Assign_2

23370209_Franco Meng

2024-09-29

```r
library(ggplot2)
```

```r
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.6 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```r
library(bayesplot)
```

```
## This is bayesplot version 1.11.1
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

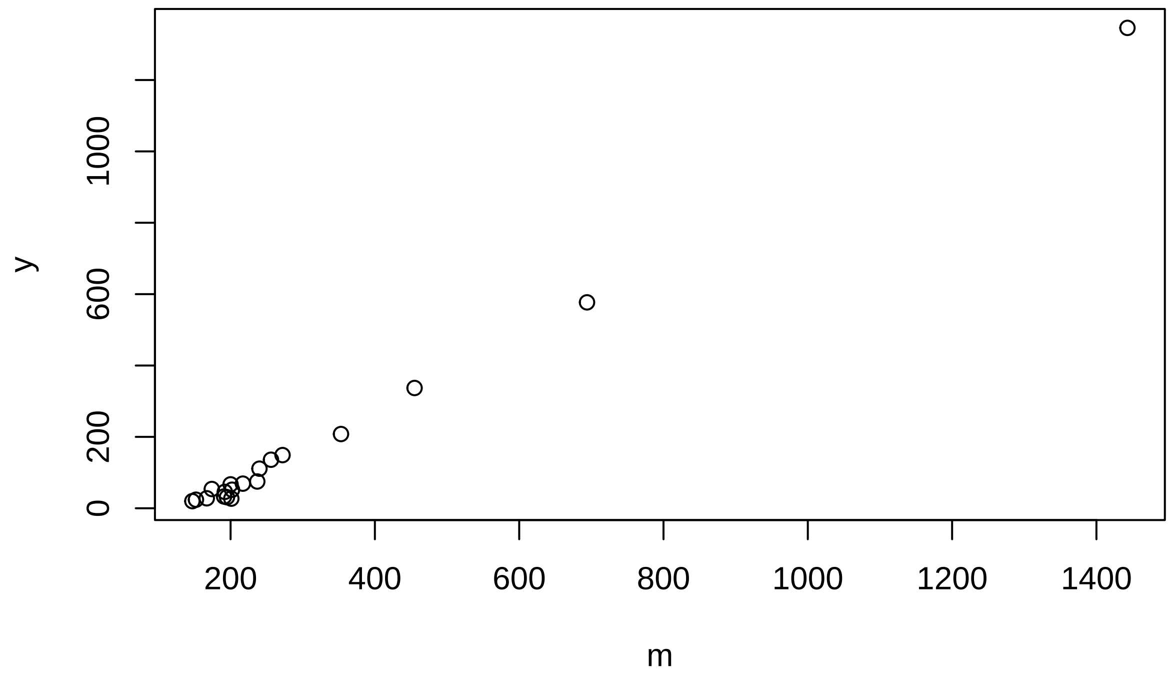
```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

```r
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

TASK 1

```r
dat <- read.table("Golf.csv", header=TRUE, sep = ",")
plot(y~m, dat)
```



```stan
  data{
    int <lower=1> n;
    vector[n] x;
    int<lower=1> m[n];
    int<lower=0> y[n];
    real <lower=0> alpha;
    real <lower=0> beta;
  }

  parameters{
    real<lower=0, upper=1> U;
    real<lower=U, upper=1> L;
    real<lower=0> a;
    real<lower=0> b;

  }

  model{
    for (i in 1:n){
      y[i] ~ binomial(m[i], L+ (U-L) / (1 + (x[i]/b)^(-a)));
    }
    // prior
    L ~ beta(alpha, beta); // This is not a hierarchical model, I have defined Alpha Beta value in Data
    U ~ beta(alpha, beta); // This is not a hierarchical model, I have defined Alpha Beta value in Data
    a ~ normal(0,1000);
    b ~ normal(0,1000);
  }
```

