# Rstanarm - easy implementation of Bayesian analysis

*Philip Dixon*

*February 17, 2019*

## Why consider a Baysian analysis?

- Jarad (Lunchinators, 8 Feb 2019): for the interpretation
  - Credible intervals are what we all want confidence intervals to be
- Other reasons
  - Account for all sources of uncertainty (more below)
  - Inference on derived quantities (see below)
  - Small sample inference for many problems (get rid of the z score)
  - Can easily fit more realistic biological models (hierarchical modeling, not discussed here)

## The Bayesian paradigm:

- prior + likelihood + data -> posterior

$$f(\theta \mid data) = \frac{f(data \mid \theta)\, g(\theta)}{\int f(data \mid \theta)\, g(\theta)}$$

The integral in the denominator can be very difficult! Could be very high dimensional.

Long ago (pre 1990): Bayes restricted to combinations of prior and model (likelihood) with analytic integrals (conjugate priors)
1990's: MCMC revolution. Can sample from the posterior distribution without integration
* Gibbs sampler, Metropolis-Hasting sampler

### How do I do a Bayesian analysis?

1. Think about how the data relate to my question(s)? i.e., what are the relevant parameters?
2. Write out an appropriate model. specifies how data relate to parameters
3. Think about what I believe before seeing the data (prior distribution(s) for parameters)

### Then

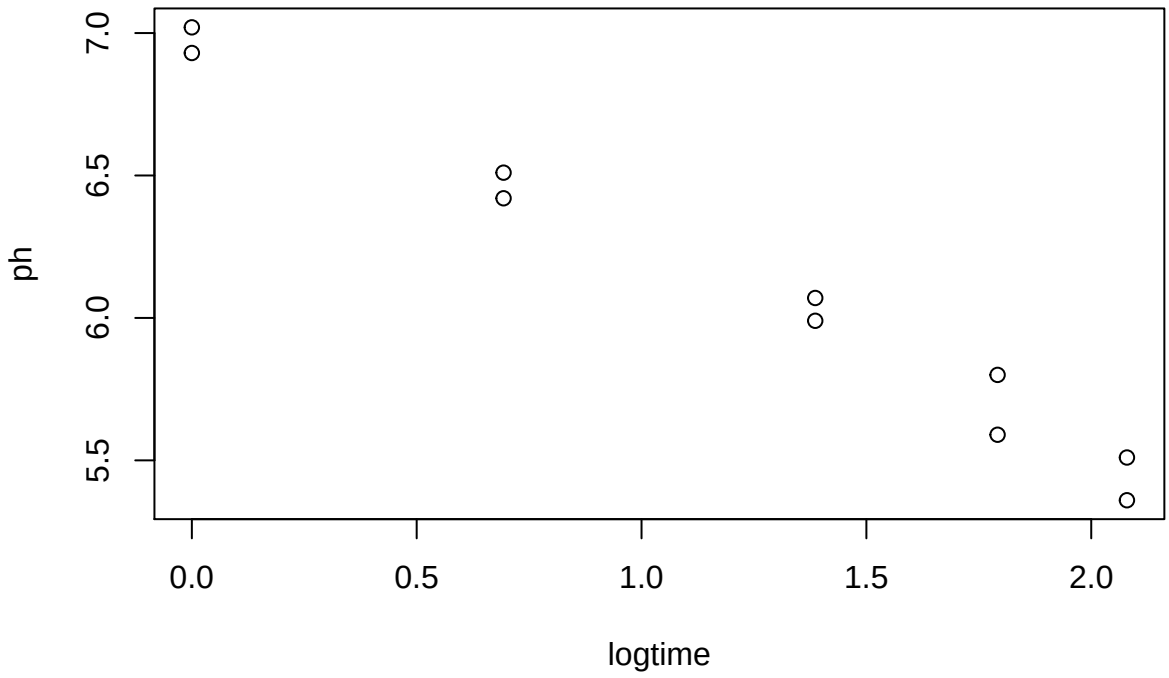
1. Do some math, write your own samplers
2. BUGS / WinBUGS: code the model and priors, requires loops
   - RWinBUGS: R interface to WinBUGS. Very inefficient.
3. JAGS: better implementation, still code the model and priors
   - rjags: R interface to JAGS - much smoother than RwinBUGS
4. STAN: new samplers (Hamiltonian MC), much faster!
   - RStan: R interface to STAN - I haven't used
   - rstanarm: uses R modeling language to write models

