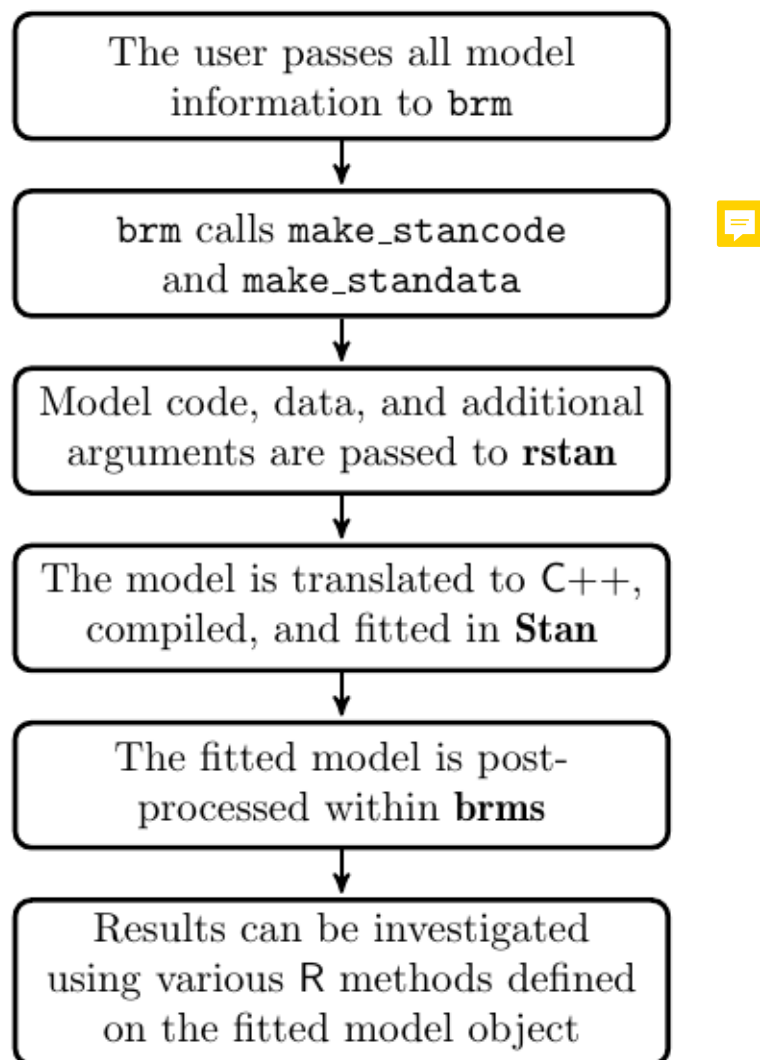


Generalized Linear Models with the brms R Package

Ben Goodrich

April 27, 2020


The brms Workflow (Figure 1 in Bürkner 2016)




The brms workflow

The Arguments to **brm**


```
library(brms)
args(brm)
```

```
## function (formula, data, family = gaussian(), prior = NULL, autocor = NULL,
##      data2 = NULL, cov_ranef = NULL, sample_prior = c("no", "yes", ,
##      "only"), sparse = NULL, knots = NULL, stanvars = NULL,
##      stan_funs = NULL, fit = NA, save_ranef = TRUE, save_mevars = FALSE,
##      save_all_pars = FALSE, inits = "random", chains = 4, iter = 2000,
##      warmup = floor(iter/2), thin = 1, cores = getOption("mc.cores",
##      1L), control = NULL, algorithm = c("sampling", "meanfield",
##      "fullrank"), future = getOption("future", FALSE), silent = TRUE,
##      seed = NA, save_model = NULL, stan_model_args = list(), save_dso = TRUE,
##      file = NULL, ...)
## NULL
```

The formula Argument to `brm`

- Everything to the right of the `~` is the same as in many other R functions
- In many cases, the thing to the left of the `~` is simply the outcome variable
- However, `brm` introduces a new possibility for this syntax like `y | fun(variable)`, where `fun` could be
 - `cens()` and `trunc()` to specify known censoring or truncation bounds
 - `weights()` and `disp()`, which should not be used with MCMC
 - `se()` to specify “known” standard errors in meta-analyses
 - `trials()`, which is used in binomial models only 
 - `cat()` to specify the possible categories for ordinal models