```
n <- NROW(dat)
data.in <- list(x=dat$distance, m=dat$m, y=dat$y, alpha=1, beta=1, n=n)
model.fit1 <- sampling(task_1a, data=data.in, iter=10000, warmup = 2000)
```

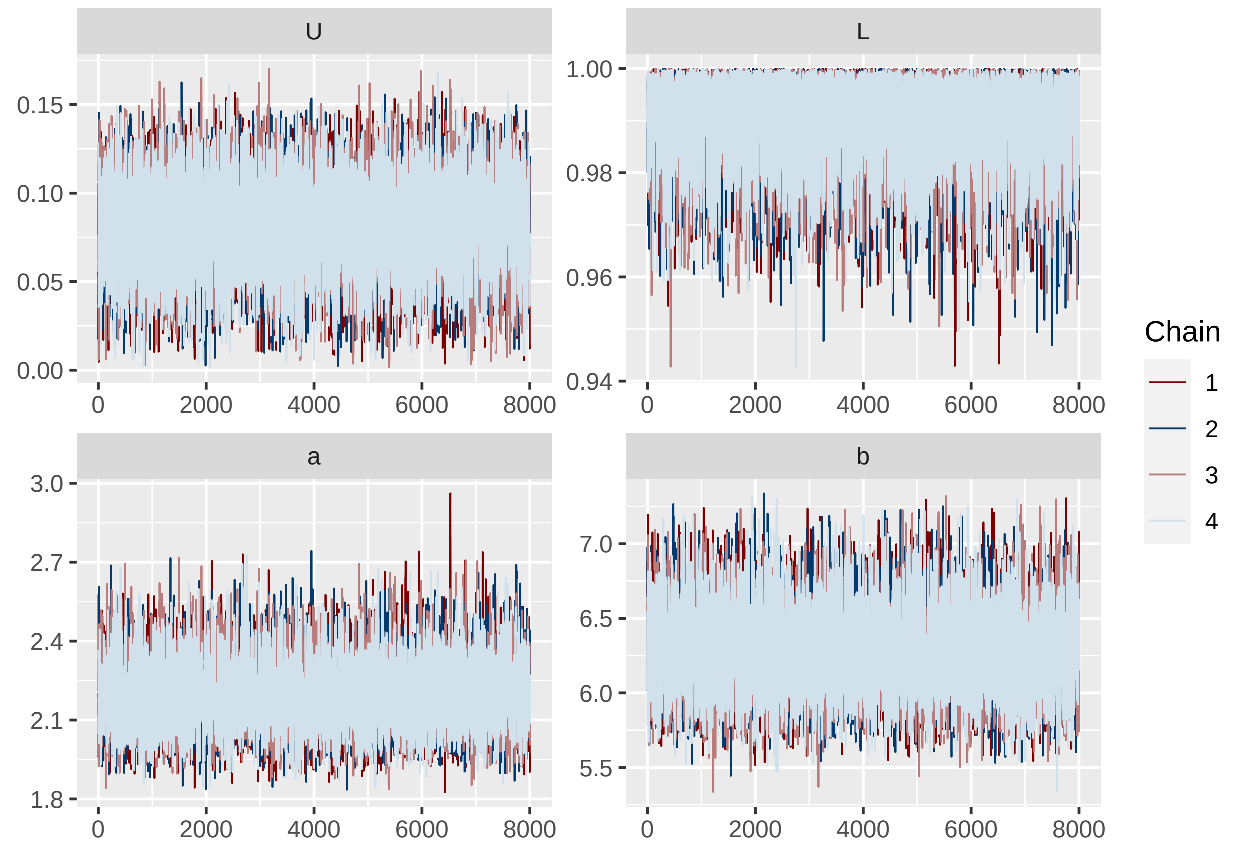
```
## Warning: There were 1 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

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

```
check_hmc_diagnostics(model.fit1)
```

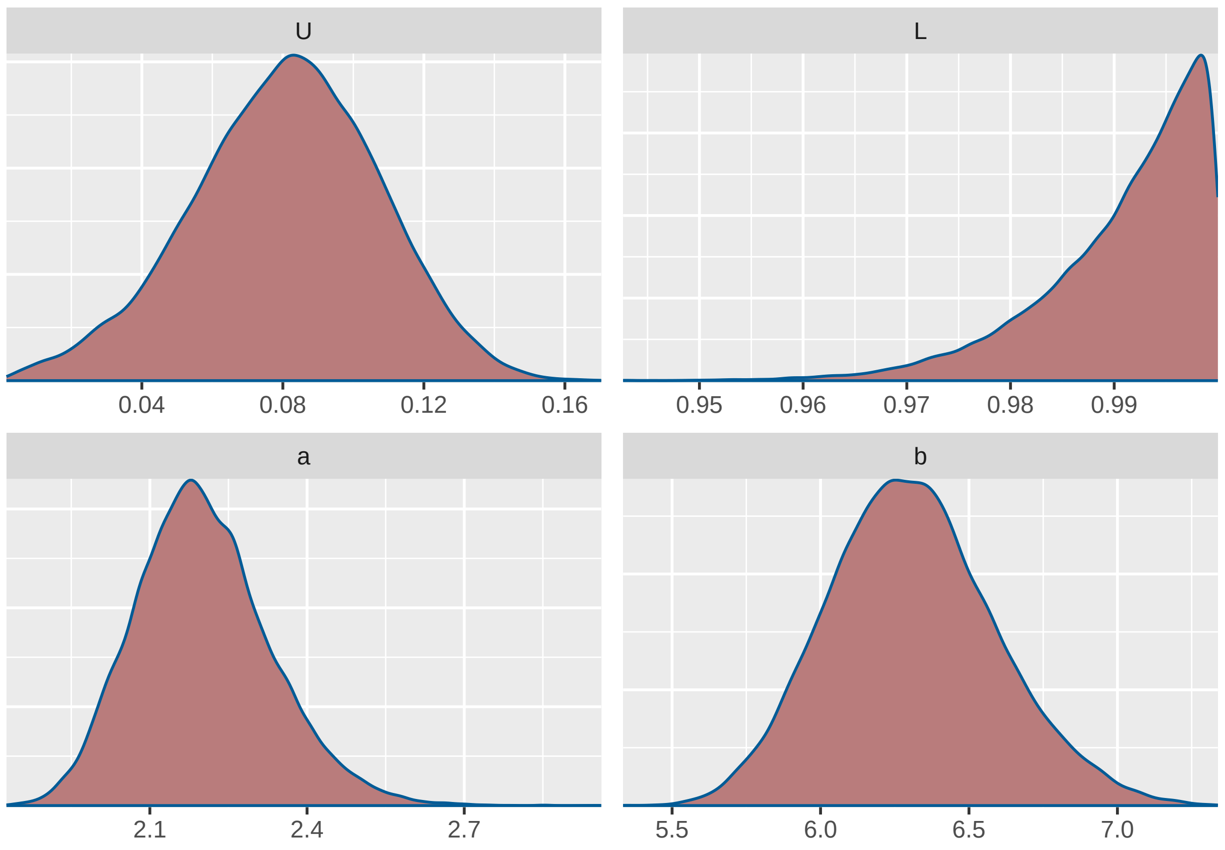
```
##
## Divergences:

## 1 of 32000 iterations ended with a divergence (0.003125%).
## Try increasing 'adapt_delta' to remove the divergences.

##
## Tree depth:

## 0 of 32000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