**rstanarm: Implements many R models, including**

1. Linear models
2. Generalized linear models
3. Linear mixed effect models
4. many others

## Simple example: Regress pH on logtime

```
meat <- read.table('meat.txt', header=T)
meat$logtime <- log(meat$time)
with(meat, plot(logtime, ph))
```




```
meat.lm <- lm(ph ~ logtime, data=meat)
summary(meat.lm)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  6.9836260 0.04853195 143.89749 6.083990e-15
## logtime     -0.7256578 0.03442633 -21.07857 2.695158e-08
```

```
confint(meat.lm)
```

```
##                 2.5 %     97.5 %
## (Intercept)  6.871711  7.0955409
## logtime     -0.805045 -0.6462705
```

```
meat.stan <- stan_glm(
    ph ~ logtime,
    family=gaussian,
    data=meat,
    chains = 4,
    cores = 4
    )
```

Explanation of code:
* stan modeling functions are stan_(R function), e.g., stan_lm, stan_glm, stan_lmer
* write model as you would write the R model, including data=
* glm with family=gaussian is an lm
* stan_lm requires priors (specify r^2 for each variable)
* stan_glm/gaussian allows different priors and has a reasonable default
* chains= specifies how many independent chains to sample, 3 or 4 are common choices
* cores= specifies how many cores to use for parallel processing. Do not exceed 1/2 to 2/3 number on your machine

How many cores does my machine have?

```
library(parallel)
detectCores()
```

```
## [1] 8
```

What are the default prior distributions?

```
prior_summary(meat.stan)
```

```
## Priors for model 'meat.stan'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##      **adjusted scale = 5.83
##
## Coefficients
##  ~ normal(location = 0, scale = 2.5)
##      **adjusted scale = 1.83
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
##      **adjusted scale = 0.58 (adjusted rate = 1/adjusted scale)
## ------
## See help('prior_summary.stanreg') for more details
```

Change by adding prior = (for regression coeff.) prior_intercept = (for intercept) and/or prior_aux = (for sigma) to the stan_glm call

stan_lm has a different set of default priors: based on expected r^2 for each variable. I find stan_glm more intuitive.

## What can I do once I fit the model?

**Diagnostics:**

- biggest concern is whether the sampler has converged to the posterior distribution

- default is 1000 samples "warmup" (discarded), keep next 1000 samples (both per chain)
- want chains to look similar
- Rhat measures discrepancy between chains. Want close to 1.
  - Over 1.1 is usually considered bad unless model really hard to sample
- how many samples: want small MC standard error
- Graphical exploration - rstanarm shiny app

```
launch_shinystan(meat.stan)
```

There are also posterior predictive checks. compare data to predictions from the posterior distribution.

