An Introduction to Bayesian Inference in Medicine using Stan

Ben Goodrich

September 8, 2018

Obligatory Disclosure

- Ben is an employee of Columbia University, which has received several research grants to develop Stan
- Ben is also a manager of GG Statistics LLC, which uses Stan for business purposes
- According to Columbia University policy, any such employee who has any equity stake in, a title (such as officer or director) with, or is expected to earn at least $\$5,000.00$ per year from a private company is required to disclose these facts in presentations

What Is Stan?

- A high-level probabilistic programming language (PPL) that resembles C
- Like BUGS but unlike most PPLs, Stan focuses on drawing from posterior distributions, i.e. conditional distributions of parameters given known data
- But the MCMC algorithm used by Stan is very different from the Gibbs sampling with fallback algorithm used by BUGS
- Unlike BUGS but like Metropolis-Hastings (MH), Stan requires you to specify the log-density of the posterior distributions, optionally ignoring constants
- Unlike MH, Stan utilizes the gradient (which is calculated automatically) of the log-density to generate proposed moves through the parameter space

Beta-Binomial Example with Ebola

What is the probability that a drug developed by Mapp Biopharmaceutical will allow a person with Ebola to survive?

- The Beta distribution for $\pi \in [0,1]$ has 2 positive shape parameters α and β
- Its mode is $M = \frac{\alpha-1}{\alpha+\beta-2}$ but only exists if $\alpha, \beta > 1$
- Its median, $m \approx \frac{\alpha-\frac{1}{3}}{\alpha+\beta-\frac{2}{3}}$, exists but this approximation is good iff $\alpha, \beta > 1$
- Given $M, m \in (0,1)$, you can solve for $\alpha > 1$ and $\beta > 1$
- $\alpha = \frac{m(4M-3)+M}{3(M-m)}$
- $\beta = \frac{m(1-4M)+5M-2}{3(M-m)}$
- But m must be between $\frac{1}{2}$ and M

Stan Program for Ebola

```
data {
  int<lower = 1> exposed;
  int<lower = 0, upper = exposed> survived;
  real<lower = 1> alpha;
  real<lower = 1> beta;
}
transformed data { // this block is only executed once
  int died = exposed - survived;
  real constant_1 = lchoose(exposed, survived); // log of binomial coefficient
  real constant_2 = -lbeta(alpha, beta);        // negative log of beta function
}
parameters { real<lower = 0, upper = 1> pi; }   // survival probability
model {
  real log_pi = log(pi);
  real log_1mpi = log1m(pi);
  target += constant_1 + survived * log_pi + died * log_1mpi; // binomial_lpmf(...)
  target += constant_2 + (alpha - 1) * log_pi + (beta - 1) * log_1mpi; // beta_lpdf(...)
}
generated quantities { real odds = pi / (1 - pi); }
```

Posterior Output from Ebola Model

```
library(rstan)
M <- 2 / 3 # prior mode
m <- 0.55  # prior median
alpha <- (m * (4 * M - 3) + M) / (3 * (M - m))
beta  <- (m * (1 - 4 * M) + 5 * M - 2) / (3 * (M - m))
post <- stan("ebola.stan", refresh = 0, # suppresses intermediate output
             data = list(exposed = 7, survived = 5, alpha = alpha, beta = beta))
```

```
post
```

