Workbook for Introduction to modeling with Stan

Logistic regression models

2023-07-13

Contents

Preliminaries for R examples

```
library(tidyverse)
library(stringr)
library(mgcv)
library(rstan)
library(tidybayes)
library(bayesplot)

theme_set(theme_bw())
bayesplot::color_scheme_set("viridis")
```

We'll continue to use the simulated data from Zecchin et al., as posted on the DDMoRe repository.

The objective of this analysis is to explore the relationship between ECOG status (0 vs 1) and baseline tumor size. We will fit two models, compare them using LOO-IC and VPCs.

We'll start by reading the data

```
# OS data
d <- read_csv('../data/source/DDmodel0218_Simulated_OS.csv', na = c('.','-99'))

# Add week 12 (Day 84) predicted tumor size
d84 <- d %>%
  filter(TIME <= 84) %>%
  group_by(ID) %>%
  mutate(rate = KG/1000 - KD0/1000*AUC0 - KD1/100*AUC1,
         prevTIME = lag(TIME,default = 0),
         change = exp(rate * (TIME-prevTIME)),
         ipred = IBASE * 1000 * cumprod(change)
         ) %>%
  arrange(ID,TIME) %>%
  slice(n()) %>%
  mutate(ipred84 = ipred * exp(rate * (84-TIME)),
         rts84 = ipred84 / ( IBASE * 1000 ) )

dos <- d %>%
```

```
  filter(TIME>0) %>%
  group_by(ID) %>%
  mutate(meanGem = mean(AUC1),
         Group = if_else(meanGem > 0, "Cb+G", "Cb")) %>%
  ungroup() %>%
  filter(CMT==2, EVID==0) %>%
  left_join(d84 %>% select(ID, ipred84, rts84)) %>%
  mutate(rts84_f = paste0("Q", ntile(rts84, n = 4)))

dos84 <- dos %>%
  filter(TIME>84) %>%
  mutate(TIME = TIME-84)
```

Next we'll fit the model from the slides, namely

$$\text{logit}\left(P(\text{ECOG} = 1|x)\right) = \alpha + \beta x$$

Open the file `../model/stan/logistic_regression_example.stan`. Look over the four code blocks. Do you understand what each of these is doing? If not, ask :)

We'll start by doing a prior-predictive check to see if our prior distributions make sense. To do this, we'll set `prior_only=1` in out data. Based on your review of the code, what do you expect this to do?

Use the code below to put together the data for the model, set up the initial values and run the model.

```
stan_data <- list(N = nrow(dos),
                  Y = dos$ECOG,
                  x = dos$SLD0/10,
                  prior_only = 1)

stan_inits <- function() {
  list(
    alpha = rnorm(1, mean=0, sd=2),
    beta = rnorm(1, mean=0, sd = 1)
  )
}

fit_lr <- stan(file = '../model/stan/logistic_regression_example.stan',
               data = stan_data,
               chains = 4,iter = 2000, warmup = 1000,
               init = stan_inits,
               cores = 2,
               seed = 31425)
```

```
. Running /opt/R/4.1.3/lib/R/bin/R CMD SHLIB foo.c
. gcc -I"/opt/R/4.1.3/lib/R/include" -DNDEBUG   -I"/data/page/binary_tte_in_stan/rlibs/Rcpp/include/"  
. In file included from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Core:88:0,
.                  from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Dense:1,
.                  from /data/page/binary_tte_in_stan/rlibs/StanHeaders/include/stan/math/prim/mat/fun/
.                  from <command-line>:0:
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unkn
.  namespace Eigen {
.  ^~~~~~~~~
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: exp
.  namespace Eigen {
```

```
.                  ^
. In file included from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Dense:1:0,
.                  from /data/page/binary_tte_in_stan/rlibs/StanHeaders/include/stan/math/prim/mat/fun/
.                  from <command-line>:0:
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such
.  #include <complex>
.           ^~~~~~~~
. compilation terminated.
. /opt/R/4.1.3/lib/R/etc/Makeconf:168: recipe for target 'foo.o' failed
. make: *** [foo.o] Error 1
```

