Hierarchical Models with the rstanarm and brms Packages

Ben Goodrich

April 27, 2022

Frequentist Estimation of Multilevel Models

- Frequentists assume that a_j and b_j deviate from the common parameters according to a (multivariate) normal distribution, whose (co)variances are common parameters to be estimated
- To Frequentists, a_j and b_j are not parameters because parameters must remained fixed in repeated sampling of observations from some population
- Since a_j and b_j are not parameters, they can't be "estimated" only "predicted"
- Since a_j and b_j aren't estimated, they must be integrated out of the likelihood function, leaving an integrated likelihood function of the common parameters
- After obtaining maximum likelihood estimates of the common parameters, each a_j and b_j can be predicted from the residuals via a regression
- Estimated standard errors produced by frequentist software are too small
- There are no clearly-defined standard errors for the a_j and b_j
- Maximum likelihood estimation often results in a corner solution

Table 2 from the lme4 Vignette (see also the FAQ)

Formula	Alternative	Meaning
`(1 | g)`	`1 + (1 | g)`	Random intercept with fixed mean
`0 + offset(o) + (1 | g)`	`-1 + offset(o) + (1 | g)`	Random intercept with *a priori* means
`(1 | g1/g2)`	`(1 | g1)+(1 | g1:g2)`	Intercept varying among `g1` and `g2` within `g1`
`(1 | g1)+(1 | g2)`	`1 + (1 | g1) + (1 | g2)`	Intercept varying among `g1` and `g2`
`x + (x | g)`	`1 + x + (1 + x | g)`	Correlated random intercept and slope
`x + (x || g)`	`1 + x + (1 | g) + (0 + x | g)`	Uncorrelated random intercept and slope

Table 2: Examples of the right-hand sides of mixed-effects model formulas. The names of grouping factors are denoted `g`, `g1`, and `g2`, and covariates and *a priori* known offsets as `x` and `o`.

lme4 syntax

Hierarchical Models in Psychology

- In political science and economics, the “big” units are often countries or sub-national political areas like states and the “small” units are people
- In psychology, the “big” units are often people and the “small” units are questions or outcomes on repeated tasks
- Hierarchical model syntax is like

```
y ~ x + (x | person) + (1 | question)
```

- Question of interest is how to predict y for a new “big” unit (person) and Bayesian hierarchical models do much better than anything else

Hierarchical Models in rstanarm (from this paper)

```
dat <- readr::read_csv("https://osf.io/5cg32/download")
library(rstanarm)
options(mc.cores = parallel::detectCores())
```

```
post <-  stan_glmer(valence ~ arousal + (1 + arousal | PID), data = dat,
                    prior = normal(0, 1), prior_intercept = normal(50, 100))
```

```
post
```

```
...
##             Median MAD_SD
## (Intercept) 30.1    5.3
## arousal      0.5    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 9.3    0.4
```

```
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  PID      (Intercept) 20.60
##           arousal      0.24    -0.65
##  Residual              9.28
## Num. levels: PID 20
##
...
```

PSISLOOCV (within a group)

```
loo(post)


...
##          Estimate   SE
## elpd_loo  -1008.5 13.7
## p_loo        30.8  3.1
## looic      2017.1 27.4
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     264   97.1%   661
##  (0.5, 0.7]   (ok)         8    2.9%   434
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
...
```

Accessor Functions (based on the lme4 package)

```
fixef(post)
```

```
## (Intercept)     arousal
##  30.0711171   0.5356677
```

```
cbind(b = head(ranef(post)$PID), total = head(coef(post)$PID))
```

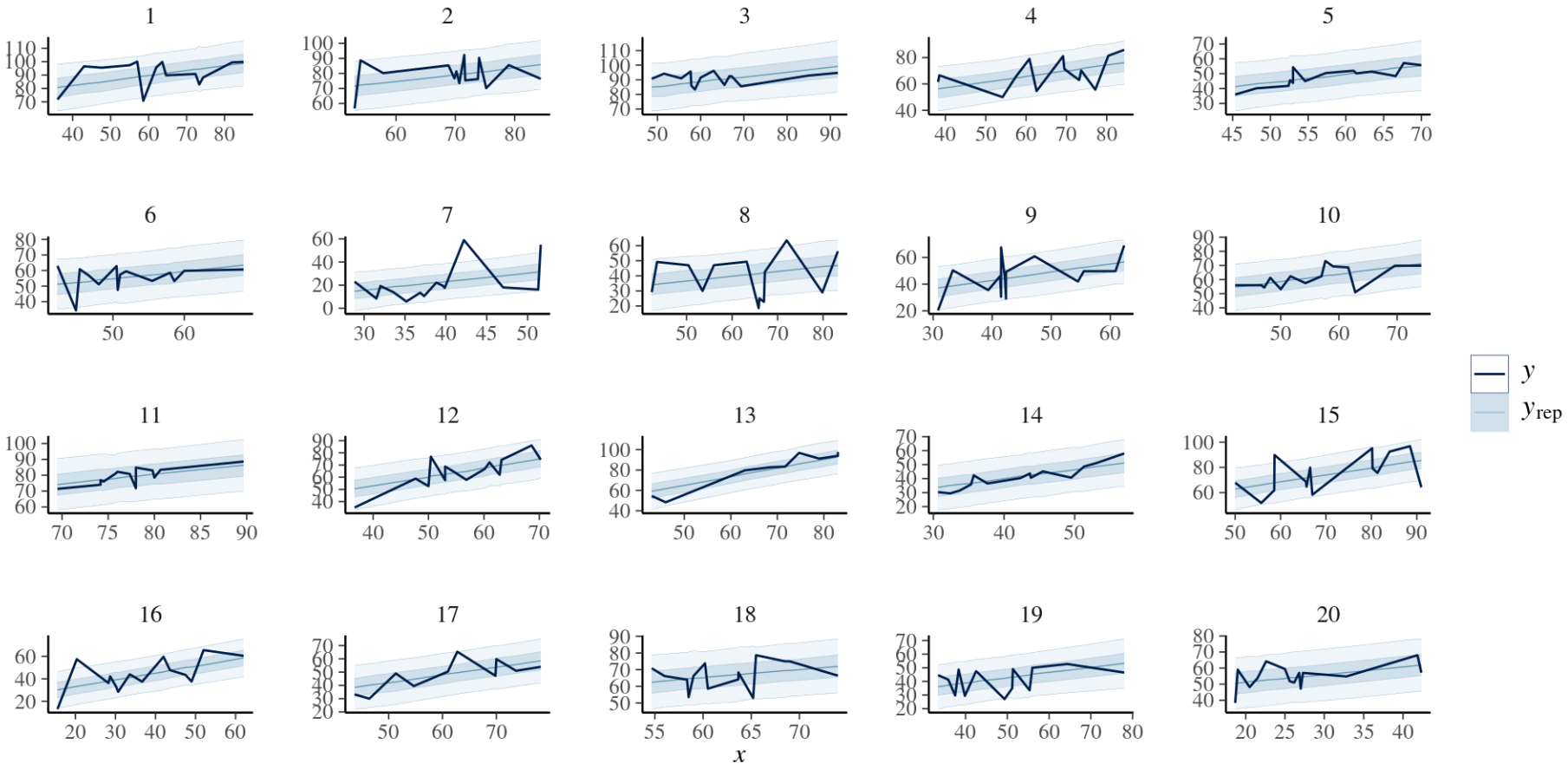
```
##   b.(Intercept)   b.arousal total.(Intercept) total.arousal
## 1     36.801357 -0.16132003          66.87247     0.3743476
## 2     16.905104 -0.07689883          46.97622     0.4587688
## 3     37.743258 -0.18874507          67.81438     0.3469226
## 4      9.436015 -0.09878442          39.50713     0.4368832
## 5    -14.412502  0.03468835          15.65861     0.5703560
## 6      1.232274 -0.06514755          31.30339     0.4705201
```

```
dim(as.matrix(post)) # 4000 x 46
```