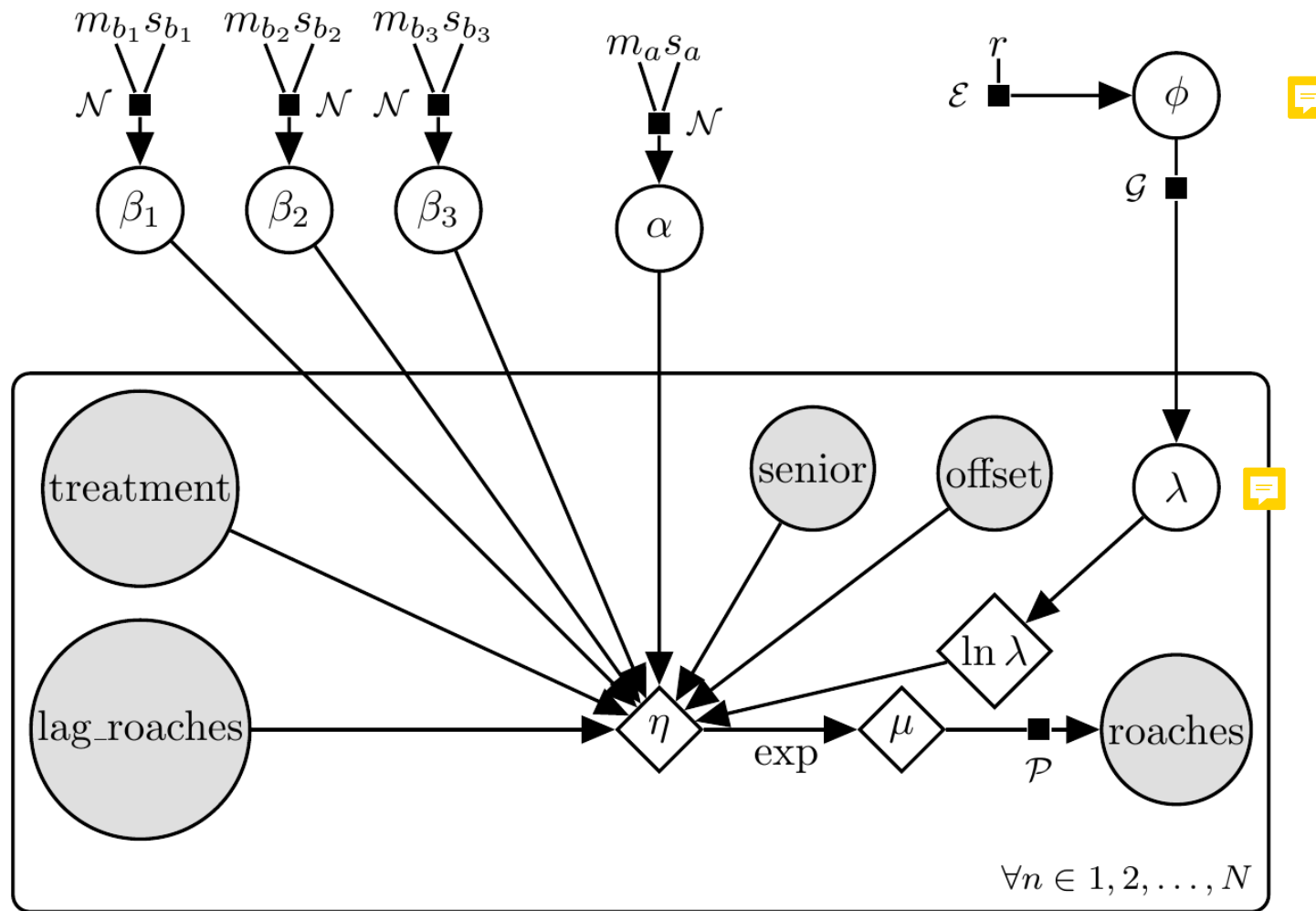
The **family** Argument to **brm**

The **family** argument can be any of the following functions, which also have a **link argument** that can be a variety of things depending on the family 

```
gaussian; student; binomial; bernoulli; poisson; negbinomial; geometric; Gamma;  
skew_normal; lognormal; shifted_lognormal; exgaussian; wiener; inverse.gaussian;  
exponential; weibull; frechet; Beta; dirichlet; von_mises; asym_laplace;  
gen_extreme_value; categorical; multinomial; cumulative; cratio; sratio; acat;  
hurdle_poisson; hurdle_negbinomial; hurdle_gamma; hurdle_lognormal;  
zero_inflated_binomial; zero_inflated_beta; zero_inflated_negbinomial;  
zero_inflated_poisson; zero_one_inflated_beta
```

- The ones involving **hurdle_**, **zero_inflated_** and / or **negbinomial** are of particular interest in the social sciences

Prior Predictive Distribution for Roach Study



Roach Model

Stan Code for this Prior Predictive Distribution

```
functions {  
  matrix roaches_PPD_rng(int S, vector log_roach1, vector treatment,  
    vector senior, vector offset) {  
    int N = rows(log_roach1);  
    matrix[S, N] PPD;  
    for (s in 1:S) {  
      real alpha = normal_rng(0, 5);  
      real beta[3] = normal_rng([0,0,0], 2);  
      real phi[N] = rep_array(exponential_rng(1), N);  
      real lambda[N] = gamma_rng(phi, phi);  
      vector[N] log_lambda = to_vector(log(lambda));  
      vector[N] eta = alpha + offset + beta[1] * log_roach1 +  
        beta[2] * treatment + beta[3] * senior + log_lambda;  
      vector[N] mu = exp(eta);  
      PPD[s, ] = to_row_vector(poisson_rng(mu));  
    }  
    return PPD;  
  }  
}
```

Breakout Rooms

```
data(roaches, package = "rstanarm"); roaches <- roaches[roaches$roach1 > 0, ]; str(roaches)
```

```
## 'data.frame':    202 obs. of  5 variables:
## $ y          : int  153 127 7 7 0 73 24 2 2 0 ...
## $ roach1      : num  308 331.25 1.67 3 2 ...
## $ treatment: int  1 1 1 1 1 1 1 0 0 0 ...
## $ senior     : int  0 0 0 0 0 0 0 0 0 0 ...
## $ exposure2: num  0.8 0.6 1 1 1.14 ...
```