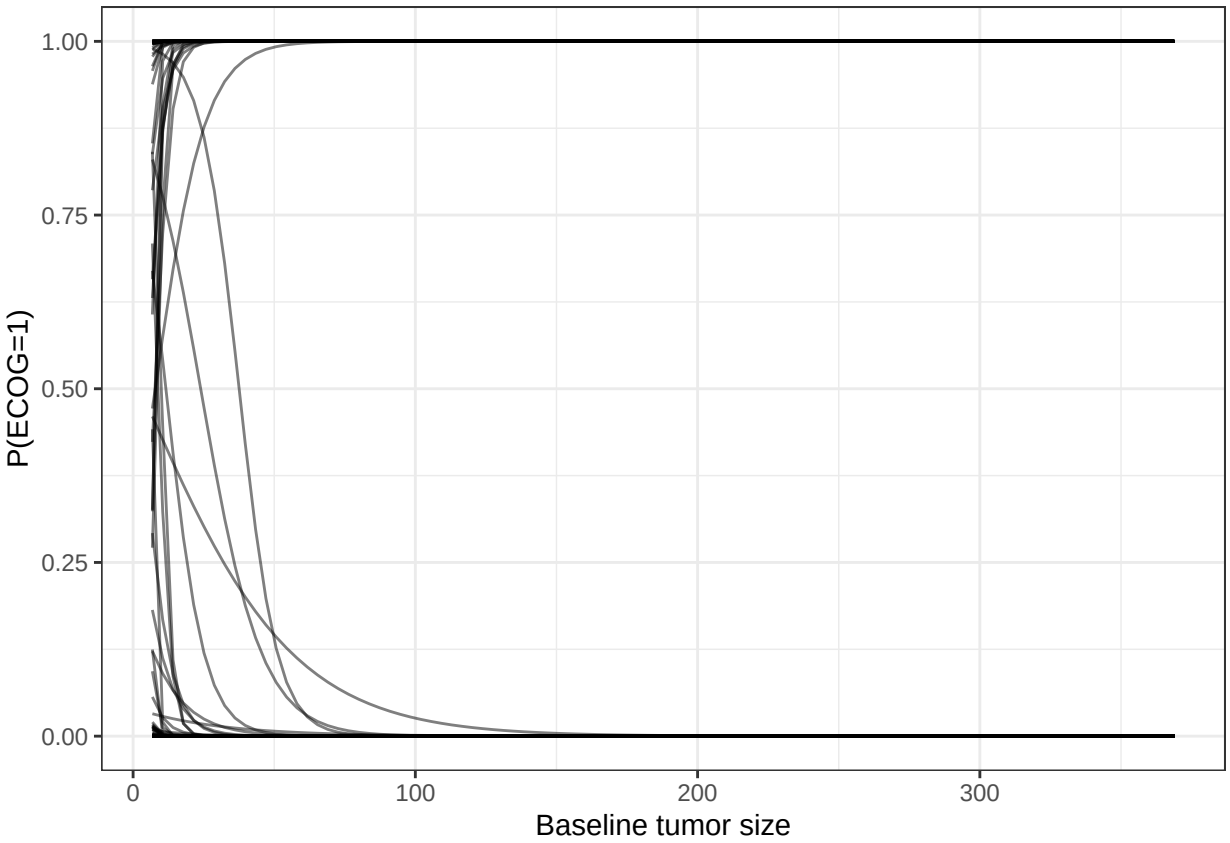
To assess the prior predictive relationship, we've just simulated from the prior distribution. Now we'll plot some of the sampled relationships between baseline tumor size and the probability of ECOG=1.

```
prior_samples = spread_draws(fit_lr, alpha, beta)
head(prior_samples)
```

```
. # A tibble: 6 x 5
.   .chain .iteration .draw alpha   beta
.    <int>      <int> <int> <dbl>  <dbl>
. 1      1          1     1 -2.78 -0.549
. 2      1          2     2  6.32 -2.41
. 3      1          3     3 -1.04 -0.528
. 4      1          4     4 -3.00  4.76
. 5      1          5     5  2.03 -2.37
. 6      1          6     6  1.07 -3.07
```

Let's plot 100 curves

```
prior_samples %>%
  sample_n(size=200) %>%
  crossing(SLD0 = seq(min(dos$SLD0), max(dos$SLD0), length=100)) %>%
  ggplot(aes(x=SLD0, y=plogis(alpha + beta*SLD0), group=.draw)) +
  geom_line(alpha=0.5) +
  labs(x='Baseline tumor size', y='P(ECOG=1)')
```



Does this seem reasonable? If not, how might you modify the prior distributions to get something that seems more reasonable to you? Try modifying the priors and re-running the prior predictive check.

Once you're happy with your priors, we'll fit the model to the data (to get samples from the posterior distribution). Note the change of `prior_only` from 1 to 0.

```
stan_data <- list(N = nrow(dos),
                  Y = dos$ECOG,
                  x = dos$SLD0/10,
                  prior_only = 0)

fit_lr <- stan(file = '../model/stan/logistic_regression_example.stan',
               data = stan_data,
               chains = 4,iter = 2000, warmup = 1000,
               init = stan_inits,
               cores = 2, seed = 31425)
```

```
posterior_samples = spread_draws(fit_lr, alpha, beta) %>%
  rename(Chain = .chain)  # The bayesplot functions look for a variable `Chain`
head(posterior_samples)
```

```
. # A tibble: 6 x 5
.   Chain .iteration .draw  alpha   beta
.   <int>      <int> <int>  <dbl>  <dbl>
. 1     1          1     1 -0.269 0.0258
. 2     1          2     2 -0.389 0.0594
. 3     1          3     3 -0.465 0.0584
. 4     1          4     4 -0.570 0.0551
```