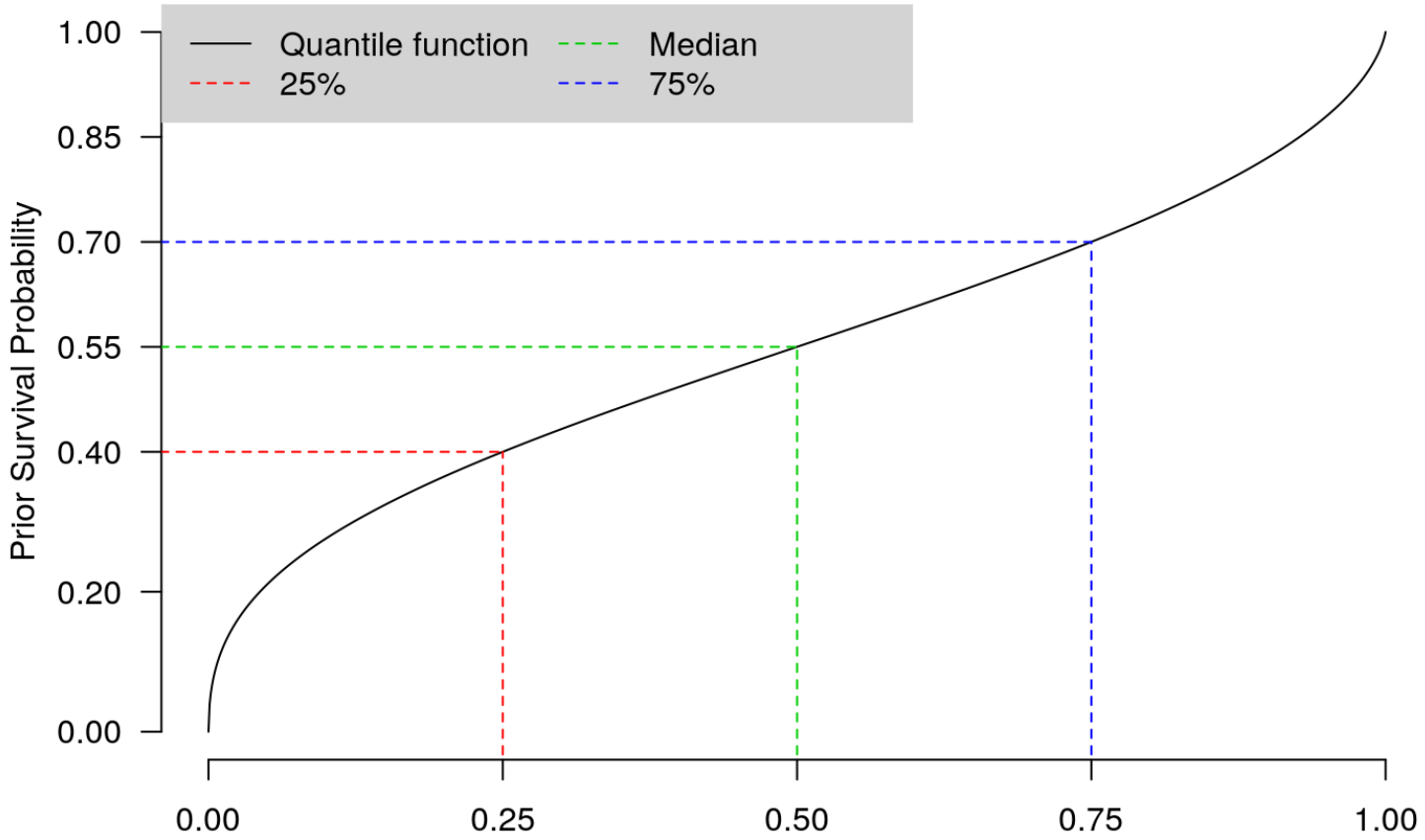
```
## Inference for Stan model: ebola.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## pi    0.67    0.00 0.15  0.37  0.57  0.68  0.78  0.91  1399    1
## odds  3.01    0.08 3.16  0.58  1.32  2.14  3.57 10.56  1463    1
## lp__ -3.07    0.02 0.76 -5.20 -3.27 -2.77 -2.57 -2.51  1428    1
##
## Samples were drawn using NUTS(diag_e) at Sat Sep  8 07:42:30 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

But Specifying Prior Distributions Is Too Hard

- Specifying priors is the biggest obstacle for people trying Bayesian methods
- Unlike BUGS, Stan is indifferent to whether you use conjugate priors
- So, use prior distributions that are easiest for you
- A quantile function is the inverse of a CDF, so it maps from cumulative probability to order statistics (although often no closed-form exists)
- Inputting a standard uniform random variate into the quantile function for distribution $\mathcal{D}$ yields a realization from $\mathcal{D}$
- Quantile Parameterized Distributions (QPDs) are basically distributions whose parameters are quantiles, such as the median, 25%, 75%, 2.5%, 97.5%, etc. See http://metalogdistributions.com/publications.html
- If you can specify bounds, the median, and a couple other quantiles for a parameter, we can *construct* a valid probability distribution that is both differentiable and consistent with those quantiles
- Similar variants exist for one-sided or unbounded prior distributions

Exposing User-Defined Stan Programs to R

```
expose_stan_functions("JQPD.stan") # JQPDB_icdf now exists in R's global environment
stopifnot(all.equal(0.7, JQPDB_icdf(p = 0.75, alpha = 0.25, 0, 0.4, 0.55, 0.7, 1)))
```



Another Stan Program for Ebola

```
// this next line brings in the JQPDB_icdf function, among others that are not used here
#include /JQPD.stan
data {
  int<lower = 1> exposed;
  int<lower = 0, upper = exposed> survived;

  real<lower = 0, upper = 0.5> alpha;   // low is the alpha quantile
  real<lower = 0, upper = 1> low;
  real<lower = low, upper = 1> median;
  real<lower = median, upper = 1> high; // the 1 - alpha quantile
}
parameters { real<lower = 0, upper = 1> p; }  // primitive with implicit uniform prior
transformed parameters {
  real pi = JQPDB_icdf(p, alpha, 0.0, low, median, high, 1.0); // survival probability
}
model {
  target += binomial_lpmf(survived | exposed, pi); // log-likelihood
}
```

Posterior Output from Alternate Ebola Model