- Call `rstan::expose_stan_functions` on a .stan file with the previous code
- Call `roaches_PPD_rng` with `S = 1`, `log_roach1 = log(roaches$roach1)`, and `offset = log(exposure2)`
- Is this prior predictive distribution of roaches reasonable in the sense that it is similar to `roaches$roach1`?
- If not, modify the priors in the previous code to make the prior predictive distribution reasonable

Integral Leading to Negative Binomial

- Now let $\eta_n = \alpha + \sum_{k=1}^K \beta_k x_{nk}$ without $\log \lambda_n$
- Poisson likelihood is
 $\mathcal{L}(\eta_n, \lambda_n; y_n) \propto (\lambda_n e^{\eta_n})^{y_n} e^{-\lambda_n e^{\eta_n}} = \lambda_n^{y_n} (e^{\eta_n})^{y_n} e^{-\lambda_n e^{\eta_n}}$
- Gamma prior is $f(\lambda_n | \phi) \propto \lambda_n^{\phi-1} e^{-\phi \lambda_n}$
- Integrating λ_n out of the product yields

$$\int_0^\infty \lambda_n^{y_n} (e^{\eta_n})^{y_n} e^{-\lambda_n e^{\eta_n}} \lambda_n^{\phi-1} e^{-\phi \lambda_n} d\lambda_n \propto$$
$$\binom{y_n + \phi - 1}{y_n} \left(\frac{e^{\eta_n}}{e^{\eta_n} + \phi} \right)^{y_n} \left(\frac{\phi}{e^{\eta_n} + \phi} \right)^\phi$$

which is the PMF of the negative binomial distribution

Equivalent Prior Predictive Distribution

```
functions {  
  matrix roaches_PPD_rng(int S, vector log_roach1, vector treatment,  
                        vector senior, vector offset) {  
    int N = rows(log_roach1);  
    matrix[S, N] PPD;  
    for (s in 1:S) {  
      real alpha = normal_rng(0, 5);  
      real beta[3] = normal_rng([0,0,0], 2);  
      real phi = exponential_rng(1);  
      vector[N] eta = alpha + offset + beta[1] * log_roach1 +  
        beta[2] * treatment + beta[3] * senior;  
      PPD[s, ] = to_row_vector(neg_binomial_2_log_rng(eta, phi));  
    }  
    return PPD;  
  }  
}
```

The **prior** Argument to **brm**

```
args(set_prior) # or just prior()
```

```
## function (prior, class = "b", coef = "", group = "", resp = "",  
##      dpar = "", nlpar = "", lb = NA, ub = NA, check = TRUE)  
## NULL
```

- **prior** is a character string (in the Stan language) such as `"normal(0,5)"`
- **class** indicates what parameters the call to **set_prior** pertains to
- **coef** is the name of the parameter in question
- **group** is the name of the grouping factor (if applicable)
- **resp** is the name of the response variable in multivariate models
- **dpar** is the name of the distribution parameter (if applicable)
- **nlpar** is the name of the non-linear parameter (if applicable)
- **lb** is the lower bound of the parameter (default $-\infty$)
- **ub** is the upper bound of the parameter (default ∞)
- **check** whether priors should be checked for validity

The `get_prior` Function

- Input the `formula`, `data`, and `family` and get back the possible prior choices (and defaults)

```
get_prior(y ~ log(roach1) + treatment + senior + offset(log(exposure2)),  
          data = roaches, family = negbinomial)
```

##	prior	class	coef	group	resp	dpar	nlpar	bound
## 1		b						
## 2		b	logroach1					
## 3		b	senior					
## 4		b	treatment					
## 5	student_t(3, 2, 10)	Intercept						
## 6	gamma(0.01, 0.01)	shape						



The `class` Argument to `set_prior`

- Refers to a type of parameter in the model
- Defaults to `"b"` which refers to (population-level) regression coefficients
- Other possible values are `"Intercept"`, `"sd"`, `"cor"`, `"sigma"` and others we may talk about later

```
my_prior <- prior(cauchy(0, 1), class = "b")
```

- If you call `prior` rather than `set_prior`, the first argument can be an R expression rather than a character string

Example of brm



```
post <- brm(y ~ log(roach1) + treatment + senior + offset(log(exposure2)),  
  data = roaches, family = negbinomial, prior = prior(normal(0, 2), class = "b") +  
  prior(normal(0, 5), class = "Intercept") + prior(exponential(1), class = "shape"))
```

post



```
...  
##  
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS  
## Intercept      1.33      0.26      0.85      1.83 1.00      4632      3367  
## logroach1       0.70      0.06      0.57      0.82 1.00      4605      3291  
## treatment      -0.62      0.21     -1.04     -0.20 1.00      4659      3060  
## senior         -0.20      0.25     -0.66      0.29 1.00      4576      3133  
##
```

Family Specific Parameters:


```
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS  
## shape         0.47      0.05      0.38      0.58 1.00      4659      3011
```




```
## Samples were drawn 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).
```

...