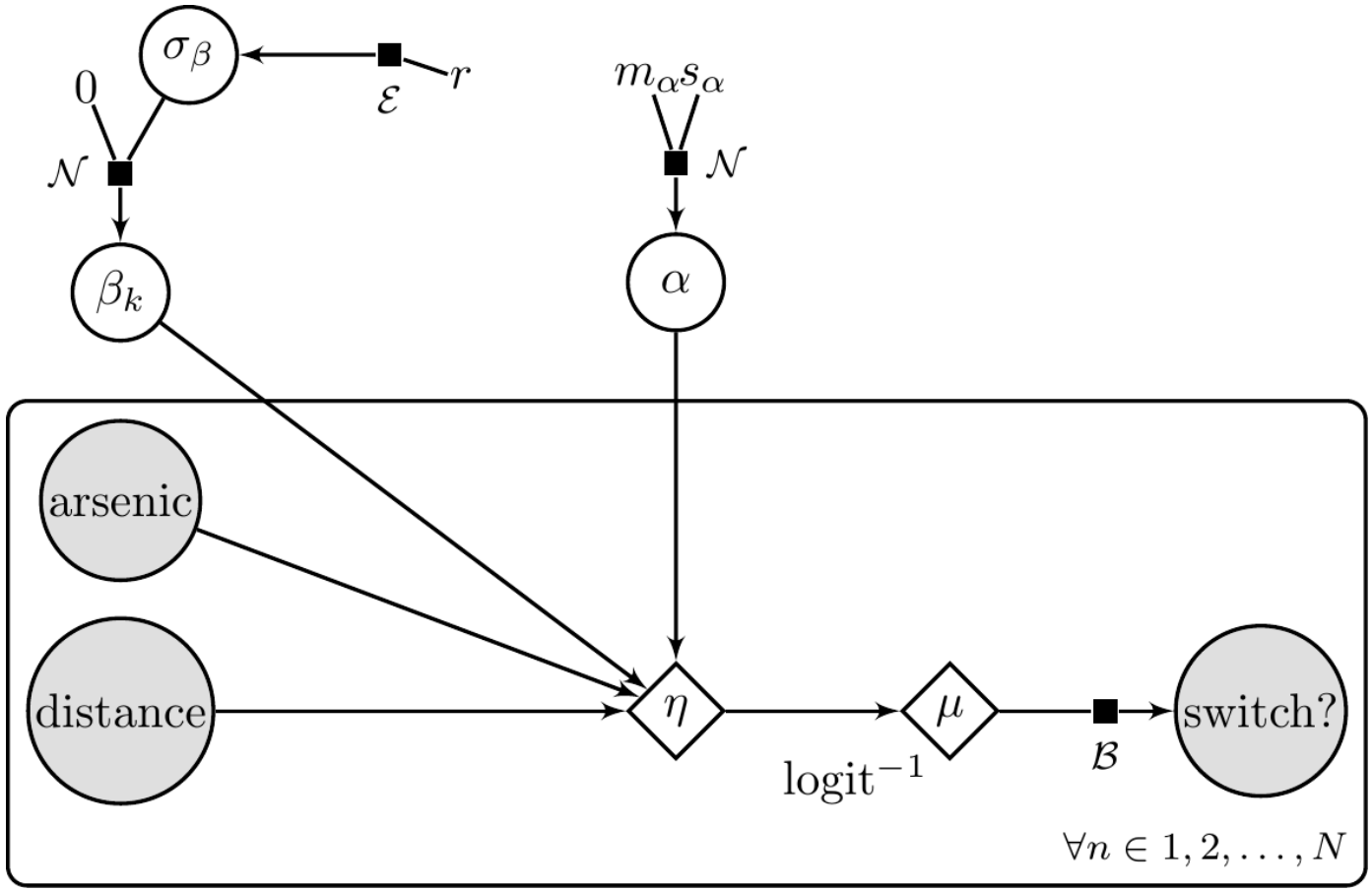
```
## [1] 4000   46
```

Posterior Predictive Checks

```
pp_check(post, plotfun = "ribbon_grouped", x = dat$arousal, group = dat$PID)
```



Prior Predictive Distribution for Well Switching



Well Switching Model

Prior Predictive Distribution in Symbols

$$\sigma_\beta :\sim \mathcal{E}(r)$$
$$\forall k : \beta_k \sim \mathcal{N}(0, \sigma_\beta)$$
$$\alpha \sim \mathcal{N}(m_\alpha, s_\alpha)$$
$$\forall n : \eta_n \equiv \alpha + s(ARSENIC_n, DISTANCE_n, \beta_1 \ldots \beta_K)$$
$$\forall n : \epsilon_n \sim \mathcal{L}(0, 1)$$
$$\forall n : u_n \equiv \eta_n + \epsilon_n$$
$$\forall n : Y_n \equiv u_n > 0$$