```
post2 <- stan("ebola2.stan", refresh = 0, # suppresses intermediate output
              data = list(exposed = 7, survived = 5, alpha = 0.25, low = 0.4,
                          median = 0.55, high = 0.7))
```

```
post2
```

```
## Inference for Stan model: ebola2.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## p     0.63    0.01 0.21  0.19  0.49  0.67  0.80  0.94  1433    1
## pi    0.64    0.00 0.14  0.36  0.54  0.65  0.74  0.87  1431    1
## lp__ -3.27    0.02 0.80 -5.39 -3.46 -2.97 -2.78 -2.72  1445    1
##
## Samples were drawn using NUTS(diag_e) at Sat Sep  8 07:42:49 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

But Organizing the Data for Stan Is Too Hard

- The **rstanarm** package comes with precompiled Stan programs that accept the same syntax as popular R functions, including
  - `lm`, `biglm::biglm`, `glm`, `MASS::glm.nb`, and `aov` for (G)LMs
  - `betareg::betareg` when the outcome is a proportion
  - `survival::clogit` for case-control studies
  - `gamm4::gamm4` (really `mgcv::jagam`) supports splines
  - `lme4::lmer`, `lme4::glmer`, and `lme4::nlmer` for hierchical models
  - `JMBayes::jm` (plus `mvmer`) for "joint" models of survival / severity
  - `MASS::polr` for ordinal outcomes
- Just prefix with `rstanarm::stan_`, change the default priors if you like, and you get draws from the posterior distribution

Nonlinear Hierchical ODE Model for Theophylline

```
library(rstanarm)
post3 <- stan_nlmer(conc ~ SSfol(Dose, Time, lKe, lKa, lCl) ~
                     (0 + lKe + lKa + lCl | Subject), data = Theoph,
                    prior = normal(location = c(-2, 0.5, -3), scale = 1, autoscale = FALSE),
                    seed = 982018, refresh = 0)
```

```
str(as.data.frame(post3))
```

```
## 'data.frame':    4000 obs. of  46 variables:
##  $ lKe                    : num  -2.55 -2.42 -2.56 -2.46 -2.53 ...
##  $ lKa                    : num  0.451 0.32 0.284 0.423 0.336 ...
##  $ lCl                    : num  -3.41 -3.25 -3.18 -3.22 -3.3 ...
##  $ b[lKe Subject:6]       : num  0.3614 0.0751 -0.0215 0.0374 -0.0385 ...
##  $ b[lKa Subject:6]       : num  -0.176041 0.171462 -0.161724 0.000188 -0.318296 ...
##  $ b[lCl Subject:6]       : num  0.584 0.297 0.021 0.228 0.093 ...
##  $ b[lKe Subject:7]       : num  0.1917 -0.126 -0.0865 -0.1058 0.1596 ...
##  $ b[lKa Subject:7]       : num  -0.602 -0.42 -0.548 -0.627 -0.564 ...
##  $ b[lCl Subject:7]       : num  0.4631 0.1137 0.0222 0.1477 0.3247 ...
##  $ b[lKe Subject:8]       : num  -0.03695 -0.16462 -0.00217 -0.20527 -0.10048 ...
##  $ b[lKa Subject:8]       : num  0.1732 0.1341 0.3111 -0.0276 0.0789 ...
##  $ b[lCl Subject:8]       : num  0.219432 0.106263 0.000412 -0.099001 0.096984 ...
##  $ b[lKe Subject:11]      : num  0.0815 0.4936 0.3042 0.3259 0.2145 ...
##  $ b[lKa Subject:11]      : num  0.687 0.722 0.708 0.879 0.583 ...
```

Results of Theophylline Model

We can use `summary()` or implicitly `print()` to see the highlights of the posterior distribution:

```
post3
```

```
## stan_nlmer
##  family:       gaussian [inv_SSfol]
##  formula:      conc ~ SSfol(Dose, Time, lKe, lKa, lCl) ~ (0 + lKe + lKa + lCl |
##     Subject)
##  observations: 132
## ------
##       Median MAD_SD
## lKe   -2.4    0.1
## lKa    0.5    0.2
## lCl   -3.2    0.1
## sigma  0.7    0.0
##
## Error terms:
##  Groups   Name Std.Dev. Corr
##  Subject  lKe  0.20
##           lKa  0.60      0.03
##           lCl  0.36      0.86 -0.05
##  Residual      0.70
## Num. levels: Subject 12
##
## Sample avg. posterior predictive distribution of y:
##          Median MAD_SD
## mean_PPD 5.0    0.1
##
## ------
## For info on the priors used see help('prior_summary.stanreg').
```