Using the `loo` Function

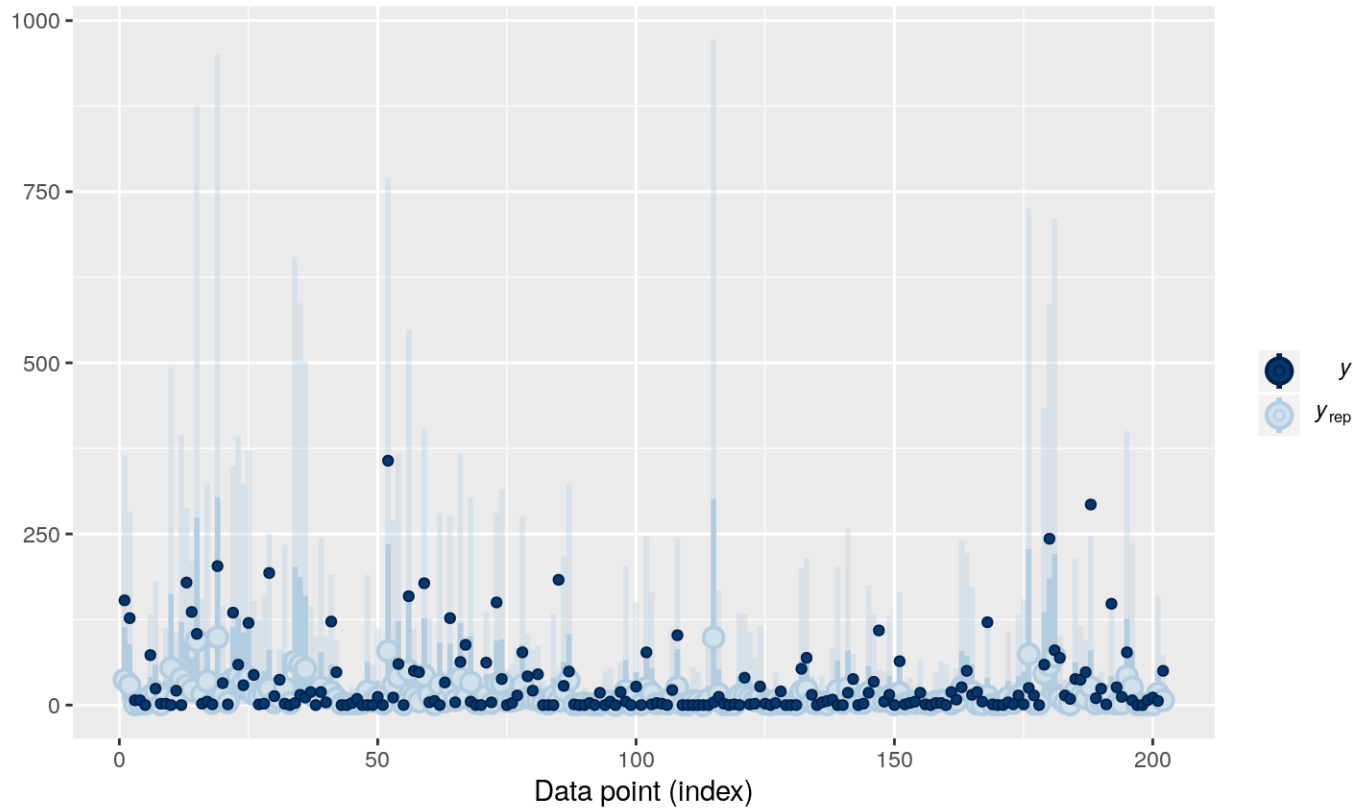
- McElreath cautions against using things like `loo` when λ_n is included
- If λ_n is integrated out of the posterior distribution by using a negative binomial likelihood, everything's fine (unless there are warnings, in which case `reloo`) 

```
loo_post <- loo(post, reloo = TRUE) # observation 85 has a Pareto k > 0.7
loo_post
```

```
##
## Computed from 4000 by 202 log-likelihood matrix
##
##           Estimate    SE
## elpd_loo   -759.3  28.3
## p_loo       5.3   1.5 
## looic      1518.6  56.6
## -----
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##
##           Count Pct.    Min. n_eff
## (-Inf, 0.5] (good)   201   99.5%   1110
## (0.5, 0.7] (ok)      1    0.5%    293
```

Using the `pp_check` Function

```
pp_check(post, type = "loo_intervals") # type is the same as plotfun with rstanarm
```



Using the `hypothesis` Function

- To do this with `rstanarm`, you would have to first call `as.matrix`

```
args(brms::hypothesis.brmfit)
```

```
## function (x, hypothesis, class = "b", group = "", scope = c("standard",  
##      "ranef", "coef"), alpha = 0.05, seed = NULL, ...)  
## NULL
```