- $s(\cdot)$ is a smooth but non-linear function of arsenic and well-distance that has many coefficients, each of which has a normal prior with expectation zero and standard deviation σ_β, which has an exponential prior with expectation r^{-1}
- $\Pr(y_n = 1 \mid \ldots) = \Pr(\eta_n + \epsilon_n > 0) = \Pr(\epsilon_n > -\eta_n) = \Pr(\epsilon_n \leq \eta_n)$, which can evaluated using the standard logistic CDF, $F(\eta_n) = \frac{1}{1+e^{-\eta_n}}$

Posterior Distribution

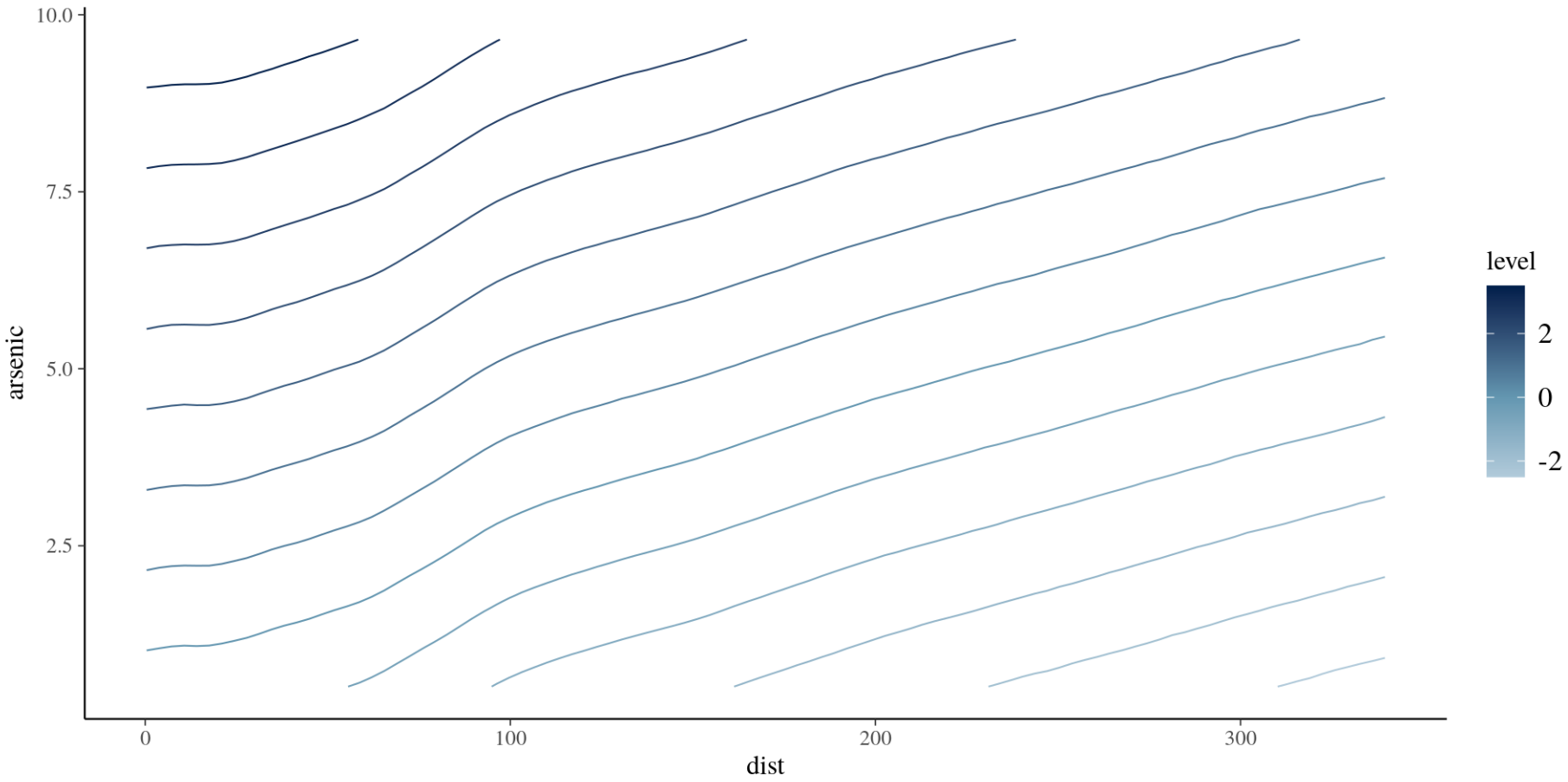
```
post <- stan_gamm4(switch ~ s(dist, arsenic), data = wells, family = binomial, adapt_delta = 0.98)
```

```
print(post, digits = 2)
```

```
...
##                     Median MAD_SD
## (Intercept)          0.33   0.04
## s(dist,arsenic).1   -0.03   0.53
## s(dist,arsenic).2    0.00   0.54
## s(dist,arsenic).3    0.02   0.55
## s(dist,arsenic).4    0.00   0.56
## s(dist,arsenic).5   -0.06   0.56
## s(dist,arsenic).6   -0.01   0.56
## s(dist,arsenic).7    0.00   0.53
## s(dist,arsenic).8   -0.04   0.56
## s(dist,arsenic).9   -0.08   0.55
## s(dist,arsenic).10 -0.04   0.54
## s(dist,arsenic).11  0.04   0.54
## s(dist,arsenic).12  0.09   0.54
## s(dist,arsenic).13 -0.31   0.63
## s(dist,arsenic).14 -0.24   0.58
## s(dist,arsenic).15  0.03   0.52
## s(dist,arsenic).16  0.02   0.55
## s(dist,arsenic).17 -0.03   0.52
## s(dist,arsenic).18 -0.12   0.53
## s(dist,arsenic).19  0.07   0.51
## s(dist,arsenic).20  0.03   0.46
## s(dist,arsenic).21 -0.05   0.48
## s(dist,arsenic).22 -0.03   0.53
## s(dist,arsenic).23 -0.64   0.51
## s(dist,arsenic).24 -0.20   0.40
## s(dist,arsenic).25 -0.18   0.55
## s(dist,arsenic).26  0.09   0.54
## s(dist,arsenic).27 -0.01   0.43
## s(dist,arsenic).28  7.94   1.06
## s(dist,arsenic).29  6.87   1.98
##
## Smoothing terms:
##                                Median MAD_SD
## smooth_sd[s(dist,arsenic)1] 0.64   0.45
## smooth_sd[s(dist,arsenic)2] 4.61   1.18
##
## ------
## * For help interpreting the printed output see ?print.sta
## * For info on the priors used see ?prior_summary.stanreg
...
```