MCMC Does *not* Generally Work in Finite Time

- Almost all differentiable Stan programs will work the first time, can be made to work with enough skill, or will yield warnings indicating they failed
- Theophylline model throws a warning about 59 divergent transitions
- Implies part of the posterior distribution is unreachable due to numerical error
- At best, only the posterior medians are trustworthy

The trialr and RBesT R Packages

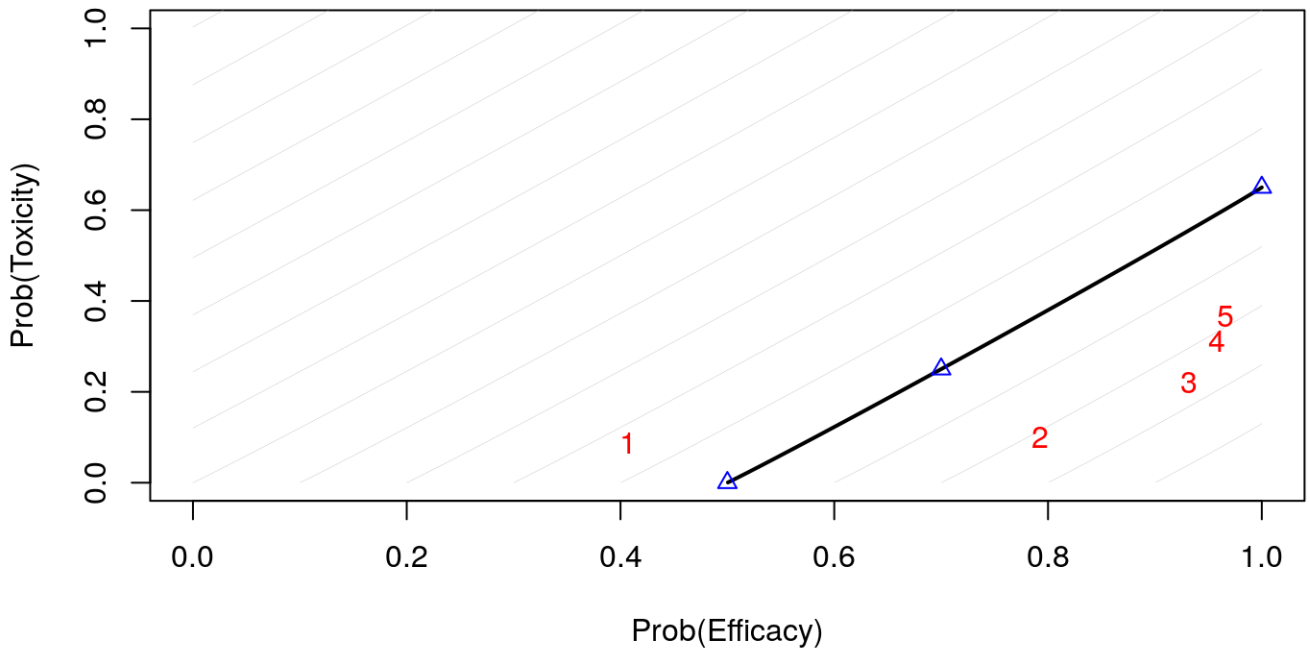
- There are several PK / PD packages using Stan on CRAN now
- Two of the best are **trialr** (Brock) and **RBesT** (Novartis)
- **trialr** goes beyond "3+3" designs to implement "model-based" designs:
  - `EffTox`: seamless phase I-II dose-finding, published by Thall & Cook 2004
  - Hierarchical model for a phase II trial of a treatment in a disease with multiple sub-types using binary responses, published by Thall et al 2003
  - `BEBOP`, a stratified medicine design for studying efficacy and toxicity in phase II that incorporates predictive baseline information, as developed for the PePS2 trial and submitted for publication by Brock, et al. 2017
- `RBesT::gMAP` implements a meta-analytic-predictive (MAP) approach that derives a prior from historical experimental data using a hierarchical model

Efficacy-Toxicity Example from trialr Vignette

```
library(trialr)
outcomes <- '1NNE 2EEB'
mod <- stan_efftox(outcomes,real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0),
                   efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,
                   p_e = 0.1, p_t = 0.1, eff0 = 0.5, tox1 = 0.65,
                   eff_star = 0.7, tox_star = 0.25,
                   alpha_mean = -7.9593, alpha_sd = 3.5487,
                   beta_mean = 1.5482, beta_sd = 3.5018,
                   gamma_mean = 0.7367, gamma_sd = 2.5423,
                   zeta_mean = 3.4181, zeta_sd = 2.4406,
                   eta_mean = 0, eta_sd = 0.2, psi_mean = 0, psi_sd = 1, seed = 123)
```