### Summarize results, once fit looks reasonable

```
summary(meat.stan, digits=2)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      ph ~ logtime
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 10
##  predictors:   2
##
## Estimates:
##                 mean   sd    2.5%   25%   50%   75%   97.5%
## (Intercept)    6.98   0.06  6.87   6.95  6.98  7.02  7.10
## logtime       -0.73   0.04 -0.81  -0.75 -0.72 -0.70 -0.64
## sigma          0.10   0.03  0.06   0.08  0.09  0.11  0.17
## mean_PPD       6.12   0.04  6.03   6.09  6.12  6.15  6.21
## log-posterior  3.80   1.38  0.19   3.16  4.15  4.81  5.38
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.00 1.00 2053
## logtime       0.00 1.00 2310
## sigma         0.00 1.00 1715
## mean_PPD      0.00 1.00 2829
## log-posterior 0.04 1.00 1241
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

One caution:
* rstanarm centers all X variables - reduces correlation of estimates
- sampling the posterior is easier
* models without interactions or polynomial terms: centering only changes the intercept. rstanarm documentation says intercept estimates are adjusted back to uncentered version. * can turn off the centering if you want to - add sparse=TRUE to stan call.

This is not obvious. sparse= specifies whether the X'X matrix is sparse (lots of 0's), so estimates independent, which happens when centered. But sparse=TRUE means NOT sparse.

Can extract all the samples of the posterior distribution.
Very useful if you want a transformation of parameters, e.g. X when pH crosses 6.0

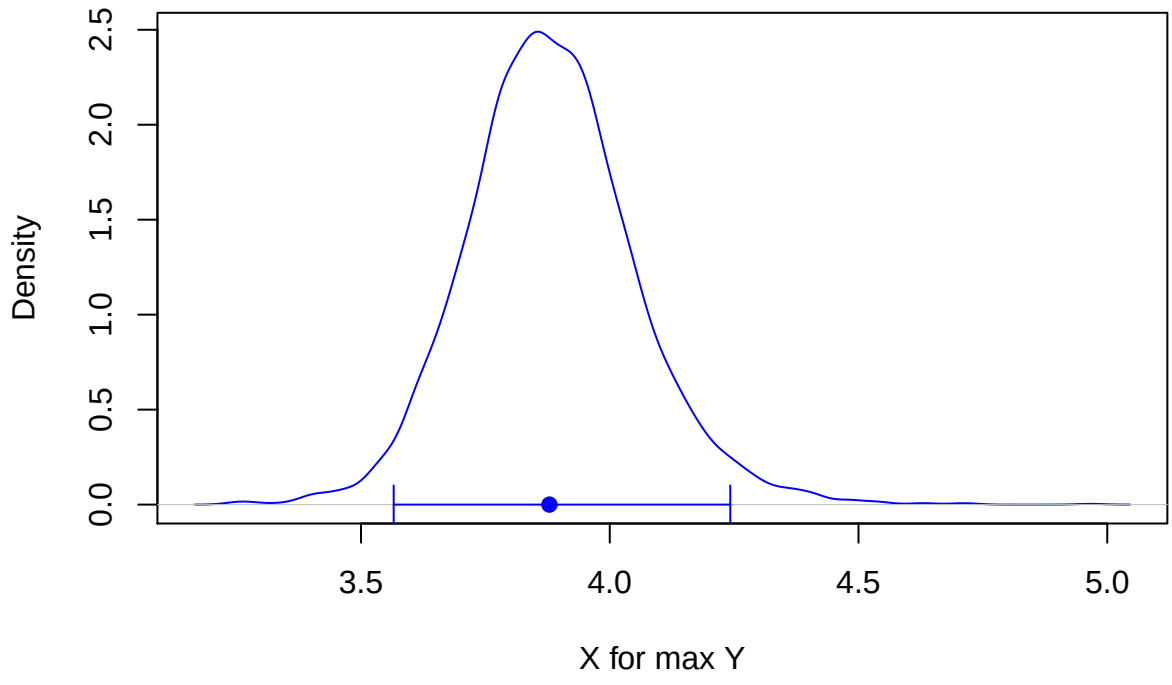
```
meat.post <- as.matrix(meat.stan)
meat.b0 <- meat.post[,1]
meat.b1 <- meat.post[,2]
time6 <- exp((6-meat.b0)/meat.b1)

plot(density(time6),  main='', xlab='X for max Y', col=4)
meat.beta <- coef(meat.lm)
points(exp((6-meat.beta[1])/meat.beta[2]), 0, pch=19, col=4)

summary(time6)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.246   3.773   3.875   3.883   3.982   4.965
```