- Here `x` is the object produced by `brm` and `hypothesis` is a string, typically with an embedded `<` or `>`, such as

```
hypothesis(post, "treatment < 0")
```

```
## Hypothesis Tests for class b:  
##      Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star  
## 1 (treatment) < 0    -0.62      0.21    -0.98    -0.27      399          1      *  
## ---  
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.  
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;  
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.  
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

Other Post-Estimation Methods

Many of the things you can do with an object produced by **brm** are analagous to **rstanarm**

##	[,1]	[,2]	[,3]
##	[1,] "add_criterion"	"loo_compare"	"posterior_samples"
##	[2,] "add_ic"	"loo_linpred"	"posterior_summary"
##	[3,] "as.array"	"loo_model_weights"	"pp_average"
##	[4,] "as.data.frame"	"loo_predict"	"pp_check"
##	[5,] "as.matrix"	"loo_predictive_interval"	"pp_expect"
##	[6,] "as.mcmc"	"loo_R2"	"pp_mixture"
##	[7,] "autocor"	"loo_subsample"	"predict"
##	[8,] "bayes_factor"	"loo"	"predictive_error"
##	[9,] "bayes_R2"	"L00"	"predictive_interval"
##	[10,] "bridge_sampler"	"marginal_effects"	"print"
##	[11,] "coef"	"marginal_smooths"	"prior_samples"
##	[12,] "conditional_effects"	"mcmc_plot"	"prior_summary"
##	[13,] "conditional_smooths"	"model_weights"	"ranef"
##	[14,] "control_params"	"model.frame"	"reloo"
##	[15,] "expose_functions"	"neff_ratio"	"residuals"
##	[16,] "extract_draws"	"ngrps"	"rhat"
##	[17,] "family"	"nobs"	"stancode"
##	[18,] "fitted"	"nsamples"	"standata"
##	[19,] "fixef"	"nuts_params"	"stanplot"
##	[20,] "formula"	"pairs"	"summary"
##	[21,] "getCall"	"parnames"	"update"
##	[22,] "hypothesis"	"plot"	"VarCorr"
##	[23,] "kfold"	"post_prob"	"vcov"
##	[24,] "launch_shinystan"	"posterior_average"	"waic"
##	[25,] "log_lik"	"posterior_interval"	"WAIC"
##	[26,] "log_posterior"	"posterior_linpred"	" "
##	[27,] "logLik"	"posterior_predict"	" "

Breakout Rooms: Hurdle Models

- Hurdle models combine a logit model for whether there is a positive number of roaches in a building with a negative binomial model for the number of roaches, conditional on there being at least 1 roach
- Augment previous Stan code to draw from its prior predictive distribution of a hurdle model
- Hints: You are going to have to loop from 1 to N and do the two parts inside an inner loop instead of vectorizing the whole thing. Also, you will need a while loop to enforce the constraint that the draw from the negative binomial distribution is not zero.

```
get_prior(brms::bf(y ~ log(roach1) + treatment + senior + offset(log(exposure2)),  
                hu ~ I(roach1 == 0) + treatment + senior), data = roaches,  
          family = hurdle_negbinomial)
```

##	prior	class	coef	group	resp	dpar	nlpar	bound
## 1		b						
## 2		b	logroach1					
## 3		b	senior					
## 4		b	treatment					
## 5	student_t(3, 2, 10)	Intercept						
## 6	gamma(0.01, 0.01)	shape						
## 7		b				hu		
## 8		b	Iroach1EQEQ0TRUE			hu		
## 9		b	senior			hu		
## 10		b	treatment			hu		
## 11	logistic(0, 1)	Intercept				hu		

Hurdle Models with **brm**

```
post_hurdle <- brm(brms::bf(y ~ log(roach1) + treatment + senior + offset(log(exposure2)),  
                    hu ~ I(roach1 == 0) + treatment + senior), data = roaches,  
family = hurdle_negbinomial, seed = 12345, prior =  
  prior(normal(0, 2), class = "b") +  
  prior(normal(0, 5), class = "Intercept") +  
  prior(exponential(1), class = "shape") +  
  prior(normal(0, 2), class = "b", dpar = "hu"))
```

```
## Compiling the C++ model
```

```
## Start sampling
```

Results of Hurdle Model

post_hurdle

```
...
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept          1.64      0.28    1.10    2.17 1.00    6474    2918
## hu_Intercept       -1.59      0.30   -2.20   -1.03 1.00    7704    3057
## logroach1           0.61      0.07    0.48    0.74 1.00    6325    2624
## treatment          -0.50      0.21   -0.94   -0.09 1.00    7225    2604
## senior             -0.04      0.27   -0.55    0.51 1.00    7097    2985
## hu_Iroach1EQEQ0TRUE -0.05      1.98   -3.86    3.82 1.00    7064    3017
## hu_treatment        0.39      0.36   -0.31    1.09 1.00    7444    3259
## hu_senior           0.81      0.35    0.13    1.49 1.00    5904    2728
##
## Family Specific Parameters:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## shape          0.64      0.11    0.44    0.87 1.00    5899    2838
##
## Samples were drawn 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).
...
```

PSISLOOCV Comparison

```
loo(post, post_hurdle, reloo = TRUE)
```

```
## Output of model 'post':
##
## Computed from 4000 by 202 log-likelihood matrix
##
##           Estimate   SE
## elpd_loo    -759.3 28.3
## p_loo         5.3  1.5
## looic        1518.6 56.6
## -----
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##           Count Pct.    Min. n_eff
## (-Inf, 0.5] (good)    201   99.5%    1110
## (0.5, 0.7]  (ok)       1    0.5%     293
## (0.7, 1]   (bad)       0    0.0%    <NA>
## (1, Inf)   (very bad)  0    0.0%    <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
##
## Output of model 'post_hurdle':
```