Decision-Theory in Efficacy-Toxicity Example

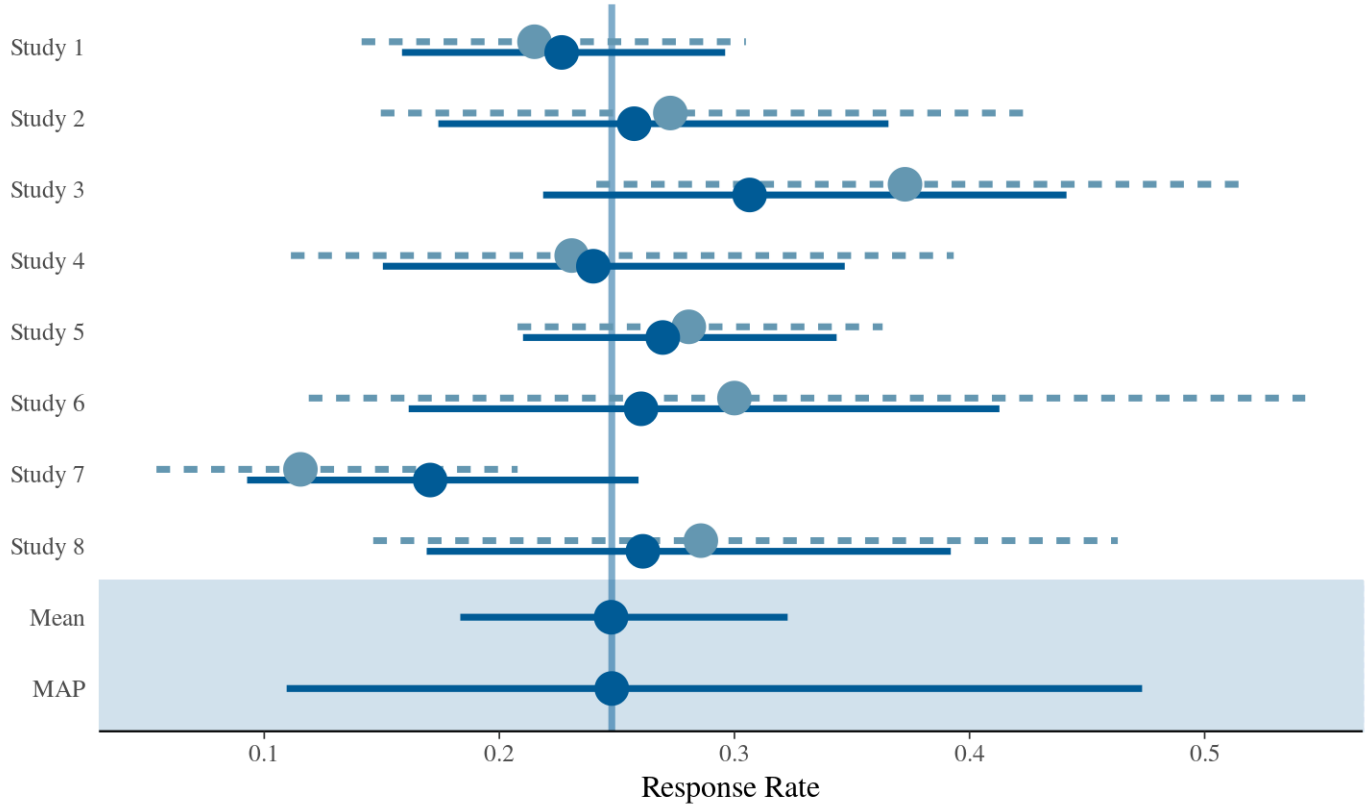
```
efftox_contour_plot(mod$dat, prob_eff = mod$prob_eff, prob_tox = mod$prob_tox)
```



Dose level 3 is slightly preferred because it is farthest to the southeast

Ankylosing Spondylitis from RBesT Vignette

```
library(RBesT)
set.seed(34563)
map_mcmc <- gMAP(cbind(r, n-r) ~ 1 | study, data = AS, family = binomial,
                 tau.dist = "HalfNormal", tau.prior = 1, beta.prior = 2)
```



Analysis of Ankylosing Spondylitis Model

```
library(RBesT)
map <- automixfit(map_mcmc)
map_robust <- robustify(map, weight = 0.2, mean = 0.5)
post_placebo <- postmix(map_robust,  r = 1, n = 6)
post_treat   <- postmix(mixbeta(c(1, 0.5, 1)), r = 14, n = 24)
pmixdiff(post_placebo, post_treat, 0) # prob negative
```

```
## [1] 0.9912264
```