```
quantile(time6, c(0.025, 0.05, 0.5, 0.95, 0.975) )
```

```
##     2.5%       5%      50%      95%    97.5%
## 3.565627 3.618325 3.874841 4.170393 4.242224
```

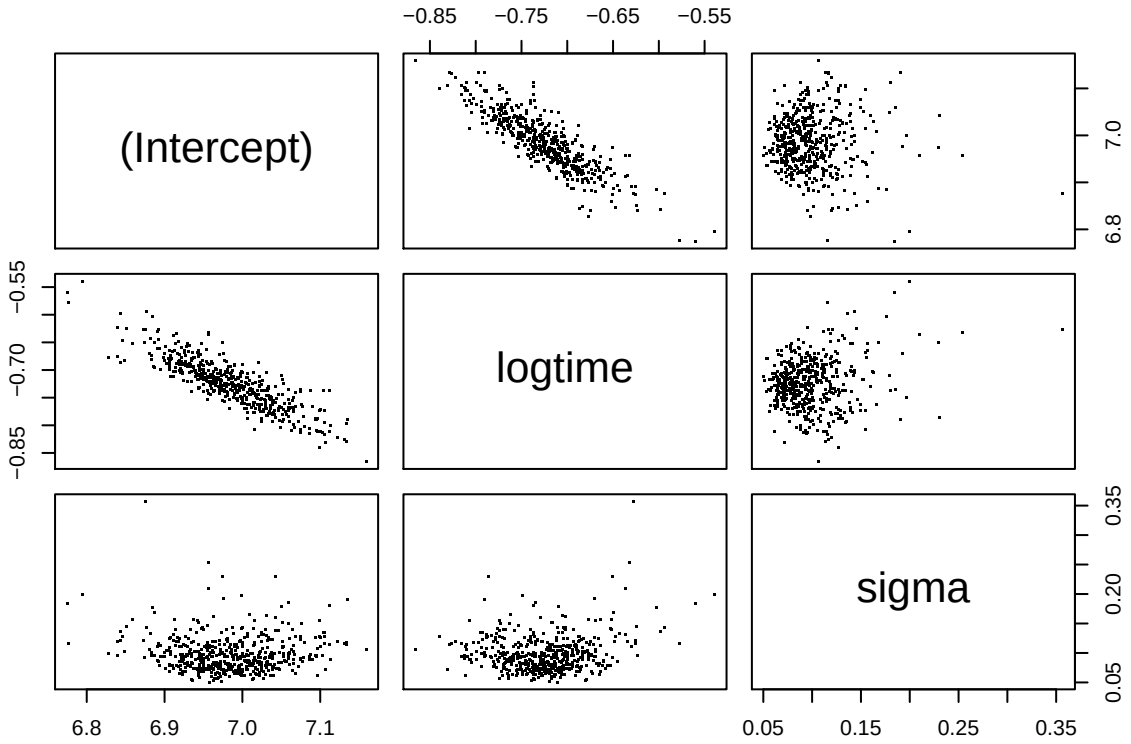
```
arrows(quantile(time6, 0.025), 0, quantile(time6, 0.975), 0,
  angle=90, length=0.1, code=3, col=4  )
```




Why does rstanarm want to center variables - look at correlations in the posterior distributions. 4000 samples,

only look at first 500.