Nonlinear Plot

```
plot_nonlinear(post) # coloring is in log-odds units
```



Covariance and Correlation Matrices

- Recall that if $g(X_i, X_j) = (X_i - \mu_i)(X_j - \mu_j)$, then

$$\mathbb{E}g(X_i, X_j) = \int_{\Omega_{X_j}} \int_{\Omega_{X_i}} (x_i - \mu_i)(x_j - \mu_j) f(x_i, x_j) dx_i dx_j = \sigma_{ij}$$

is the covariance between X_i and X_j, while $\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \in [-1, 1]$ is their correlation, which is a measure of LINEAR dependence

- Let $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$ be $K \times K$, such that $\Sigma_{ij} = \sigma_{ij} \, \forall i, j$ and $\Lambda_{ij} = \rho_{ij} \, \forall i \neq j$
  - Since $\sigma_{ij} = \sigma_{ji} \, \forall i, j$, $\mathbf{\Sigma} = \mathbf{\Sigma}^\top$ is symmetric
  - Since $\sigma_{ij} = \sigma_i^2$ iff $i = j$, $\Sigma_{ii} = \sigma_i^2 > 0$
  - Hence, $\mathbf{\Sigma} = \mathbb{E}(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top$ is the variance-covariance matrix of $\mathbf{x}$
  - $\mathbf{\Sigma} = \mathbf{\Delta \Lambda \Delta}$ where $\mathbf{\Delta}$ is a diagonal matrix of standard deviations

Multivariate CDFs, PDFs, and Expectations

- If $\mathbf{x}$ is a K-vector of continuous random variables

$$F(\mathbf{x}) = \Pr\left(X_1 \leq x_1 \bigcap X_2 \leq x_2 \bigcap \cdots \bigcap X_K \leq x_K\right)$$

$$f(\mathbf{x}) = \frac{\partial^K F(\mathbf{x})}{\partial x_1 \partial x_2 \cdots \partial x_K} = f_1(x_1) \prod_{k=2}^{K} f_k(x_k | x_1, \ldots, x_{k-1})$$

$$F(\mathbf{x}) = \int_{-\infty}^{x_k} \cdots \int_{-\infty}^{x_2} \int_{-\infty}^{x_1} f(\mathbf{x})\, dx_1 dx_2 \cdots dx_K$$

$$\mathbb{E}g(\mathbf{x}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(\mathbf{x})\, f(\mathbf{x})\, dx_1 dx_2 \cdots dx_K$$

$$\boldsymbol{\mu}^\top = \mathbb{E}\mathbf{x}^\top = \begin{bmatrix} \mathbb{E}X_1 & \mathbb{E}X_2 & \cdots & \mathbb{E}X_K \end{bmatrix}$$

$$\boldsymbol{\Sigma}^\top = \boldsymbol{\Sigma} = \mathbb{E}\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top\right] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1K} \\ \sigma_{12} & \sigma_2^2 & \cdots & \vdots \\ \vdots & \cdots & \ddots & \sigma_{(K-1)K} \\ \sigma_{1K} & \cdots & \sigma_{(K-1)K} & \sigma_K^2 \end{bmatrix}$$

Cholesky Factors and Positive Definiteness

- Let $\mathbf{L}$ be lower triangular w/ positive diagonal entries such that $\mathbf{L}\mathbf{L}^\top = \mathbf{\Sigma}$, which is a Cholesky factor of $\mathbf{\Sigma}$ and can uniquely be defined via recursion:

$$L_{ij} = \begin{cases} \sqrt{\Sigma_{jj} - \sum_{k=1}^{j-1} L_{kj}^2} & \text{if } i = j \\ \frac{1}{L_{jj}} \left(\Sigma_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} \right) & \text{if } i > j \\ 0 & \text{if } i < j \end{cases}$$

- Positive definiteness of $\mathbf{\Sigma}$ implies L_{jj} is real and positive for all j and implies the existence of $\mathbf{\Sigma}^{-1} = \mathbf{L}^{-1} \left(\mathbf{L}^{-1} \right)^\top$, which is called a "precision matrix". But not all symmetric matrices are positive definite, so $\Theta \subset \mathbb{R}^{K + \binom{K}{2}}$ in this case
- The `cholesky_decompose` function in Stan outputs $\mathbf{L}$, while the `chol` function in R outputs $\mathbf{L}^\top$ instead

Determinants

- A determinant is “like” a multivariate version of the absolute value operation and is denoted with the same symbol, $|\mathbf{X}|$
- Iff $|\mathbf{X}| \neq 0$, then $\mathbf{X}^{-1}$ exists and $\left|\mathbf{X}^{-1}\right| = \frac{1}{|\mathbf{X}|}$
- All you need to know about how determinants are calculated:
  - Determinant of a product of square matrices is equal to the product of their determinants
  - Determinant of a triangular matrix is the product of its diagonal elements
  - Thus, the determinant of a covariance matrix is the squared product of the diagonal elements of its Cholesky factor

Frequentist Example

```
poll <- readRDS("GooglePoll.rds") # WantToWin is coded as 1 for Romney and 0 for Obama
poll$Income[poll$Income == "150,000+"] <- "100,000-149,999"
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
```

```
mle <- lme4::glmer(cbind(Romney, Obama) ~ Gender + Urban_Density + Age + Income +
                   (Gender + Urban_Density + Age + Income | Region),
                   data = collapsed, family = binomial(link = "logit"))
```

```
## boundary (singular) fit: see ?isSingular
```

- For models that are more complicated than `(1 + x | g)`, the MLE of $\boldsymbol{\Sigma}$ usually implies that $\widehat{\boldsymbol{\Sigma}}^{-1}$ does not exist

The LKJ Distribution for Correlation Matrices

- Let $\mathbf{\Delta}$ be a $K \times K$ diagonal matrix such that Δ_{kk} is the k-th standard deviation, σ_k, and let $\mathbf{\Lambda}$ be a correlation matrix
- Formulating a prior for $\mathbf{\Sigma} = \mathbf{\Delta}\mathbf{\Lambda}\mathbf{\Delta}$ is harder than putting a prior on $\mathbf{\Delta}$ & $\mathbf{\Lambda}$
- LKJ PDF is $f\left(\mathbf{\Lambda} \middle| \eta\right) = \frac{1}{c(K,\eta)} \left|\mathbf{\Lambda}\right|^{\eta-1} = \left|\mathbf{L}\right|^{2(\eta-1)}$ where $\mathbf{\Lambda} = \mathbf{L}\mathbf{L}^\top$ with $\mathbf{L}$ a Cholesky factor and $c\left(K, \eta\right)$ is the normalizing constant that forces the PDF to integrate to 1 over the space of correlation matrices
  - Iff $\eta = 1$, $f\left(\mathbf{\Lambda} \middle| \eta\right) = \frac{1}{c(K,\eta)}$ is constant
  - If $\eta > 1$, the mode of $f\left(\mathbf{\Lambda} \middle| \eta\right)$ is at $\mathbf{I}$ and as $\eta \uparrow \infty$, $\mathbf{\Lambda} \to \mathbf{I}$
  - If $0 < \eta < 1$, trough of $f\left(\mathbf{\Lambda} \middle| \eta\right)$ is at $\mathbf{I}$, which is an odd thing to believe
- Can also derive the distribution of the Cholesky factor $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}^\top$ is a correlation matrix with an LKJ$\left(\eta\right)$ distribution