```
##
## Computed from 4000 by 202 log-likelihood matrix
##
##           Estimate   SE
## elpd_loo    -773.0 27.9
## p_loo         8.1  1.5
## looic        1546.0 55.9
## -----
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##           Count Pct.    Min. n_eff
## (-Inf, 0.5] (good)    201   99.5%    3096
## (0.5, 0.7]  (ok)       1    0.5%     366
## (0.7, 1]   (bad)       0    0.0%    <NA>
## (1, Inf)   (very bad)  0    0.0%    <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
##
## Model comparisons:
##           elpd_diff se_diff
## post           0.0      0.0
## post_hurdle -13.7      4.9
```

Simplexes

- Let X_1, X_2, \dots, X_K be defined for a sample space Ω or a parameter space Θ such that $X_k \geq 0 \forall k$ and $\sum_{k=1}^K X_k = 1$
- Then X_1, X_2, \dots, X_K are said to be a simplex, which is essentially a PMF
- The constraint that $\sum_{k=1}^K X_k = 1$ implies X_i is NOT independent of X_j
- The cumulative sum of a simplex is often useful and implies the last element is exactly 1 while all the previous elements are between 0 and 1

Dirichlet Distribution

- Dirichlet distribution is a PDF over PMFs that has the following form

$$f(\boldsymbol{\pi} | \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^K \pi_k^{\alpha_k - 1}$$

where $\alpha_k \geq 0 \forall k$ and the multivariate Beta function is $B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^K \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^K \alpha_k)}$

where $\Gamma(z) = \int_0^\infty u^{z-1} e^{-u} du$ is the Gamma function, which is implemented in R as `gamma` and Stan as `tgamma`

- $\mathbb{E}\pi_i = \frac{\alpha_i}{\sum_{k=1}^K \alpha_k} \forall i$ and the mode of π_i is $\frac{\alpha_i - 1}{-1 + \sum_{k=1}^K \alpha_k}$ if $\alpha_i > 1$
- Iff $\alpha_k = 1 \forall k$, $f(\boldsymbol{\pi} | \boldsymbol{\alpha} = \mathbf{1})$ is constant over Θ (simplexes)
- Dirichlet distribution is conjugate with the multinomial and categorical

Categorical Distribution

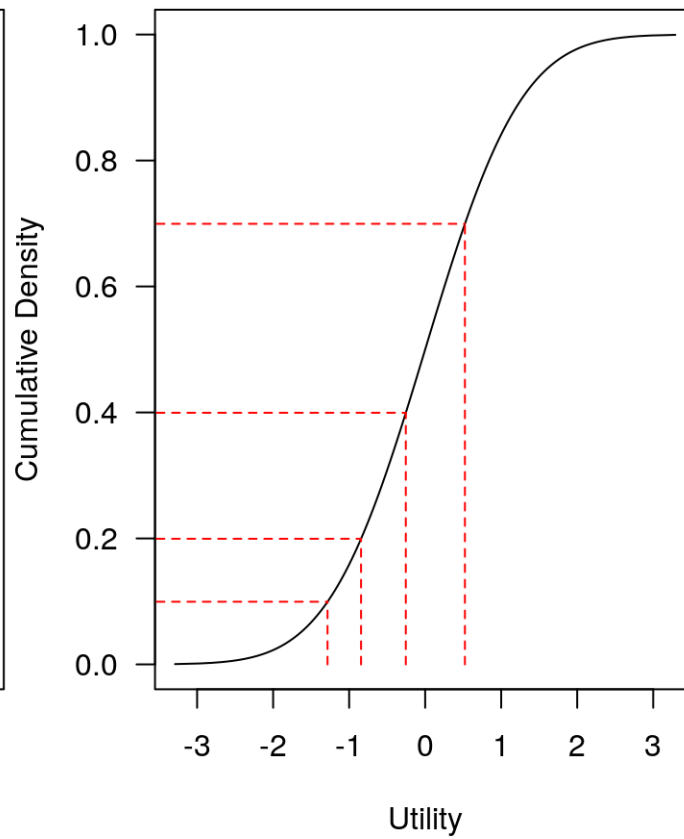
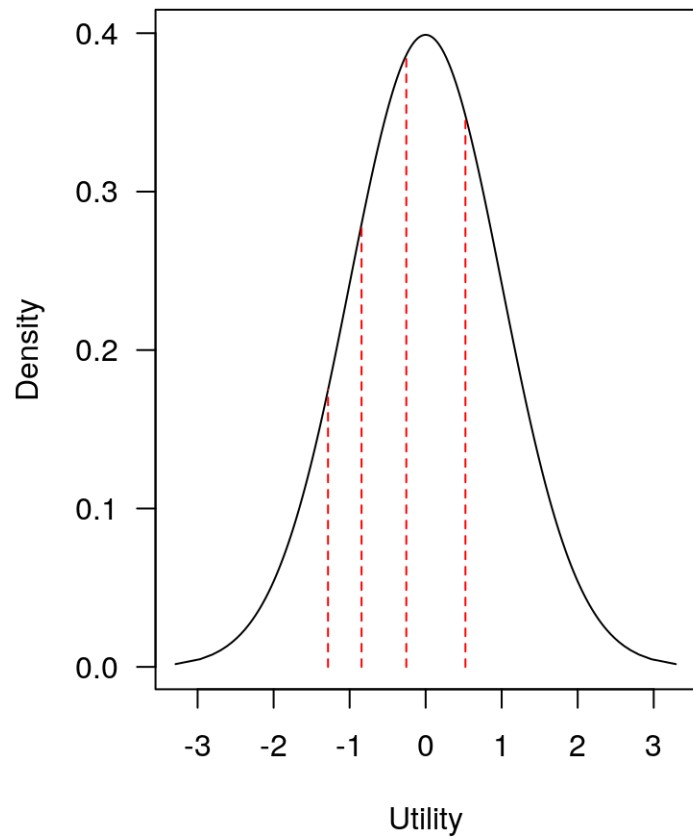


- The categorical distribution over $\Omega = \{1, 2, \dots, K\}$ has a PMF $\Pr(x | \pi_1, \pi_2, \dots, \pi_K) = \prod_{k=1}^K \pi_k^{\mathbb{I}(x=k)}$ where the parameters satisfy
 1. $\pi_k \geq 0 \forall k$
 2. $\sum_{k=1}^K \pi_k = 1$
- The categorical distribution is a generalization of the Bernoulli distribution to the case where there are K categories rather than merely failure vs. success
- To draw randomly from it, you can do `sample(0mega, size = 1, prob = c(pi_1, pi_2, ..., pi_K))`
- You can make each π_k a function of predictors in a regression model

Multinomial Distribution

- The multinomial distribution over $\Omega = \{0, 1, \dots, n\}$ has a PMF $\Pr(x | \pi_1, \pi_2, \dots, \pi_K) = n! \prod_{k=1}^K \frac{\pi_k^{x_k}}{x_k!}$ where the parameters satisfy $\pi_k \geq 0 \forall k, \sum_{k=1}^K \pi_k = 1$, and $n = \sum_{k=1}^K x_k$
- The multinomial distribution is a generalization of the binomial distribution to the case that there are K possibilities rather than merely failure vs. success
- The multinomial distribution is the count of n independent categorical random variables with the same π_k values
- Can draw from it with `rmultinom(1, size = n, prob = c(pi_1, pi_2, ..., pi_K))`
- Categorical is a special case where $n = 1$

Graphs of Standard Normal Utility with Cutpoints



Likelihood for an Ordered Observation

- Likelihood for an observation is just categorical:

$$\mathcal{L}(\beta, \zeta; y) \propto \prod_{j=1}^J \Pr(y = j | \beta, \zeta)$$

- If $F(\cdot)$ is in the location-scale family (normal, logistic, etc.), then $F(\beta x + \epsilon \leq \zeta_j) = F_{0,1}(\zeta_j - \beta x)$, where $F_{0,1}(\cdot)$ is the “standard” version of the CDF
- $\Pr(y = j | \beta, \zeta) = F(\beta x + \epsilon \leq \zeta_j) - F(\beta x + \epsilon \leq \zeta_{j-1})$

Prior Predictive Distribution of an Ordinal Model

$$\forall k : \beta_k \sim ???$$



$$\forall n : \eta_n = \sum_{k=1}^K \beta_k x_{nk}$$

$$\zeta_1 \sim ???$$

$$\forall 1 < j < J - 1 : \zeta_j - \zeta_{j-1} \sim ???$$

$$\forall n : \epsilon_n \sim \mathcal{N}(0, 1)$$



$$\forall n : u_n = \eta_n + \epsilon_n$$

$$\forall n : y_n = 1 + \sum_{j=1}^{J-1} \mathcal{I}\{u_n > \zeta_j\}$$



Estimating an Ordinal Model with `stan_polr`