```
posterior_1 <- as.array(model.fit1)
```

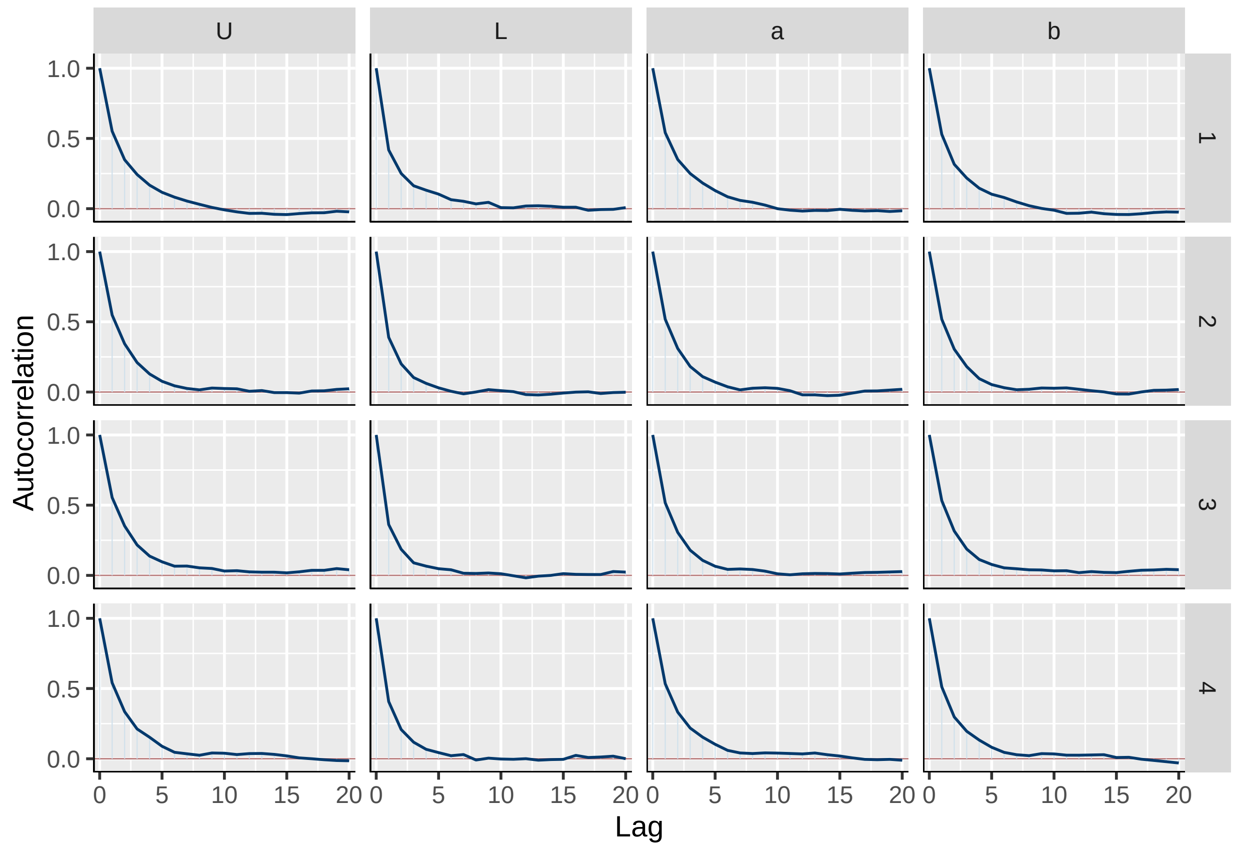
```
color_scheme_set("mix-blue-red")
mcmc_trace(posterior_1, pars = c("U", "L" , "a" , "b"))
```