Stuff for the Data Block

```
library(lme4)
X <- model.matrix(mle)[ , -1]
Z <- getME(mle, name = "Z")
class(Z)
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

```
parts <- rstan::extract_sparse_parts(Z)
str(parts)
```

```
## List of 3
##  $ w: num [1:2005] 1 1 1 1 1 1 1 1 1 1 ...
##  $ v: int [1:2005] 1 10 1 11 1 5 10 1 5 11 ...
##  $ u: int [1:514] 1 3 5 8 11 14 17 20 23 25 ...
```

Bayesian Version of the "Same" Model

```
post_h <- stan_glmer(cbind(Romney, Obama) ~ Gender + Urban_Density + Age + Income +
                       (Gender + Urban_Density + Age + Income | Region),
                     data = collapsed, family = binomial(link = "logit"),
                     QR = TRUE, adapt_delta = 0.98, seed = 12345)
```

```
post_h
```

```
...
##                        Median MAD_SD
## (Intercept)            -0.5    0.2
## GenderMale              0.4    0.1
## Urban_DensitySuburban -0.2    0.1
## Urban_DensityUrban    -0.5    0.1
## Age25-34               0.1    0.1
## Age35-44               0.5    0.1
## Age45-54               0.8    0.1
## Age55-64               0.9    0.1
## Age65+                 1.3    0.1
## Income25,000-49,999   -0.1    0.1
## Income50,000-74,999   -0.1    0.1
## Income75,000-99,999   -0.1    0.2
## Income100,000-149,999  0.2    0.3
##
## Error terms:
##  Groups Name                   Std.Dev. Corr
##  Region (Intercept)            0.171
##         GenderMale             0.092     0.00
##         Urban_DensitySuburban 0.099    -0.02 -0.01
##         Urban_DensityUrban    0.098     0.03  0.00  0.06
##         Age25-34              0.108     0.06 -0.01  0.00  0.01
##         Age35-44              0.124     0.00  0.02  0.02  0.02  0.01
##         Age45-54              0.116     0.08 -0.02 -0.01  0.02  0.05 -0.04
##         Age55-64              0.104     0.01 -0.01 -0.01  0.03  0.01 -0.01  0.00
##         Age65+                0.111     0.03 -0.01 -0.01 -0.01  0.00 -0.01 -0.01
##         Income25,000-49,999   0.115    -0.08 -0.02 -0.04 -0.01 -0.01  0.00  0.00
##         Income50,000-74,999   0.113    -0.04  0.00  0.01 -0.01 -0.01  0.00  0.00
##         Income75,000-99,999   0.130    -0.04  0.03 -0.01 -0.01 -0.01  0.00  0.00
##         Income100,000-149,999 0.130     0.02 -0.01  0.01 -0.01 -0.01  0.00  0.00
##
##
##
##
##
##
##
##
##
##   0.03
##  -0.01  0.01
##  -0.02  0.00  0.00
##   0.01 -0.01  0.01 -0.02
## Num. levels: Region 4
##
## ------
## * For help interpreting the printed output see ?print.sta
## * For info on the priors used see ?prior_summary.stanreg
```

Matrix Factorization

- How many ways can 24 be factored over the positive integers?

  1. 1×24
  2. 2×12
  3. 3×8
  4. 4×6
  5. $2^3 \times 3$

- Matrices can be factored into the product of two (or more) special matrices, and the restrictions on the special matrices can make the factorization unique, such as $\boldsymbol{\Sigma} = \boldsymbol{\Delta}\boldsymbol{\Lambda}\boldsymbol{\Delta}$

- Another example is the QR factorization $\underbrace{\mathbf{X}}_{N \times K} = \underbrace{\mathbf{Q}}_{N \times K}\underbrace{\mathbf{R}}_{K \times K}$, where $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}$ and $\mathbf{R}$ is upper triangular with non-negative diagonal elements

What Does `QR = TRUE` Do?

- Let the vector of linear predictions in a GLM be $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$
- If we apply the QR decomposition to $\mathbf{X}$,

$$\boldsymbol{\eta} = \overbrace{\mathbf{QR}}^{\mathbf{X}}\boldsymbol{\beta} = \overbrace{\mathbf{Q}\frac{R_{KK}}{R_{KK}}\mathbf{R}}^{\mathbf{X}}\boldsymbol{\beta} = \overbrace{\mathbf{Q}^*\mathbf{R}^*}^{\mathbf{X}}\boldsymbol{\beta} = \mathbf{Q}^*\overbrace{\boldsymbol{\theta}}^{\mathbf{R}^*\boldsymbol{\beta}}$$

- When you specify `QR = TRUE`, rstanarm internally does a GLM using $\mathbf{Q}^* = \mathbf{Q}R_{KK}$ as the matrix of predictors instead of $\mathbf{X}$ to get the posterior distribution of $\boldsymbol{\theta}$ and then pre-multiplies each posterior draw of $\boldsymbol{\theta}$ by $\frac{1}{R_{KK}}\mathbf{R}^{-1}$ to get a posterior draw of $\boldsymbol{\beta}$
- Doing so makes it easier for NUTS to sample from the posterior distribution (of $\boldsymbol{\theta}$) efficiently because the columns of $\mathbf{Q}$ are orthogonal, whereas the columns of $\mathbf{X}$ are not

Distributions of Different Random Variables

- α and each β_k have a posterior (or prior) distribution in a regression model
- Let $\eta_n = \alpha + \sum_{k=1}^{K} \beta_k x_{nk}$. The `posterior_linpred` function produces draws of each η_n induced by the posterior distribution of α and each β_k
- In a GLM, $\mu_n = g\left(\eta_n\right)$. The `posterior_epred` function produces draws of each μ_n induced by the posterior distribution of η_n
- The P{D,M}F of the outcome is $f\left(y_n \mid \mu_n, \ldots\right)$. The `posterior_predict` function produces draws of each y_n induced by the posterior distribution of μ_n whose P{D,M}F is $f\left(y_n \mid \mu_n, \ldots\right)$
- But y_n is not conditionally deterministic given μ_n because it includes noise, whose posterior distribution may be governed by other parameters like σ
- In the case of a logit model, $\eta_n \in \mathbb{R}$, $\mu_n = \frac{1}{1+e^{-\eta_n}} \in (0, 1)$, and $y_n \in \{0, 1\}$

Poststratification