The rstantools R Package

- One thing that makes **trialr** and **RBesT** great is that they were created with `rstantools::rstan_package_skeleton`, which provides the skeleton of an R package for Stan programs (see the **rstantools** vignette)
- The package maintainer only has to
  - Write a useful Stan program
  - Write R wrapper functions that usually input a `formula`, `data.frame`, etc. process the data into the form specified by the Stan program, and call it
  - Write post-estimation methods (can import generics from **rstantools**)
  - Test the functions in the package
- We want to see more R packages using Stan this way

But Writing Stan Programs Is Too Hard

- The `brms::brm` function inputs a `formula`, `data.frame`, etc., writes a Stan program, compiles it, and executes it, which supports a wider range of regression models than does **rstanarm**
- Example adapted from Matti Vuorre

```
library(metafor)
dat <- escalc(measure="ZCOR", ri = ri, ni = ni, data = dat.molloy2014)
dat$sei <- c(.1, .04, .14, .12, .1, .13, .08, .06, .13, .04, .14, .11, .09, .04, .08, .13)
dat$study <- LETTERS[1:nrow(dat)]
library(brms)
brm_out <- brm(yi | se(sei) ~ (1|study), data = dat)
```

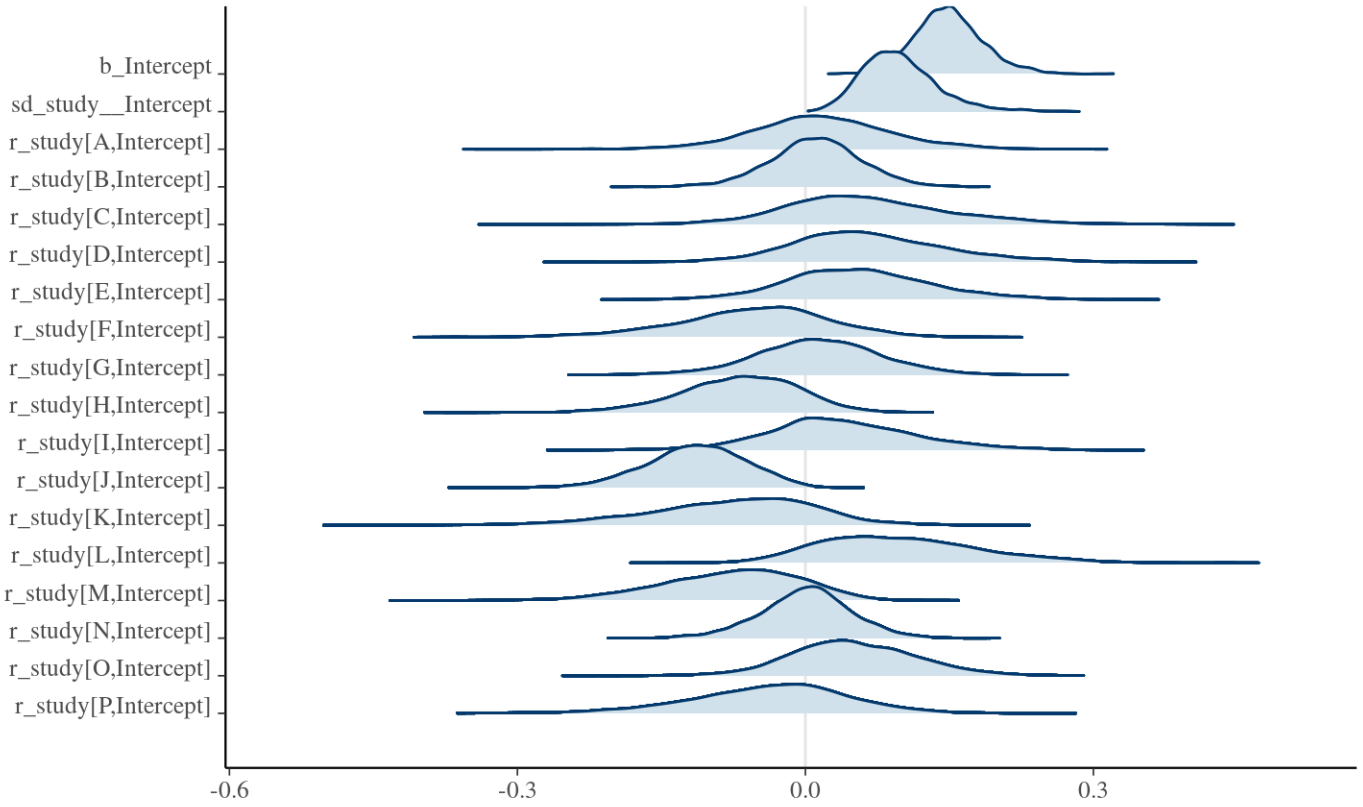
Results from Meta-Analysis

```
brm_out
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: yi | se(sei) ~ (1 | study)
##    Data: dat (Number of observations: 16)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~study (Number of levels: 16)
##               Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)     0.10      0.04     0.03     0.19       1241 1.00
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     0.15      0.04     0.09     0.23       1613 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Plot of Results from Meta-Analysis