```r
mcmc_dens(posterior_1, pars = c("U", "L" , "a" , "b"))
```



```r
mcmc_acf(posterior_1, pars = c("U", "L" , "a" , "b"))
```



```
print(model.fit1, pars = c("U", "L" , "a" , "b"), digits=5)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=2000; thin=1;
## post-warmup draws per chain=8000, total post-warmup draws=32000.
##
##      mean se_mean      sd    2.5%     25%     50%     75%   97.5% n_eff    Rhat
## U 0.08071 0.00030 0.02654 0.02544 0.06325 0.08167 0.09918 0.13027  7658 1.00048
## L 0.99089 0.00008 0.00790 0.97090 0.98686 0.99302 0.99697 0.99972 10938 1.00039
## a 2.20175 0.00144 0.12867 1.97509 2.11283 2.19300 2.28028 2.47990  7939 1.00049
## b 6.30769 0.00310 0.28291 5.78359 6.11072 6.29706 6.48907 6.90081  8314 1.00060
##
## Samples were drawn using NUTS(diag_e) at Sun Oct 13 13:08:34 2024.
## 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).
```

I have chosen the non-informative proper normal priors with mean 0 and sd 1000 for a and b. Which are : a ~ Normal (0, 1000^2), b ~ Normal (0, 1000^2), variance being 1000^2. These are in fact truncated normal due to a and b declared to be greater than 0.

For the L(lower) and U(upper) asymptote as defined in the task. I have used gamma distribution where alpha = 0.001, beta = 0.001 as priors for both parameters,

With 4 chains of 10000 Iterations (2000 warm-ups) each, my Bayesian estimate for parameters are:

U: 0.08 L: 0.99 a: 2.2 b: 6.3

```
data{
  int <lower=1> n;
  vector[n] x;
  int<lower=1> m[n];
  int<lower=0> y[n];
  real <lower=0> alpha;
  real <lower=0> beta;
}

parameters{
  real<lower=0, upper=1> U;
  real<lower=U, upper=1> L;
  real<lower=0> a;
  real<lower=0> b;
  real<lower=0> g;

}

model{
  for (i in 1:n){
    y[i] ~ binomial(m[i], L+ (U-L) / (1 + g * (x[i]/b)^(-a))^(1/g));
  }
  L ~ beta(alpha, beta); // This is not a hierarchical model, I have defined Alpha Beta value in Data
  U ~ beta(alpha, beta); // This is not a hierarchical model, I have defined Alpha Beta value in Data
  a ~ normal(0,1000);
  b ~ normal(0,1000);
  g ~ normal(0,1000);
}
```

```
model.fit1b <- sampling(task_1b, data=data.in, iter=10000, warmup = 2000)
```

```
## Warning: There were 22 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