```
mu <- posterior_epred(post_h); dim(mu)
```

```
## [1] 4000  513
```

```
table(poll$Gender)
```

```
##
## Female   Male
##   5272   2733
```

- Assume `shares` is the proportion of voters in the population for each level of `Gender`, `Urban_Density`, `Age`, and `Income` crossed with `Region`

```
mu_ <- mu %*% shares
```

- Now you have a posterior distribution for the proportion supporting Romney for the country as a whole

What Were the Priors?

```
prior_summary(post_h)
```

```
## Priors for model 'post_h'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 2.5)
##
## Coefficients (in Q-space)
##  ~ normal(location = [0,0,0,...], scale = [2.5,2.5,2.5,...])
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

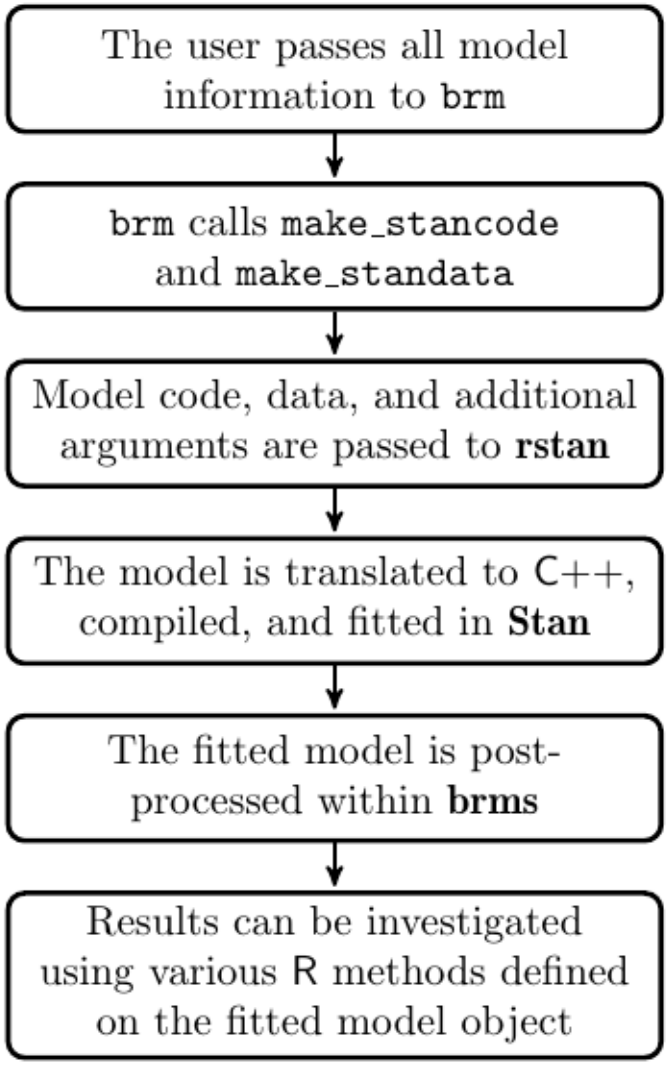
What Is `decov(1, 1, 1, 1)`?

- `decov` = Decomposition of Covariance
- `reg.` is the regularization parameter in the LKJ prior on the correlation matrix
- `conc.` is the concentration parameter in the Dirichlet prior on the variance components
- `shape` and `scale` pertain to the Gamma prior on multiplier for the variance components
- You usually do not need to change these defaults to get good results

Dirichlet Distribution

- Dirichlet distribution is over the parameter space of PMFs — i.e. $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$ — and the Dirichlet PDF is $f\left(\boldsymbol{\pi} \mid \boldsymbol{\alpha}\right) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}$ where $\alpha_k \geq 0 \, \forall k$ and the multivariate Beta function is $B\left(\boldsymbol{\alpha}\right) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma\left(\prod_{k=1}^{K} \alpha_k\right)}$ where $\Gamma\left(z\right) = \frac{1}{z} \prod_{n=1}^{\infty} \frac{\left(1 + \frac{1}{n}\right)^n}{1 + \frac{z}{n}} = \int_0^{\infty} u^{z-1} e^{-u} du$ is the Gamma function
- $\mathbb{E}\pi_i = \frac{\alpha_i}{\sum_{k=1}^{K} \alpha_k} \, \forall i$ and the mode of π_i is $\frac{\alpha_i - 1}{-1 + \sum_{k=1}^{K} \alpha_k}$ if $\alpha_i > 1$
- Iff $\alpha_k = 1 \, \forall k$, $f\left(\boldsymbol{\pi} \middle| \boldsymbol{\alpha} = \mathbf{1}\right)$ is constant over Θ (simplexes)
- Beta distribution is a special case of the Dirichlet where $K = 2$
- Marginal and conditional distributions for subsets of $\boldsymbol{\pi}$ are also Dirichlet

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 = "no", sparse = NULL,
##     knots = NULL, stanvars = NULL, stan_funs = NULL, fit = NA,
##     save_pars = NULL, save_ranef = NULL, save_mevars = NULL,
##     save_all_pars = NULL, inits = "random", chains = 4, iter = 2000,
##     warmup = floor(iter/2), thin = 1, cores = getOption("mc.cores",
##         1), threads = NULL, normalize = getOption("brms.normalize",
##         TRUE), control = NULL, algorithm = getOption("brms.algorithm",
##         "sampling"), backend = getOption("brms.backend", "rstan"),
##     future = getOption("future", FALSE), silent = 1, seed = NA,
##     save_model = NULL, stan_model_args = list(), file = NULL,
##     file_refit = "never", empty = FALSE, rename = TRUE, ...)
## 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

The `prior` Argument to `brm`

```
args(set_prior) # or usually 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)"` but you can omit the quotation marks if you instead call `prior`, which forwards to `set_prior`
- `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)