```
pairs(meat.post[1:500,], pch='.')
```




# Examples of other models fit with rstanarm

## Generalized linear model

### benefit of Bayes - appropriate inferences for small samples

```
donner <- read.csv('donner.csv')
donner.glm <- glm(survival ~ age + femc, data=donner,     family=binomial)
summary(donner.glm)
```

```
##
## Call:
## glm(formula = survival ~ age + femc, family = binomial, data = donner)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7445  -1.0441  -0.3029   0.8877   2.0472
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.23041    1.38686   2.329   0.0198 *
## age         -0.07820    0.03728  -2.097   0.0359 *
```

```
## femcM       -1.59729    0.75547  -2.114   0.0345 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 61.827  on 44  degrees of freedom
## Residual deviance: 51.256  on 42  degrees of freedom
## AIC: 57.256
##
## Number of Fisher Scoring iterations: 4
```

```
confint(donner.glm)
```

```
## Waiting for profiling to be done...
```

```
##                  2.5 %      97.5 %
## (Intercept)  0.8514190  6.42669512
## age         -0.1624377 -0.01406576
## femcM       -3.2286705 -0.19510198
```

```
donner.stan <- stan_glm(
    survival ~ age + femc,
    family=binomial,
    data=donner,
    chains = 4,
    cores = 4
    )
summary(donner.stan, digits=2)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      survival ~ age + femc
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 45
##  predictors:   3
##
## Estimates:
##                 mean   sd     2.5%   25%    50%    75%    97.5%
## (Intercept)     3.44   1.41   0.88   2.47   3.33   4.33   6.48
## age            -0.09   0.04  -0.17  -0.11  -0.08  -0.06  -0.02
## femcM          -1.59   0.74  -3.13  -2.08  -1.57  -1.08  -0.18
## mean_PPD        0.44   0.09   0.27   0.38   0.44   0.51   0.62
## log-posterior -32.54   1.35 -35.91 -33.11 -32.20 -31.60 -31.06
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.03 1.00 3123
## age           0.00 1.00 2841
## femcM         0.01 1.00 3328
## mean_PPD      0.00 1.00 4033
```

```
## log-posterior 0.03 1.00 1503
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
# probability that a male more likely to die than a female of same age= P[femc < 0]
donner.post <- as.matrix(donner.stan)
mean(donner.post[,3] < 0)
```

```
## [1] 0.98725
```

## Incomplete blocks, with random block effects

### Benefit of Bayes - do not assume block variance is known exactly

```
ib <- read.csv('IBtest.csv')
ib.stan <- stan_lmer(
    y ~ trt.f + (1 | block.f),
    data=ib,
    chains = 4,
    cores = 4
    )
```

```
## Warning: There were 4 divergent transitions after warmup. Increasing adapt_delta above 0.95 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
ib.stan <- stan_lmer(
    y ~ trt.f + (1 | block.f),
    data=ib,
    chains = 4,
    cores = 4,
    adapt_delta = 0.98
    )

prior_summary(ib.stan)
```

```
## Priors for model 'ib.stan'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##      **adjusted scale = 9.07
##
## Coefficients
##  ~ normal(location = 0, scale = 2.5)
##      **adjusted scale = 2.27
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
##      **adjusted scale = 0.91 (adjusted rate = 1/adjusted scale)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

```
summary(ib.stan, digits=2,
  pars=c('(Intercept)', 'trt.fb', 'sigma'),
  regex_pars='Sigma*')
```

```
##
## Model Info:
##
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      y ~ trt.f + (1 | block.f)
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 80
##  groups:       block.f (60)
##
## Estimates:
##                                          mean   sd    2.5%   25%   50%
## (Intercept)                              0.06   0.14 -0.22  -0.04  0.06
## trt.fb                                   0.19   0.18 -0.17   0.07  0.19
## sigma                                    0.76   0.11  0.56   0.67  0.75
## Sigma[block.f:(Intercept),(Intercept)]  0.27   0.19  0.00   0.12  0.26
##                                          75%   97.5%
## (Intercept)                              0.16  0.34
## trt.fb                                   0.32  0.55
## sigma                                    0.84  0.99
## Sigma[block.f:(Intercept),(Intercept)]  0.40  0.66
##
## Diagnostics:
##                                         mcse Rhat n_eff
## (Intercept)                             0.00 1.00 3533
## trt.fb                                  0.00 1.00 5222
## sigma                                   0.01 1.01  382
## Sigma[block.f:(Intercept),(Intercept)] 0.01 1.01  343
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

## Overdispersed count data

### Benefit of Bayes - do not assume known amount of overdispersion