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

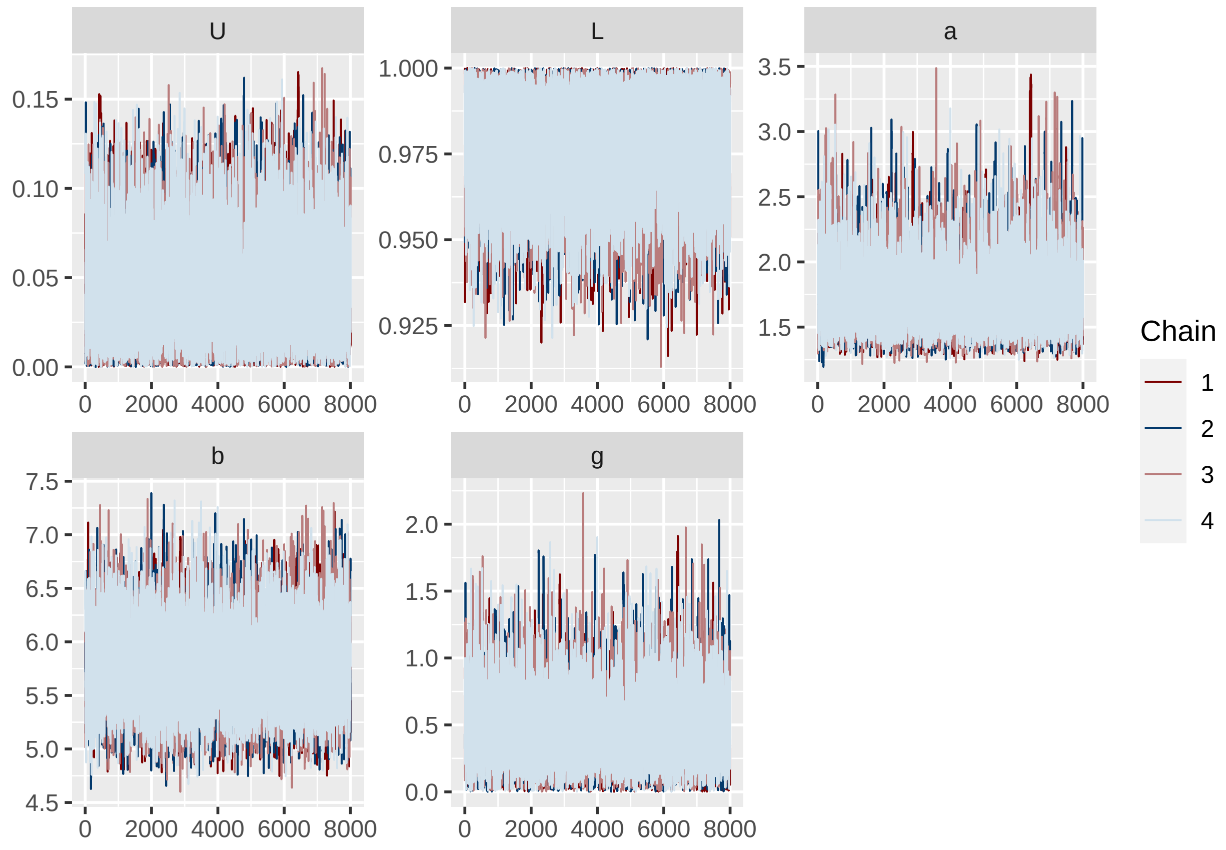
```
check_hmc_diagnostics(model.fit1b)
```

```
##
## Divergences:

## 22 of 32000 iterations ended with a divergence (0.06875%).
## Try increasing 'adapt_delta' to remove the divergences.

##
## Tree depth:

## 0 of 32000 iterations saturated the maximum tree depth of 10.

##
## Energy:
```

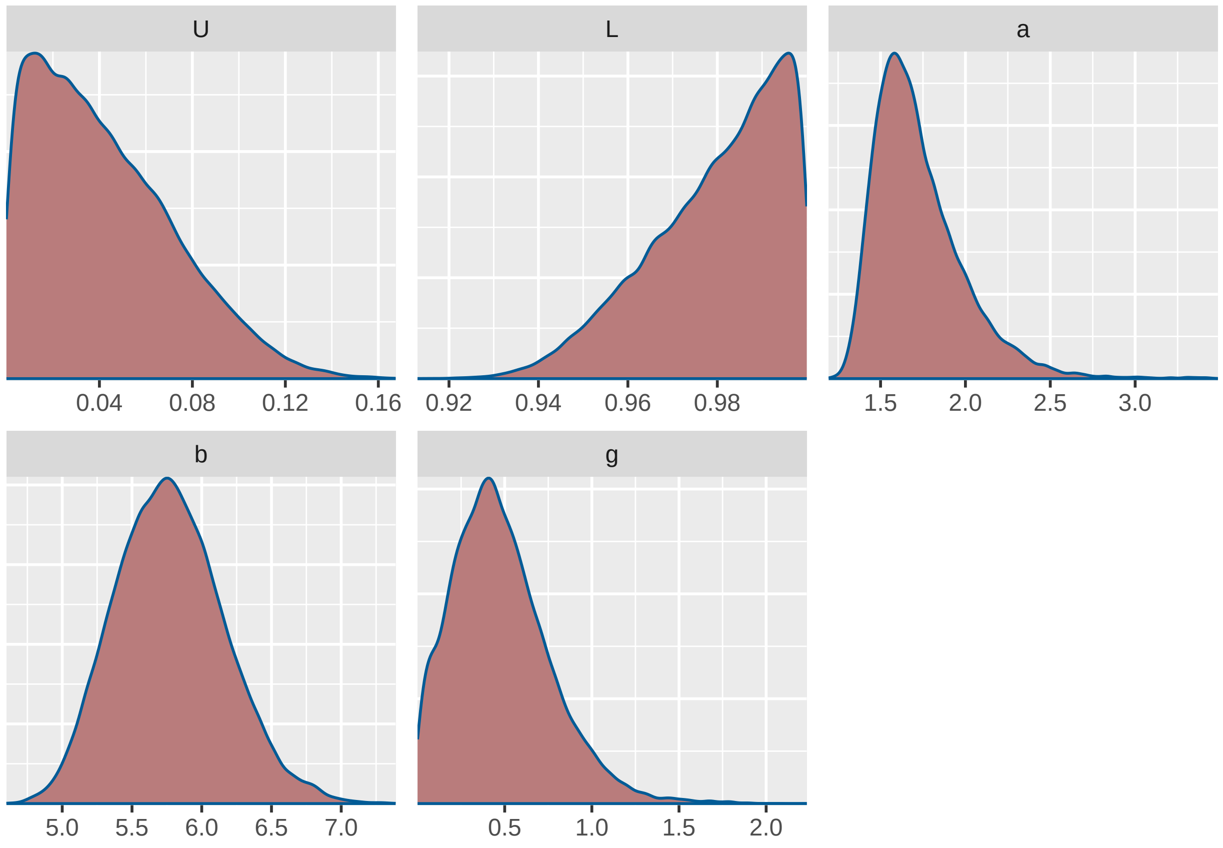
```
## E-BFMI indicated no pathological behavior.
```

```
posterior_1b <- as.array(model.fit1b)
```