```
data(roaches, package = "rstanarm"); roaches <- roaches[roaches$roach1 > 0, ]
get_prior(y ~ log(roach1) + treatment + senior + offset(log(exposure2)),
          data = roaches, family = zero_inflated_negbinomial)
```

```
##                   prior     class      coef group resp dpar nlpar bound       source
##                  (flat)         b                                             default
##                  (flat)         b logroach1                               (vectorized)
##                  (flat)         b    senior                               (vectorized)
##                  (flat)         b treatment                               (vectorized)
##  student_t(3, 1.9, 2.9) Intercept                                             default
##       gamma(0.01, 0.01)     shape                                             default
##              beta(1, 1)        zi                                             default
```

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(normal(0, 2), class = "b") + prior(normal(0, 5), class = "Intercept") +
  prior(exponential(1), class = "shape")
```

Example of `brm`

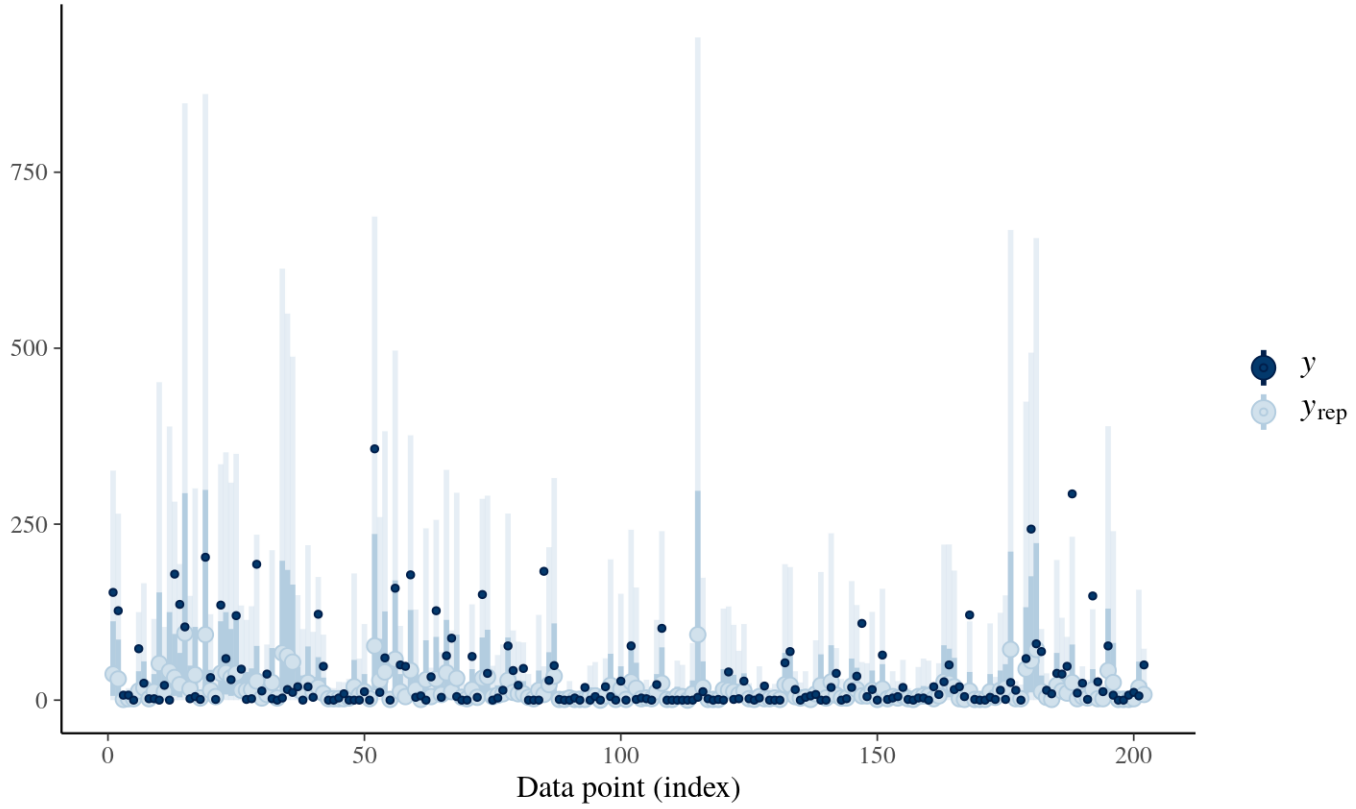
```
post <- brm(y ~ log(roach1) + treatment + senior + offset(log(exposure2)), data = roaches,
            family = zero_inflated_negbinomial, prior = my_prior)
```

```
post
```

```
...
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.41      0.26     0.91     1.93 1.00     2721     2495
## logroach1     0.69      0.06     0.56     0.81 1.00     2962     2616
## treatment    -0.60      0.22    -1.03    -0.18 1.00     2994     2565
## senior       -0.18      0.24    -0.64     0.32 1.00     3906     2661
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## shape     0.55      0.08     0.41     0.74 1.00     2404     2299
## zi        0.06      0.04     0.00     0.15 1.00     2227     1430
##
## 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 **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.brmsfit)
```

```
## 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     -0.6      0.22    -0.97    -0.26        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_predict"
##  [2,] "add_ic"              "loo_linpred"              "posterior_samples"
##  [3,] "as.array"            "loo_model_weights"        "posterior_smooths"
##  [4,] "as.data.frame"       "loo_moment_match"         "posterior_summary"
##  [5,] "as.matrix"           "loo_predict"              "pp_average"
##  [6,] "as.mcmc"             "loo_predictive_interval" "pp_check"
##  [7,] "autocor"             "loo_R2"                   "pp_mixture"
##  [8,] "bayes_factor"        "loo_subsample"            "predict"
##  [9,] "bayes_R2"            "loo"                      "predictive_error"
## [10,] "bridge_sampler"      "LOO"                      "predictive_interval"
## [11,] "coef"                "marginal_effects"         "prepare_predictions"
## [12,] "conditional_effects" "marginal_smooths"         "print"
## [13,] "conditional_smooths" "mcmc_plot"                "prior_samples"
## [14,] "control_params"      "model_weights"            "prior_summary"
## [15,] "cv_varsel"           "model.frame"              "ranef"
## [16,] "expose_functions"    "neff_ratio"               "reloo"
## [17,] "family"              "ngrps"                    "residuals"
## [18,] "fitted"              "nobs"                     "rhat"
## [19,] "fixef"               "nsamples"                 "stancode"
## [20,] "formula"             "nuts_params"              "standata"
## [21,] "get_refmodel"        "pairs"                    "stanplot"
## [22,] "getCall"             "parnames"                 "summary"
## [23,] "hypothesis"          "plot"                     "update"
## [24,] "kfold"               "post_prob"                "VarCorr"
## [25,] "launch_shinystan"    "posterior_average"        "varsel"
## [26,] "log_lik"             "posterior_epred"          "vcov"
## [27,] "log_posterior"       "posterior_interval"       "waic"
## [28,] "logLik"              "posterior_linpred"        "WAIC"
```