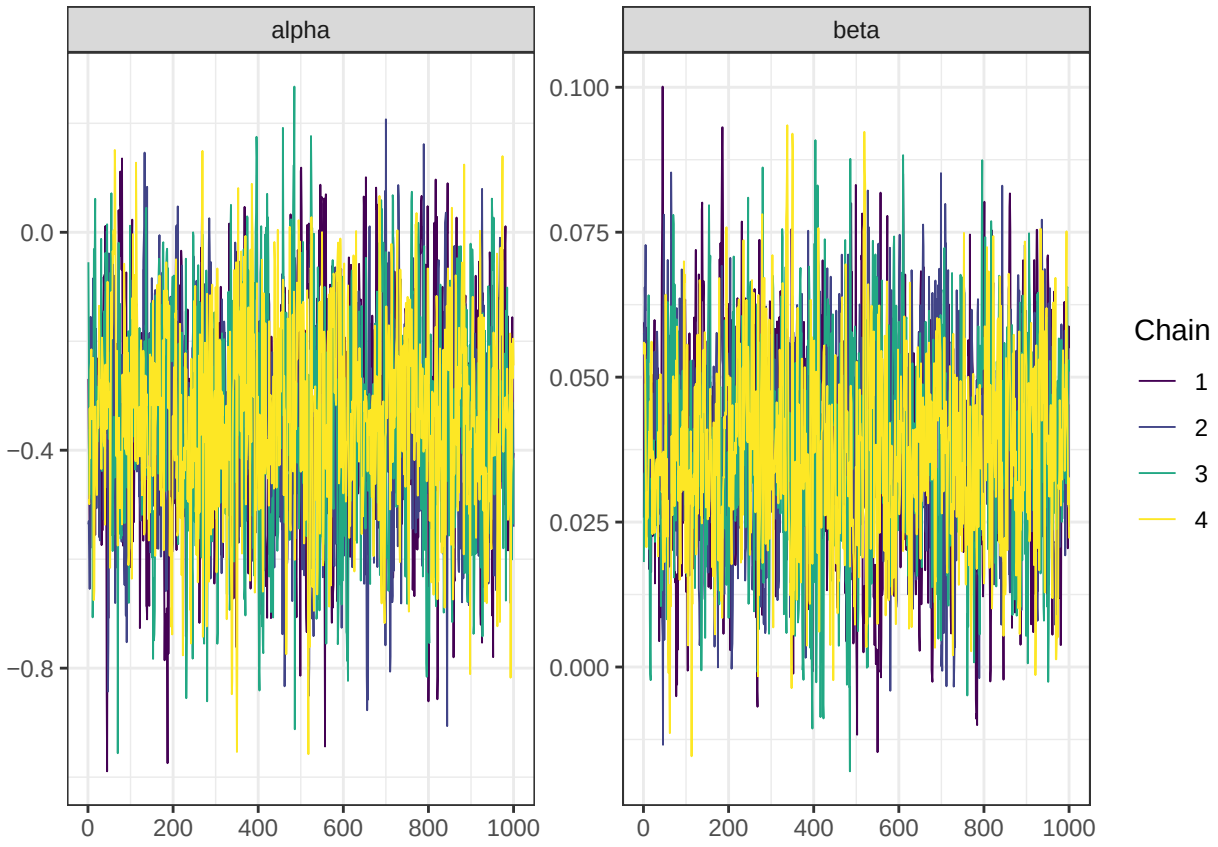
```
. 5     1           5     5 -0.578 0.0577
. 6     1           6     6 -0.579 0.0585
```

First, we'll check for convergence of the Markov chains by looking at the Rhat values and trace plots

```
print(fit_lr, pars=c('alpha','beta'))
```

```
. Inference for Stan model: logistic_regression_example.
. 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
. alpha -0.34    0.01 0.19 -0.72 -0.47 -0.34 -0.21  0.01  1309 1.00
. beta   0.04    0.00 0.02  0.00  0.03  0.04  0.05  0.07  1416 1.01
.
. Samples were drawn using NUTS(diag_e) at Thu Jul 13 16:39:08 2023.
. 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).
```

```
mcmc_trace(posterior_samples, pars=c('alpha','beta'))
```



Based on these diagnostics, do you think the sampler has converged?

Let's look at a VPC as a function of baseline tumor size. First, we'll define our summary function.

```
summary_function <- function(.data, .x_name, .y_name='value') {
  .data <- .data %>% ungroup() %>% rename('xvar' = .x_name, 'yvar'=.y_name)
  x_grid <- with(.data, seq(from = min(xvar),
```

```
                          to = quantile(xvar,probs = 0.95),
                          length = 100))
  fit <- gam(yvar ~ s(xvar), family=binomial(link='logit'), data=.data)
  predictions <- predict(fit, newdata = data.frame(xvar=x_grid), type='response')
  return( data.frame(xvar=x_grid, prediction = predictions))
}
```

Next, we'll compute the summary statistics for the observed data

```
obs_summary <- summary_function(dos, .x_name = 'SLD0', .y_name='ECOG') %>%
  mutate(type='Observed')

head(obs_summary)
```

```
.        xvar prediction     type
. 1  6.838455  0.4216450 Observed
. 2  8.875752  0.4235038 Observed
. 3 10.913050  0.4253647 Observed
. 4 12.950347  0.4272278 Observed
. 5 14.987645  0.4290930 Observed
. 6 17.024942  0.4309602 Observed
```

Next, we'll extract the posterior predictive samples and calculate the smooth relationship for each simulated dataset

```
posterior_predictive_samples = gather_draws(fit_lr, Ysim[ID]) %>%
  rename(Chain = .chain)

head(posterior_predictive_samples)
```