```
pod <- read.csv('PODtest.csv')
pod.glmm <- glmer(y ~ xc + (1|obs), data=pod, family=poisson)
summary(pod.glmm)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: y ~ xc + (1 | obs)
##    Data: pod
##
##      AIC      BIC   logLik deviance df.resid
##    345.6    349.8   -169.8    339.6       27
##
```

```
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.13140 -0.04849 -0.00471  0.02156  0.12427
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  obs    (Intercept) 2.241    1.497
## Number of obs: 30, groups:  obs, 30
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.8104     0.2800  13.610  < 2e-16 ***
## xc            0.8154     0.1818   4.486 7.26e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##    (Intr)
## xc -0.040
```

```
confint(pod.glmm)
```

```
## Computing profile confidence intervals ...
```

```
##                 2.5 %   97.5 %
## .sig01      1.1458084 2.045047
## (Intercept) 3.2219027 4.367875
## xc          0.4516029 1.196259
```

```
pod.stan <- stan_glmer(
    y ~ xc + (1 | obs),
    data=pod,
    family=poisson,
    chains = 4,
    cores = 4
    )
summary(pod.stan, digits=2,
  pars=c('(Intercept)', 'xc'),
  regex_pars='Sigma*')
```

```
##
## Model Info:
##
##  function:     stan_glmer
##  family:       poisson [log]
##  formula:      y ~ xc + (1 | obs)
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 30
##  groups:       obs (30)
##
## Estimates:
##                                mean   sd   2.5%   25%   50%   75%
## (Intercept)                   3.80   0.30 3.20   3.61  3.80  4.00
## xc                            0.80   0.19 0.41   0.67  0.80  0.92
```

```
## Sigma[obs:(Intercept),(Intercept)] 2.55   0.85 1.34   1.98  2.41  2.98
##                                      97.5%
## (Intercept)                         4.38
## xc                                  1.16
## Sigma[obs:(Intercept),(Intercept)] 4.62
##
## Diagnostics:
##                                     mcse Rhat n_eff
## (Intercept)                         0.01 1.00 728
## xc                                  0.01 1.00 812
## Sigma[obs:(Intercept),(Intercept)] 0.03 1.00 917
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

lmer() and glmer() condition on the estimated variance components.
I.e., inference on fixed effect parameters considers those variance components to be known precisely. Bayes accounts for that uncertainty
Some models (e.g. RCBD) variance has no effect on estimates, just uncertainty
Other models (e.g. OD, IB), different variances changes the estimates - here's where Bayes matters

## Final words:

the prior is an important part of the model: be critical of both
from the WinBUGS reference manual: BEWARE: MCMC sampling can be dangerous

## Resources:

- rstanarm:
  - Articles: Muth, Oravecz and Gabry (2018) User-friendly Bayesian regression modeling: a tutorial with rstandarm and shinystan. The Quantitative Methods for Psychology 14(2):99-119 with code at https://osf.io/ebz2f/
  - STAN project has wonderful vignettes about using rstanarm. Start with http://mc-stan.org/rstanarm/articles/rstanarm.html
  - Then look at the model-specific vignettes (vignettes tab)
    mixed models are in the Group Specific Terms vignette
- Bayes in Ecology: now lots of great books
  - Barker and Link: Bayesian Inference (my favorite)
  - McCarthy: Bayesian Methods for Ecology
  - King et al.: Bayesian Analysis for Population Ecology
  - Korner-Niervergelt et al.: Bayesian Analysis in Ecology using linear models with R, BUGS and Stan
  - Parent and Rivot: Introduction to Hierarchical Bayesian Modeling for Ecological Data