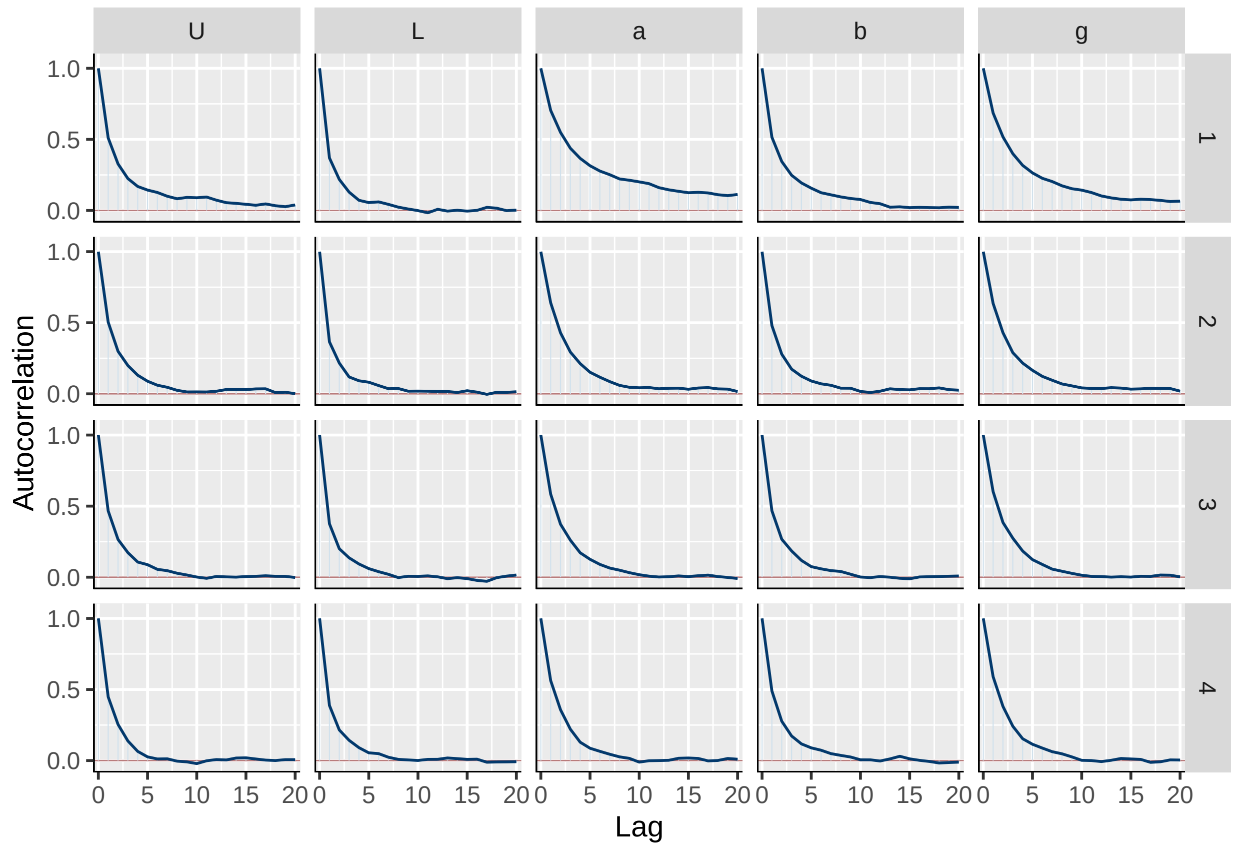
```
color_scheme_set("mix-blue-red")
mcmc_trace(posterior_1b, pars = c("U", "L" , "a" , "b", "g"))
```



```
mcmc_dens(posterior_1b, pars = c("U", "L" , "a" , "b", "g"))
```



```
mcmc_acf(posterior_1b, pars = c("U", "L" , "a" , "b", "g"))
```



```r
print(model.fit1b, pars = c("U", "L" , "a" , "b", "g"), digits=5)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=2000; thin=1;
## post-warmup draws per chain=8000, total post-warmup draws=32000.
##
##      mean se_mean      sd    2.5%     25%     50%     75%   97.5% n_eff    Rhat
## U 0.04224 0.00034 0.03007 0.00180 0.01749 0.03681 0.06196 0.11055  7613 1.00026
## L 0.98000 0.00015 0.01497 0.94611 0.97030 0.98309 0.99239 0.99930 10608 1.00018
## a 1.72688 0.00392 0.26476 1.37155 1.54042 1.67257 1.85788 2.36937  4554 1.00022
## b 5.77404 0.00456 0.38900 5.07239 5.49771 5.75838 6.02766 6.58999  7292 1.00009
## g 0.47992 0.00393 0.28143 0.04190 0.27639 0.44536 0.64104 1.12033  5139 1.00015
##
## Samples were drawn using NUTS(diag_e) at Sun Oct 13 13:08:52 2024.
## 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).
```

```r
extract_g <- extract(model.fit1b)$g
mean(abs(extract_g - 1) > 0.03)
```

```
## [1] 0.9847187
```