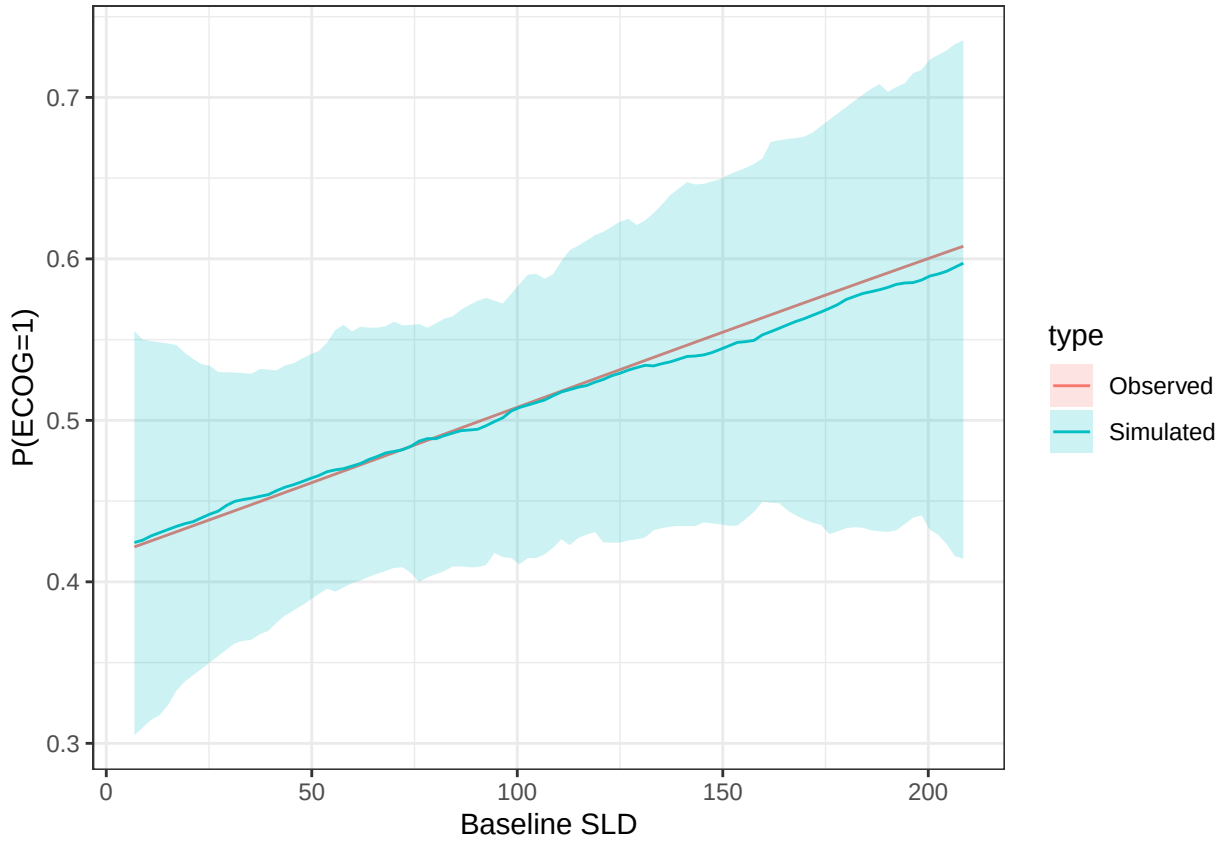
```
. # A tibble: 6 x 6
. # Groups:   ID, .variable [1]
.      ID Chain .iteration .draw .variable .value
.   <int> <int>      <int> <int> <chr>      <dbl>
. 1     1     1          1     1 Ysim           0
. 2     1     1          2     2 Ysim           1
. 3     1     1          3     3 Ysim           1
. 4     1     1          4     4 Ysim           1
. 5     1     1          5     5 Ysim           1
. 6     1     1          6     6 Ysim           1
```

```
sim_summary <- posterior_predictive_samples %>%
  # Join original data
  left_join(dos %>% select(ID,SLD0)) %>%
  # Nest everything except the simulation name
  nest(cols=-.draw) %>%
  # Use 200 sims for demonstration
  sample_n(size=200) %>%
  # Compute summary stats for each simulated dataset
  mutate(predictions = map(cols, ~summary_function(.x,
                                                   .x_name='SLD0',
                                                   .y_name='.value'))) %>%
  select(.draw,predictions) %>%
  unnest(cols=predictions) %>%
  # Summarise across simulated data sets
  group_by(xvar) %>%
```

```
  summarise(qlo = quantile(prediction, probs = 0.05),
            qhi = quantile(prediction, probs = 0.95),
            prediction=median(prediction)
            ) %>%
  mutate(type = 'Simulated')
```

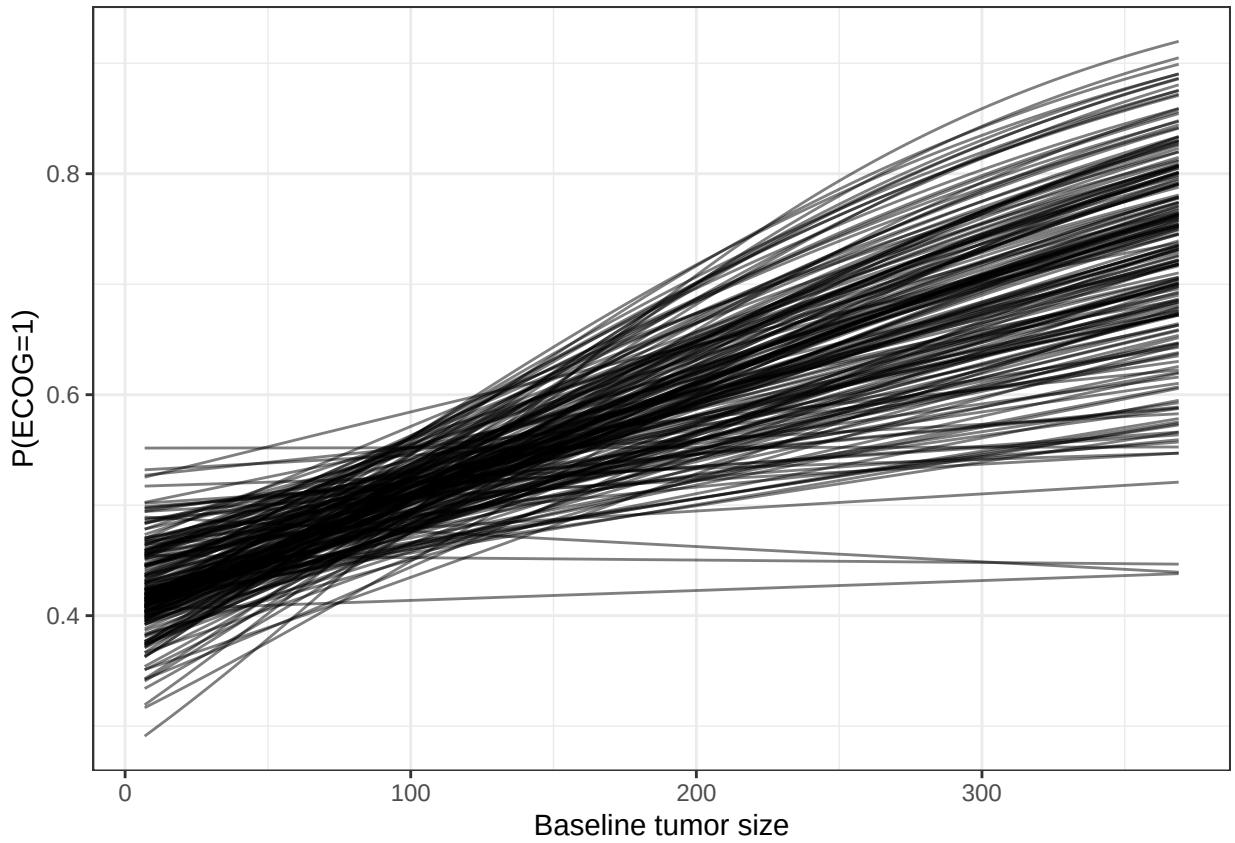
Plot VPC

```
sim_summary %>% bind_rows(obs_summary) %>%
  ggplot(aes(x=xvar, y=prediction)) +
  geom_line(aes(col=type, group=type)) +
  geom_ribbon(aes(ymin=qlo, ymax=qhi, fill=type), alpha=0.2) +
  labs(x='Baseline SLD', y='P(ECOG=1)')
```