```
bayesplot::mcmc_areas_ridges(as.matrix(brm_out), regex_pars = "Intercept")
```



Learn How to Code Stan Models from brms

```
str(brms::make_standata(yi | se(sei) ~ (1|study), data = dat), give.attr = FALSE)
```

```
## List of 12
##  $ N         : int 16
##  $ Y         : num [1:16(1d)] 0.189 0.163 0.354 0.332 0.277 ...
##  $ se        : num [1:16(1d)] 0.1 0.04 0.14 0.12 0.1 0.13 0.08 0.06 0.13 0.04 ...
##  $ K         : int 1
##  $ X         : num [1:16, 1] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Z_1_1     : num [1:16(1d)] 1 1 1 1 1 1 1 1 1 1 ...
##  $ sigma     : num 0
##  $ J_1       : int [1:16(1d)] 1 2 3 4 5 6 7 8 9 10 ...
##  $ N_1       : int 16
##  $ M_1       : int 1
##  $ NC_1      : num 0
##  $ prior_only: int 0
```

The `data` and `transformed data` Blocks

```
brms::make_stancode(yi | se(sei) ~ (1|study), data = dat)
```

```
## // generated with brms 2.4.0
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   vector[N] Y;  // response variable
##   vector<lower=0>[N] se;  // known sampling error
##   real<lower=0> sigma;  // residual SD
##   // data for group-level effects of ID 1
##   int<lower=1> J_1[N];
##   int<lower=1> N_1;
##   int<lower=1> M_1;
##   vector[N] Z_1_1;
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
##   vector<lower=0>[N] se2 = square(se);
## }
## parameters {
##   real temp_Intercept;  // temporary intercept
##   vector<lower=0>[M_1] sd_1;  // group-level standard deviations
##   vector[N_1] z_1[M_1];  // unscaled group-level effects
```

The Remaining Blocks