I have chosen the non-informative proper normal priors with mean 0 and sd 1000 for a, b and g. Which are : a ~ Normal (0, 1000ˆ2), b ~ Normal (0, 1000ˆ2),g ~ Normal (0, 1000ˆ2), variance being 1000ˆ2. These are in fact truncated normal due to a, b and g declared to be greater than 0.

For the L(lower) and U(upper) asymptote as defined in the task. I have used gamma distribution where alpha = 0.001, beta = 0.001 as priors for both parameters,

With 4 chains of 10000 Iterations (2000 warm-ups) each, my Bayesian estimate for parameters are:

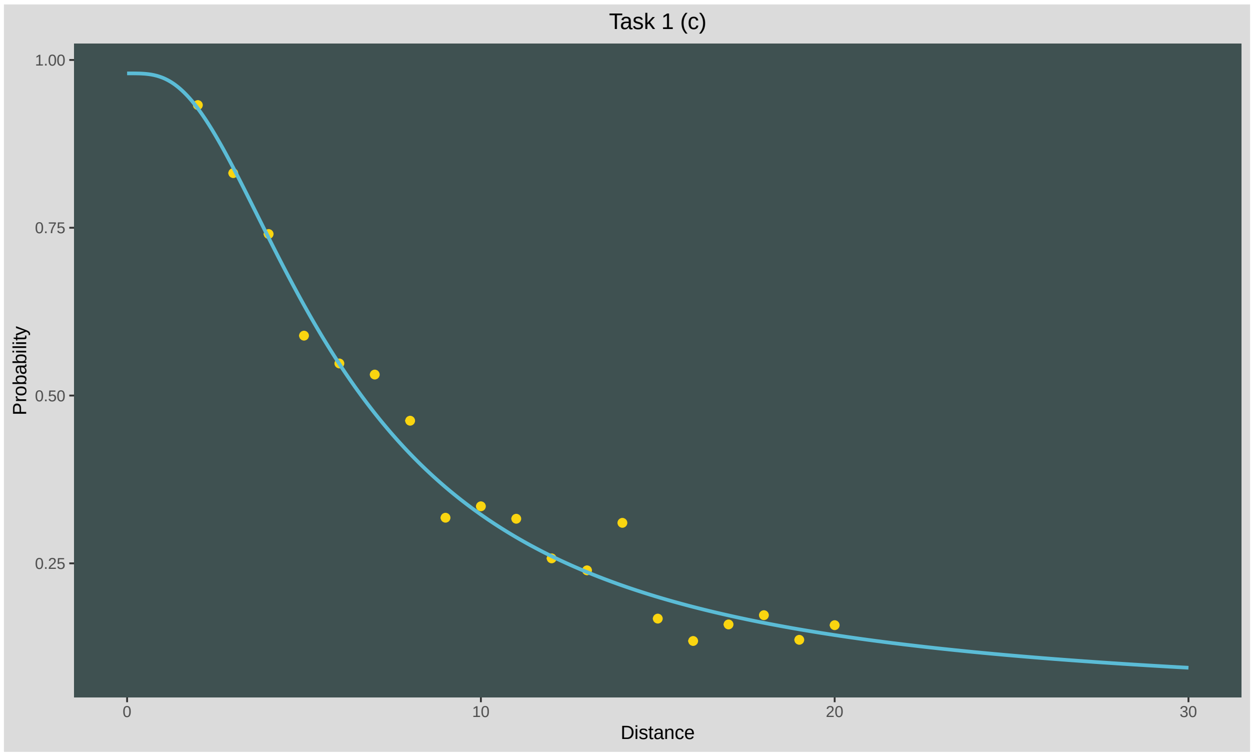
U: 0.04 L: 0.98 a: 1.7 b: 5.7 g: 0.5

The posterior probability that g differs from 1 by more than 3% is 0.98. Based on this result, I would prefer the second model. Where the introduction of g parameter may help the model to control the influences of distance on the success probability.

```
 fit1b <- extract(model.fit1b, c("U", "L" , "a" , "b", "g"))
 fit1b <- c(mean(fit1b$U),mean(fit1b$L),mean(fit1b$a),mean(fit1b$b),mean(fit1b$g))
 names(fit1b) <- c("U", "L" , "a" , "b", "g")
xgr <- with(dat, seq(from = 0, to = 30, length = 601))
line_to_plot <- data.frame(distance = xgr, ob_successful_proportion = fit1b["L"]+ (fit1b["U"]-fit1b["L"]
```

```
data_to_plot <- data.frame(distance = dat$distance, ob_successful_proportion = dat$y/dat$m)
#data_to_plot
ggplot() +
  geom_point(data = data_to_plot, aes(x = distance, y= ob_successful_proportion), color = "#FAD510", si
  geom_line(data = line_to_plot, aes(x=distance, y =ob_successful_proportion ), color = "#5BBCD6", size
  ggtitle("Task 1 (c)") +
  labs(x = "Distance", y = "Probability") +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "#3F5151"),