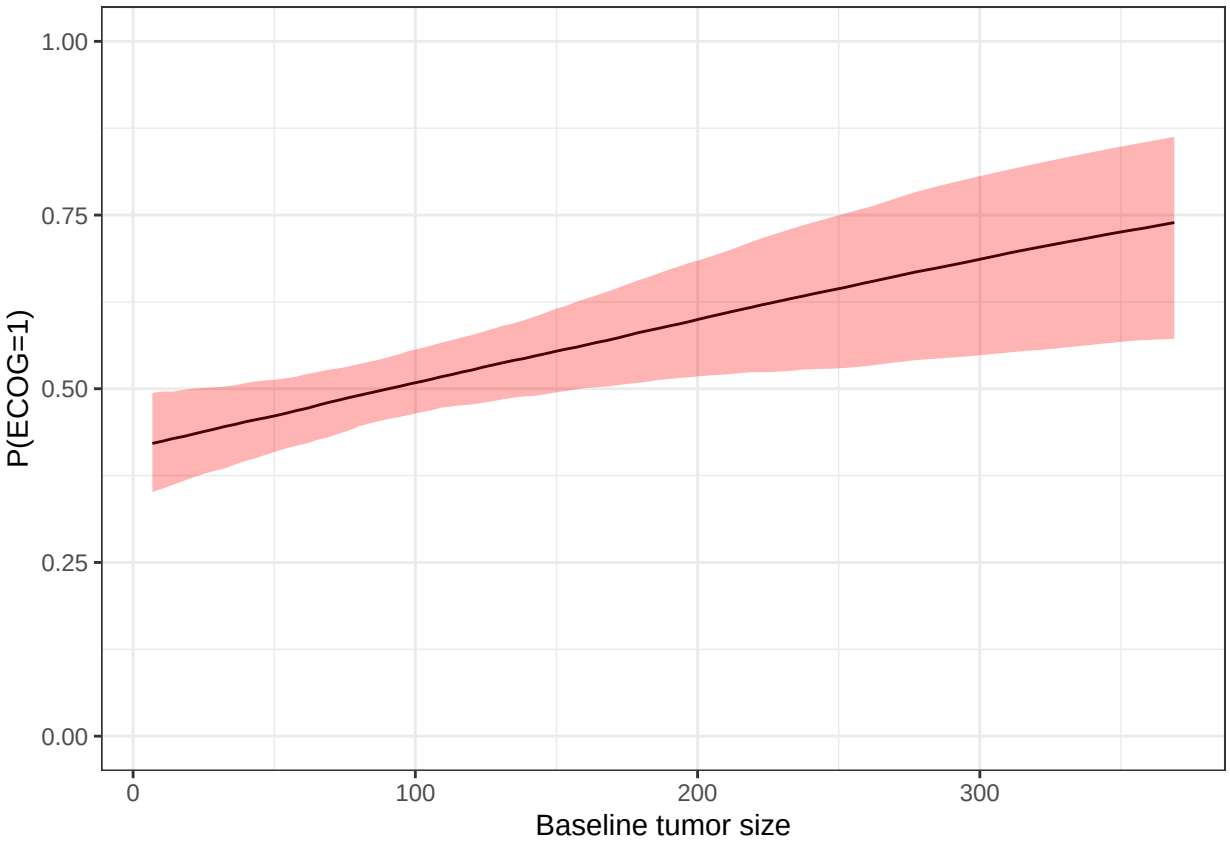
Let's plot 100 curves from the posterior

```
posterior_samples %>%
  sample_n(size=200) %>%
  crossing(SLD0 = seq(min(dos$SLD0), max(dos$SLD0), length=100)) %>%
  ggplot(aes(x=SLD0, y=plogis(alpha + beta*SLD0/10), group=.draw)) +
  geom_line(alpha=0.5) +
  labs(x='Baseline tumor size', y='P(ECOG=1)')
```