```
## parameters {
##   real temp_Intercept;  // temporary intercept
##   vector<lower=0>[M_1] sd_1;  // group-level standard deviations
##   vector[N_1] z_1[M_1];  // unscaled group-level effects
## }
## transformed parameters {
##   // group-level effects
##   vector[N_1] r_1_1 = sd_1[1] * (z_1[1]);
## }
## model {
##   vector[N] mu = temp_Intercept + rep_vector(0, N);
##   for (n in 1:N) {
##     mu[n] += r_1_1[J_1[n]] * Z_1_1[n];
##   }
##   // priors including all constants
##   target += student_t_lpdf(temp_Intercept | 3, 0, 10);
##   target += student_t_lpdf(sd_1 | 3, 0, 10)
##     - 1 * student_t_lccdf(0 | 3, 0, 10);
##   target += normal_lpdf(z_1[1] | 0, 1);
##   // likelihood including all constants
##   if (!prior_only) {
##     target += normal_lpdf(Y | mu, se);
##   }
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = temp_Intercept;
## }
```

Part of a Warfarin Model

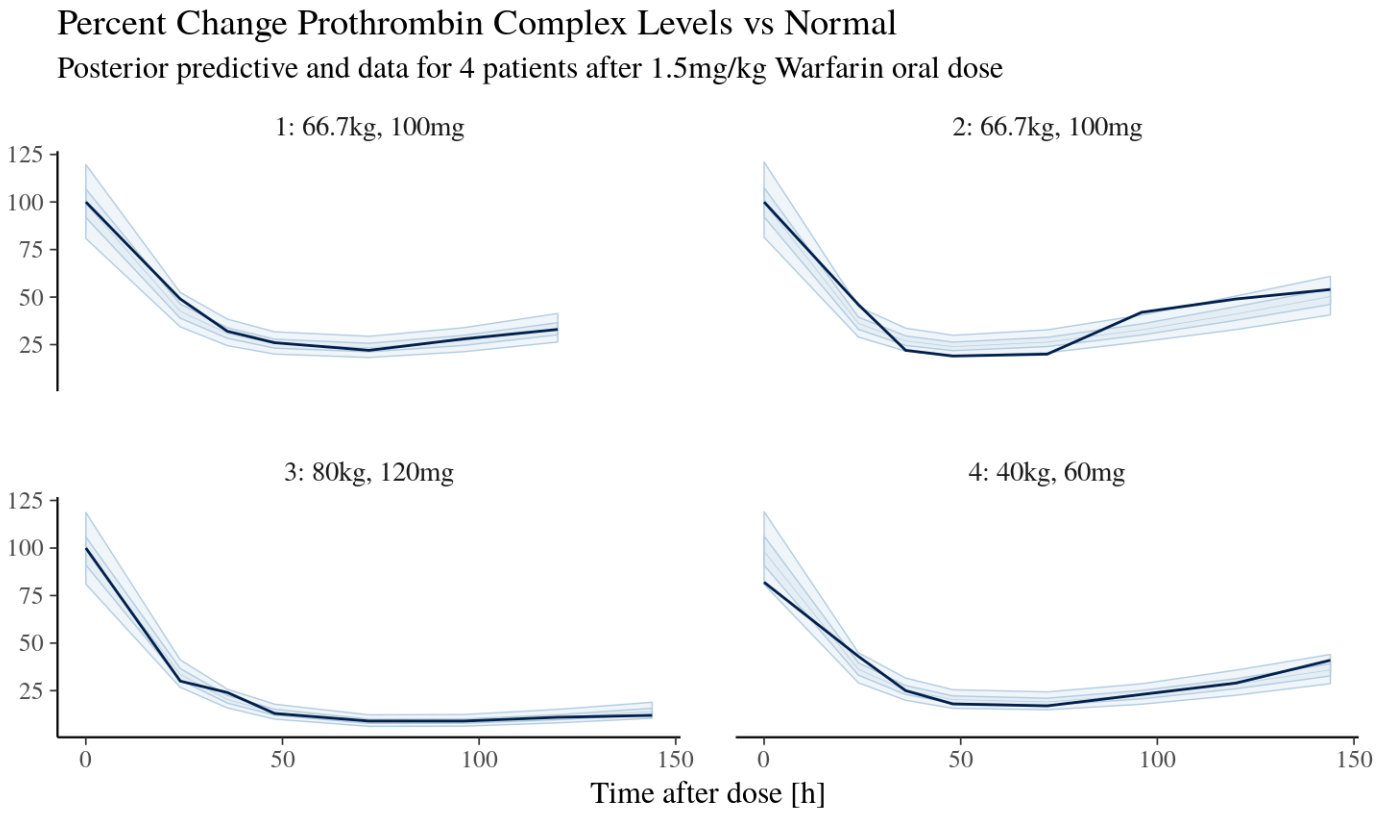
- PK part is easy; PD part entails solving an ODE numerically

$$\frac{dR_i}{dt} = k_i^{in}\left(1 - \frac{C_i(t)}{C_i(t) + EC50_i}\right) - k_i^{out} R_i$$

$$\log(R_i(0)) \sim \mathcal{N}(\theta_1, \omega_1); \log\left(k_i^{out}\right) \sim \mathcal{N}(-\theta_2, \omega_2); \log(EC50_i) \sim \mathcal{N}(\theta_3, \omega_3)$$

```
  real[] turnover_kin_inhib_2(real t, real[] R, real[] theta, real[] x_r, int[] x_i) {
//real ldose=x_r[1]; real llag=x_r[2]; real lka=x_r[3]; real lCl=x_r[4]; real lV=x_r[5];
    real lconc = pk_1cmt_oral_tlag_t(t, x_r[1], x_r[2], x_r[3], x_r[4], x_r[5]);
    real lkout = -theta[2];
    real lkin = theta[1] + lkout;
    real lEC50 = theta[3];
    real lS = log_inv_logit(lconc - lEC50);
    return { exp(lkin + log1m_exp(lS)) - R[1] * exp(lkout) };
  }
```

Posterior Predictions from a Warfarin Model



Example was taken from Sebastian Weber's presentation at StanCon2018 Helsinki

Conclusion

- Stan can be and has been used for medical research
- Novartist and AstraZeneca have contributed both funding and code to Stan
- Stan has the most advanced MCMC algorithm, which will either work or fail with warnings
- Without constraints that are self-imposed by Gibbs samplers, you can use intuitive priors, user-defined functions, numerical solutions to ODEs, etc.
- The **rstantools** R package makes it easy to bundle up *your* Stan program for CRAN so that others can use it, like **rstanarm**, **trialr**, **RBesT** and many others
- The **brms** R package does almost all regression models you might need
- The **bayesplot** and **shinystan** packages help you visualize the output
- Check out http://mc-stan.org/ and https://discourse.mc-stan.org/ !