Gaussian Processes

A simple Gaussian Process logit model with a squared exponential covariance function is

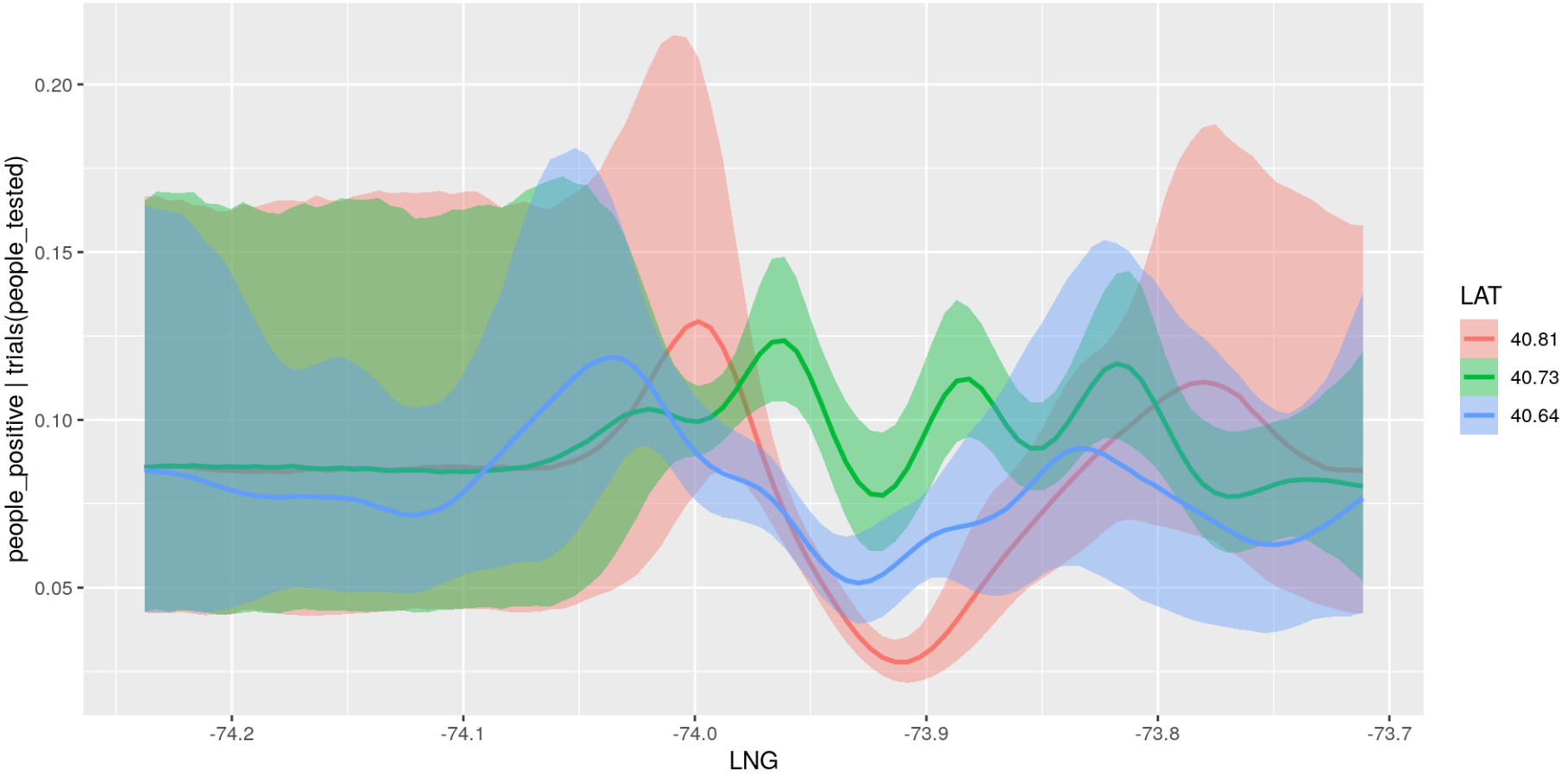
- $\frac{1}{\rho} \sim \mathcal{G}amma\left(a, b\right)$
- $\alpha \sim t_{+}\left(v, 0, s\right)$
- $\Sigma_{ij} = \Sigma_{ji} = \alpha^2 e^{-\frac{1}{\rho}\sum_{d=1}^{D}\left(x_{id} - x_{jd}\right)^2}$
- $\gamma \sim \mathcal{N}\left(0, 2\right)$
- $\boldsymbol{\eta} \sim \mathcal{N}\left(\gamma \mathbf{1}, \boldsymbol{\Sigma}\right)$
- $\mu_j = \frac{1}{1+e^{-\eta_j}}$
- $y_j \sim \text{Binomial}(n_j, \mu_j)$

where, for example, n_j is the number of coronavirus tests in zipcode j and y_j is the number of positives

```
tests <- readr::read_csv("https://raw.githubusercontent.com/nychealth/coronavirus-data/master/latest/last7days-by-m
zipcodes <- readr::read_csv("https://gist.githubusercontent.com/erichurst/7882666/raw/5bdc46db47d9515269ab12ed6fb28
zipcodes$ZIP <- as.integer(zipcodes$ZIP)
tests <- inner_join(tests, zipcodes, by = c("modzcta" = "ZIP"))
post <- brm(people_positive | trials(people_tested) ~ 1 + gp(LAT, LNG, k = 15, c = 5 / 4),
            data = tests, family = binomial, save_pars = save_pars(all = TRUE))
```

What Did the Gaussian Process Model Imply?

```
conditional_effects(post, effects = "LNG:LAT")
```



How Good Was the Model?

```
bayes_R2(post)
```

```
##     Estimate Est.Error      Q2.5     Q97.5
## R2 0.8066437 0.1548843 0.5420572 0.9066818
```

```
loo(post, moment_match = TRUE)
```

```
##
## Computed from 4000 by 177 log-likelihood matrix
##
##           Estimate      SE
## elpd_loo -493663.9 22103.7
## p_loo     492947.7 22095.9
## looic     987327.7 44207.4
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```