Let's plot the estimated relationship and pointwise 90% credible interval

```
posterior_samples %>%
  sample_n(size=500) %>%
  crossing(SLD0 = seq(min(dos$SLD0), max(dos$SLD0), length=100)) %>%
  # NB: We passed SLD0 in cm into the Stan model...
  mutate(phat = plogis(alpha + beta*SLD0/10)) %>%
  group_by(SLD0) %>%
  mutate(est = median(phat), lcl=quantile(phat, probs = 0.05), ucl=quantile(phat, probs=0.95)) %>%
  ggplot(aes(x=SLD0, y=est)) +
  geom_line() +
  geom_ribbon(aes(ymin=lcl, ymax=ucl), fill='red', alpha=0.30) +
  ylim(0,1) +
  labs(x='Baseline tumor size', y='P(ECOG=1)')
```


1.00
0.75
0.50
0.25
0.00
P(ECOG=1)
0
100
200
300
Baseline tumor size

*** Exercises: ***

1. Inspect the Stan model file `../model/stan/logistic_regression_power.stan`
2. See how the code was modified to use a power model for SLD0 i.e., logit $(P(\text{ECOG} = 1|x)) = \alpha + \beta x^{\gamma}$. Focus on the parameters, model, and generated quantities blocks.
3. Run the model. Did you get some post-warmup divergent transitions? If so, follow the link provided by the Stan output (http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup) to read more.

Answer:

```
fit_lr_power <- stan(file = '../model/stan/logistic_regression_power.stan',
                data = stan_data,
                chains = 4,iter = 2000, warmup = 1000,
                cores = 2,  seed=31425)
```

```
. Running /opt/R/4.1.3/lib/R/bin/R CMD SHLIB foo.c
. gcc -I"/opt/R/4.1.3/lib/R/include" -DNDEBUG   -I"/data/page/binary_tte_in_stan/rlibs/Rcpp/include/"  
. In file included from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Core:88:0,
.                  from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Dense:1,
.                  from /data/page/binary_tte_in_stan/rlibs/StanHeaders/include/stan/math/prim/mat/fun/
.                  from <command-line>:0:
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unkn
.  namespace Eigen {
.  ^~~~~~~~~
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: exp
.  namespace Eigen {
.                  ^
. In file included from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Dense:1:0,
.                  from /data/page/binary_tte_in_stan/rlibs/StanHeaders/include/stan/math/prim/mat/fun/
.                  from <command-line>:0:
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such
.  #include <complex>
.           ^~~~~~~~~
. compilation terminated.
. /opt/R/4.1.3/lib/R/etc/Makeconf:168: recipe for target 'foo.o' failed
. make: *** [foo.o] Error 1
```

```
print(fit_lr_power,pars = c('alpha','beta','gamma'))
```

```
. Inference for Stan model: logistic_regression_power.
. 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
. alpha -0.63    0.02 0.46 -1.62 -0.90 -0.59 -0.35  0.26   602 1.00
. beta   0.31    0.01 0.33 -0.20  0.10  0.22  0.47  1.10   503 1.00
. gamma  0.42    0.01 0.25  0.04  0.23  0.39  0.59  0.96   313 1.01
.
. Samples were drawn using NUTS(diag_e) at Thu Jul 13 16:40:35 2023.
. 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).
```

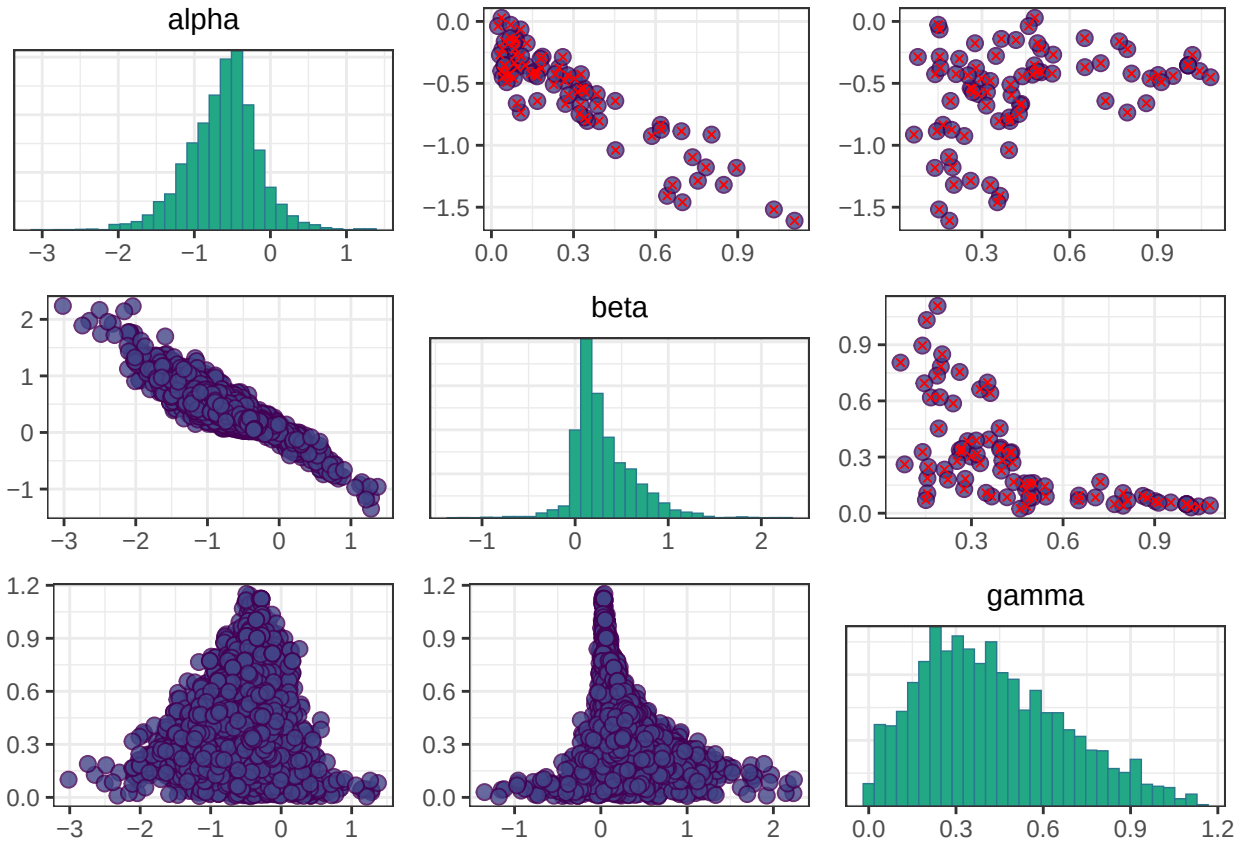
There are some divegent transitions, so we need to be cautious about interpreting the posterior summaries...