```
library(rstanarm); options(mc.cores = parallel::detectCores())
data("inhaler", package = "brms")
inhaler$rating <- as.ordered(inhaler$rating)
post <- stan_polr(rating ~ treat + period + carry, data = inhaler,
                  method = "probit", prior = R2(0.25), seed = 12345)
```

- Now we can estimate the causal effect of `treat` on utility for rating:

```
nd <- inhaler; nd$treat <- 1
y1_star <- posterior_linpred(post, newdata = nd)
nd$treat <- 0
y0_star <- posterior_linpred(post, newdata = nd)
summary(c(y1_star - y0_star))
```



```
##      Min.  1st Qu.   Median     Mean  3rd Qu.    Max.
## -0.99207 -0.58247 -0.48490 -0.48631 -0.38750  0.06511
```

Results

```
...  
## -----  
##           Median MAD_SD  
## treat   -0.48   0.14  
## period   0.12   0.11  
## carry   -0.12   0.10  
##  
## Cutpoints:  
##           Median MAD_SD  
## 1|2  0.33   0.05  
## 2|3  1.77   0.10  
## 3|4  2.28   0.13  
##  
...
```



Similar Models with **brm**

- **brm** can estimate similar models, but with priors on the coefficients

```
po <- brm(rating ~ treat + period + carry,  
          data = inhaler, family = cumulative) # similar to rstanarm::stan_polr  
sr <- brm(formula = rating ~ period + carry + cs(treat), data = inhaler,  
          family = sratio, prior = prior(normal(-1, 2), coef = "treat"))
```

- Latter model considers when a person “stops” and allows the effect of **treat** to vary across categories of **rating**

loo to the Rescue



```
loo(po, sr)
```

```
## Output of model 'po':  
##  
## Computed from 4000 by 572 log-likelihood matrix  
##  
##           Estimate    SE  
## elpd_loo    -459.0 17.2  
## p_loo         6.0  0.5  
## looic        917.9 34.5  
## -----  
## Monte Carlo SE of elpd_loo is 0.0.  
##  
## All Pareto k estimates are good (k < 0.5).  
## See help('pareto-k-diagnostic') for details.  
##  
## Output of model 'sr':
```

