      513      920
#> type          -0.42      0.09   -0.59   -0.25 1.00      7457     6505
#> year          -0.14      0.01   -0.15   -0.13 1.00     10420     6334
#> type:year     -0.09      0.01   -0.11   -0.06 1.00      7405     6314
#>
#> Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
#> and Tail_ESS are effective sample size measures, and Rhat is the potential
#> scale reduction factor on split chains (at convergence, Rhat = 1).
```

# Convergence

```
plot(bayes.brms)
```