---

a. Look at the pairs plot for alpha, beta and the power parameter. What do you see? Try increasing the `adapt_delta` option to 0.95. Does that fix the problem?

Let's look at the pairs plot, highlighting the divergent transitions by putting them in the upper off-digonal plots.

```
mcmc_pairs(fit_lr_power,
           condition = pairs_condition(nuts='divergent__'),  # How to handle the divergent  transitions
           pars=c('alpha','beta','gamma'),                   # Which parameters to plot
           np = nuts_params(fit_lr_power))                   # Where to find the divergent transition i
```



The DTs seem to be scattered throughout the distributions. Let's increase the `adapt_delta` setting and re-run the model.

```
fit_lr_power_delta0.95 <- stan(file = '../model/stan/logistic_regression_power.stan',
                data = stan_data,
                chains = 4,iter = 2000, warmup = 1000,
                cores = 2,  seed=31425,
                control = list(adapt_delta=0.95))
```

We are still getting divergent transitions, so simply increasing the `adapt_delta` setting wasn't the solution.

---

b. Try a model which forces $\beta > 0$, does that fix the problem? Do you think this is a reasonable solution, or would you try a different functional form?

The new model code is in the file `../model/stan/logistic_regression_power_postive_beta.stan`.

```
fit_lr_power2 <- stan(file = '../model/stan/logistic_regression_power_positive_beta.stan',
                data = stan_data,
                chains = 4,iter = 2000, warmup = 1000,
                cores = 2,  seed=31425)
```

```
. Running /opt/R/4.1.3/lib/R/bin/R CMD SHLIB foo.c
. gcc -I"/opt/R/4.1.3/lib/R/include" -DNDEBUG   -I"/data/page/binary_tte_in_stan/rlibs/Rcpp/include/"  
. In file included from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Core:88:0,
.                  from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Dense:1,
.                  from /data/page/binary_tte_in_stan/rlibs/StanHeaders/include/stan/math/prim/mat/fun/
.                  from <command-line>:0:
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unkn
.  namespace Eigen {
.  ^~~~~~~~~
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: exp
.  namespace Eigen {
.                  ^
. In file included from /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Dense:1:0,
.                  from /data/page/binary_tte_in_stan/rlibs/StanHeaders/include/stan/math/prim/mat/fun/
.                  from <command-line>:0:
. /data/page/binary_tte_in_stan/rlibs/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such
.  #include <complex>
.           ^~~~~~~~~
. compilation terminated.
. /opt/R/4.1.3/lib/R/etc/Makeconf:168: recipe for target 'foo.o' failed
. make: *** [foo.o] Error 1
```

```
print(fit_lr_power2, pars=c('alpha','beta','gamma'))
```

```
. Inference for Stan model: logistic_regression_power_positive_beta.
. 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
. alpha -0.64    0.02 0.42 -1.64 -0.88 -0.58 -0.33 -0.04   515 1.01
. beta   0.32    0.01 0.31  0.01  0.09  0.22  0.46  1.15   540 1.01
. gamma  0.50    0.02 0.35  0.07  0.25  0.42  0.67  1.38   494 1.01
.
. Samples were drawn using NUTS(diag_e) at Thu Jul 13 16:41:56 2023.
. 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).
```

No divergent transitions (Yay!). We can probably address the Rhat and ESS values by running our chains a bit longer.

---

4. Check convergence and model fit using a VPC. For the VPC, you can use the code above with no changes.

Let's run the model with more posterior samples and nudging up the adapt delta paramter, then check the VPC.

```
fit_lr_power3 <- stan(file = '../model/stan/logistic_regression_power_positive_beta.stan',
               data = stan_data,
               chains = 4,iter = 4000, warmup = 2000,
               cores = 2,  seed=31425, control=list(adapt_delta=0.90))
```