```
##  
## Computed from 4000 by 572 log-likelihood matrix  
##  
##           Estimate    SE  
## elpd_loo    -459.5 17.5  
## p_loo         8.1  1.2  
## looic        919.1 35.1  
## -----  
## Monte Carlo SE of elpd_loo is 0.1.  
##  
## All Pareto k estimates are good (k < 0.5).  
## See help('pareto-k-diagnostic') for details.  
##  
## Model comparisons:  
##      elpd_diff se_diff  
## po   0.0         0.0  
## sr -0.6         2.0
```

Results of Cumulative Model

po

```
...  
##  
##      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS  
## Intercept[1]    0.55    0.09    0.38    0.72 1.00    5398    3477  
## Intercept[2]    3.23    0.20    2.85    3.62 1.00    6138    2803  
## Intercept[3]    4.49    0.36    3.84    5.23 1.00    5894    3050  
## treat          -0.80    0.25   -1.29   -0.31 1.00    3292    3195  
## period          0.18    0.18   -0.15    0.54 1.00    5089    2794  
## carry          -0.22    0.18   -0.58    0.14 1.00    3054    3110  
##  
## Samples were drawn 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).  
...
```



Results of Stopping Ratio Model

sr

```
...  
##  
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS  
## Intercept[1]      0.55      0.09      0.38      0.73 1.00      4905      2942  
## Intercept[2]      2.40      0.29      1.88      3.01 1.00      3198      2431  
## Intercept[3]      0.68      0.56     -0.39      1.81 1.00      3754      3325  
## period            0.22      0.17     -0.11      0.55 1.00      3985      2843  
## carry             -0.21      0.17     -0.53      0.11 1.00      3739      3196  
## treat[1]          -0.78      0.23     -1.24     -0.34 1.00      3790      2767  
## treat[2]          -1.08      0.57     -2.29     -0.04 1.00      3047      2443  
## treat[3]           0.61      1.02     -1.42      2.56 1.00      3482      3122  
##  
## Samples were drawn 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).  
...
```



Models with Ordinal Predictors

- Often but not always it is reasonable to assume that the coefficients on dummy variables derived from the same ordered factor are **monotonic**

```
theta <- MCMCpack::rdirichlet(n = 1, alpha = c(1, 1, 1))[1, ] # PDF over PMFs
rbind(theta, cumsum(theta))
```

```
##           [,1]      [,2]      [,3]
## theta 0.1071258 0.7715094 0.1213648
##       0.1071258 0.8786352 1.0000000
```

```
gamma <- rnorm(n = 1)
beta <- gamma * cumsum(theta)
```

- This is what **brms** does when you use `mo(ordered_factor)` on the right-hand side of a formula and put a standard normal prior on the scale factor
- For more examples, see https://cran.r-project.org/package=brms/vignettes/brms_monotonic.html

Ordinal Predictors in Polling

```
poll <- readRDS("GooglePoll.rds") # WantToWin is coded as 1 for Romney and 0 for Obama
library(dplyr)
collapsed <- filter(poll, !is.na(WantToWin)) %>%
  group_by(Region, Gender, Urban_Density, Age, Income) %>%
  summarize(Romney = sum(grepl("Romney", WantToWin)), Obama = n() - Romney) %>%
  na.omit
```



```
post <- brm(Romney | trials(Romney + Obama) ~ Region + Gender + Urban_Density +
  # Age and Income are restricted to have monotonic effects
  mo(Age) + mo(Income), data = collapsed, family = binomial)
```

Results of Model with Ordinal Predictors

```
...  
##  
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS  
## Intercept          -0.63      0.12   -0.86   -0.39 1.00     2795     2754  
## RegionNORTHEAST     -0.13      0.09   -0.31    0.04 1.00     3250     2845  
## RegionSOUTH         0.31      0.07    0.17    0.45 1.00     3218     2256  
## RegionWEST          -0.14      0.08   -0.29    0.00 1.00     2901     3038  
## GenderMale          0.39      0.06    0.28    0.50 1.00     4238     2983  
## Urban_DensitySuburban -0.19      0.09   -0.36   -0.02 1.00     3239     2343  
## Urban_DensityUrban   -0.50      0.09   -0.68   -0.32 1.00     3071     2643  
## moAge               0.27      0.02    0.23    0.31 1.00     3304     2991  
## moIncome            0.01      0.06   -0.09    0.15 1.00     2007     1465
```

##

Simplex Parameters:

```
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS  
## moAge1[1]         0.09      0.05    0.01    0.21 1.00     2677     1634  
## moAge1[2]         0.31      0.07    0.17    0.46 1.00     3641     2696  
## moAge1[3]         0.21      0.07    0.07    0.35 1.00     3426     2107  
## moAge1[4]         0.05      0.04    0.00    0.14 1.00     3021     1821  
## moAge1[5]         0.34      0.06    0.22    0.44 1.00     4552     3398  
## moIncome1[1]      0.18      0.17    0.00    0.61 1.00     2630     2155  
## moIncome1[2]      0.15      0.14    0.00    0.53 1.00     2591     1561  
## moIncome1[3]      0.18      0.15    0.01    0.56 1.00     3729     2463  
## moIncome1[4]      0.22      0.18    0.01    0.65 1.00     3508     2236  
## moIncome1[5]      0.25      0.20    0.01    0.73 1.00     3102     2386
```

##

Samples were drawn 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).

...

Effects Plot

```
plot(conditional_effects(post, effects = "Age"))
```