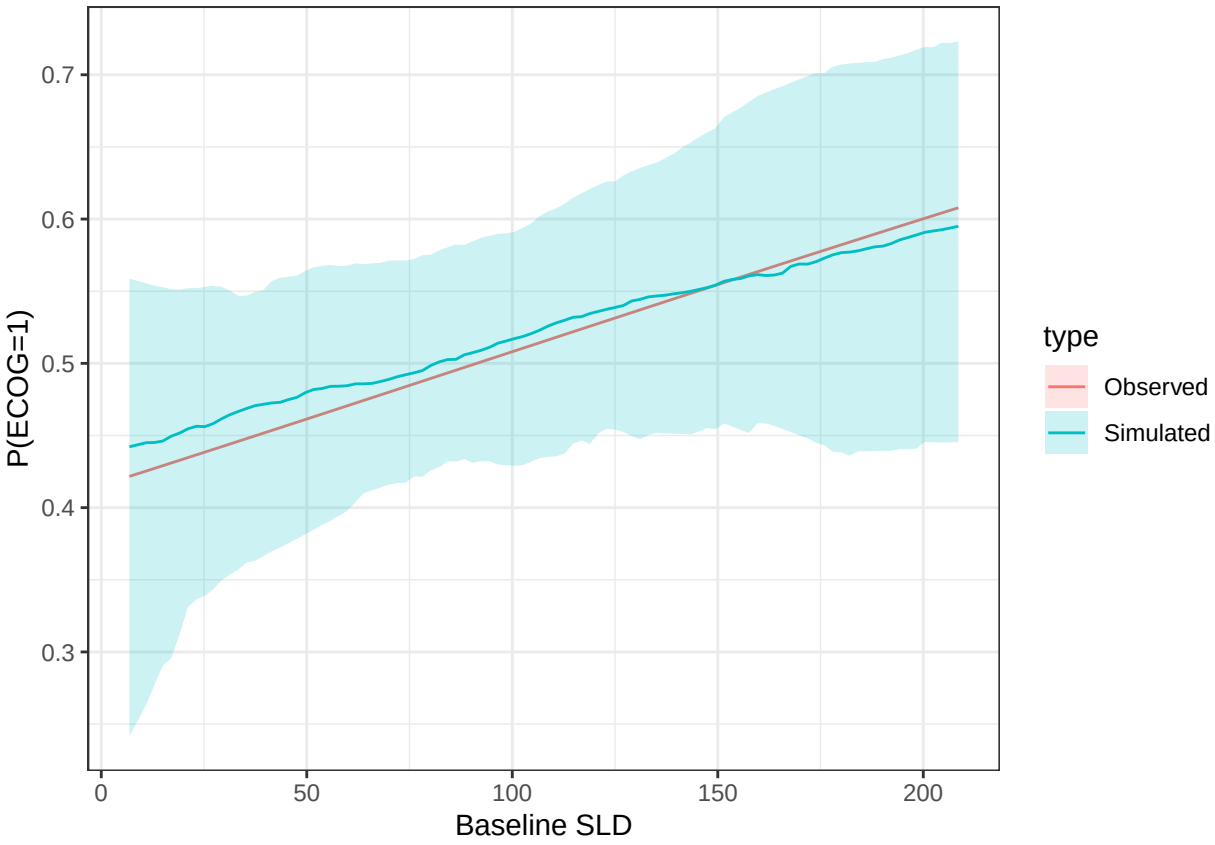
```
print(fit_lr_power3, pars=c('alpha','beta','gamma'))
```

```
. Inference for Stan model: logistic_regression_power_positive_beta.
. 4 chains, each with iter=4000; warmup=2000; thin=1;
. post-warmup draws per chain=2000, total post-warmup draws=8000.
.
.        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
. alpha -0.66    0.01 0.43 -1.71 -0.90 -0.58 -0.34 -0.03  1140    1
. beta   0.32    0.01 0.32  0.01  0.08  0.22  0.47  1.18  1127    1
. gamma  0.51    0.01 0.34  0.06  0.26  0.43  0.68  1.34  1364    1
.
. Samples were drawn using NUTS(diag_e) at Thu Jul 13 16:42:25 2023.
. 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).
```

```
posterior_predictive_samples3 = gather_draws(fit_lr_power3, Ysim[ID]) %>%
  rename(Chain = .chain)

sim_summary3 <- posterior_predictive_samples3 %>%
  left_join(dos %>% select(ID,SLD0)) %>%
  nest(cols=-.draw) %>%
  sample_n(size=200) %>%
  mutate(predictions = map(cols, ~summary_function(.x,
                                                   .x_name='SLD0',
                                                   .y_name='.value'))) %>%
  select(.draw,predictions) %>%
  unnest(cols=predictions) %>%
  group_by(xvar) %>%
  summarise(qlo = quantile(prediction, probs = 0.05),
            qhi = quantile(prediction, probs = 0.95),
            prediction=median(prediction)
            ) %>%
  mutate(type = 'Simulated')

sim_summary3 %>% bind_rows(obs_summary) %>%
  ggplot(aes(x=xvar, y=prediction)) +
  geom_line(aes(col=type, group=type)) +
  geom_ribbon(aes(ymin=qlo, ymax=qhi, fill=type), alpha=0.2) +
  labs(x='Baseline SLD', y='P(ECOG=1)')
```


0.7
0.6
0.5
0.4
0.3
P(ECOG=1)
0
50
100
150
200
Baseline SLD
type
Observed
Simulated

5. Compare models using LOO-IC. Which model do you think will provide better predictions for future patients?

Answer:

```
loo_base <- loo(fit_lr)
loo_power <- loo(fit_lr_power3)

print(loo_base)
```

```
.
. Computed from 4000 by 336 log-likelihood matrix
.
.          Estimate  SE
. elpd_loo   -232.5 2.1
. p_loo         1.9 0.1
. looic       465.0 4.3
. ------
. 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.
```

```
print(loo_power)
```

```
.
. Computed from 8000 by 336 log-likelihood matrix
.
.          Estimate  SE
. elpd_loo   -232.7 1.7
. p_loo         1.9 0.1
. looic       465.3 3.4
. ------
. 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.
```

```
loo::loo_compare(loo_base, loo_power)
```

```
.        elpd_diff se_diff
. model1  0.0       0.0
. model2 -0.2       0.5
```

Based on ELPD, the two models provide similar out-of-sample predictive performance. So, either model would be a reasonable choice (given that the VPCs look okay). Following the parsimony principle, we might select the simpler model.