        plot.background = element_rect(fill = "gray86"))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

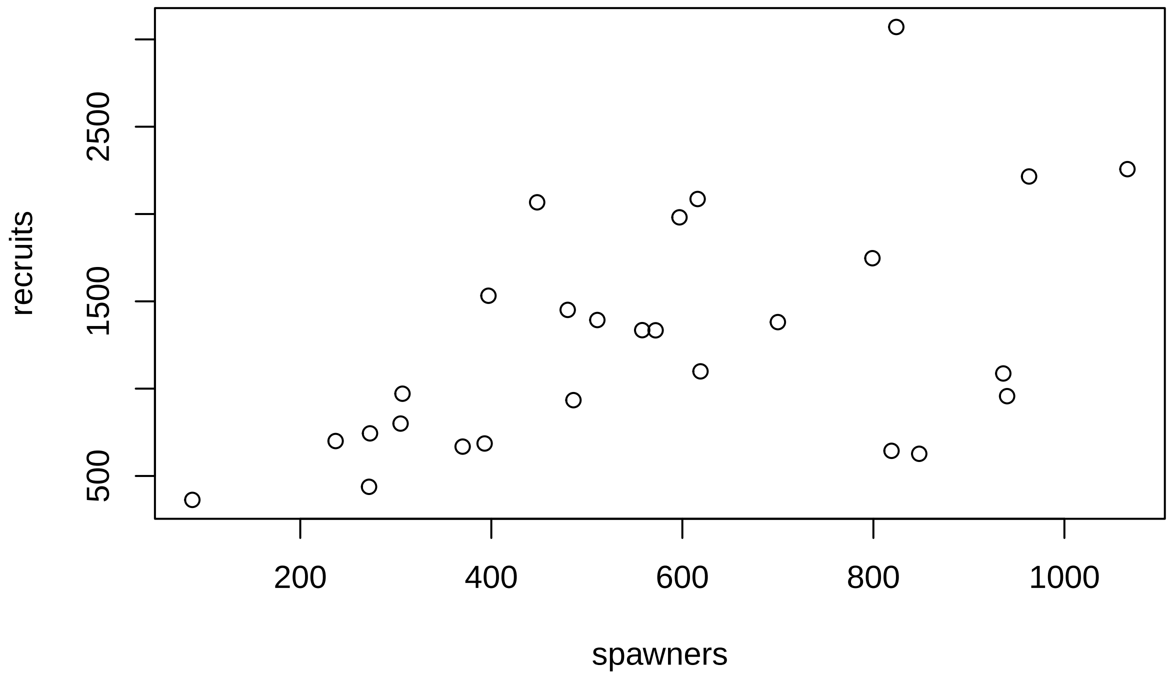


I have made the plot by using the second (preferred) model. Based on the plot, the model look sensible over the range of the observed distances. It seems safe to extrapolate the model the distances larger than observed, where the line start to flatten out and approaching to 0, it also seems safe to extrapolate to the shorter distance. but probably not when distance is 0 which means the ball is already in the hole.. But it is nice and realistic to see the model not modeling the probably of success to be 1 when distance is approaching 0.

```
dat_task2 <- read.table("SockeyeSR.csv", header=TRUE, sep = ",")
dat_task2 <- dat_task2[!dat_task2$year == 1951,]
dat_task2
```

```
##    year spawners recruits
## 1  1940      963     2215
## 2  1941      572     1334
## 3  1942      305      800
## 4  1943      272      438
## 5  1944      824     3071
## 6  1945      940      957
## 7  1946      486      934
## 8  1947      307      971
## 9  1948     1066     2257
## 10 1949      480     1451
## 11 1950      393      686
## 13 1952      237      700
## 14 1953      700     1381
## 15 1954      511     1393
## 16 1955       87      363
## 17 1956      370      668
## 18 1957      448     2067
## 19 1958      819      644
## 20 1959      799      1747
## 21 1960      273       744
## 22 1961      936      1087
## 23 1962      558      1335
## 24 1963      597      1981
## 25 1964      848       627
## 26 1965      619      1099
## 27 1966      397      1532
## 28 1967      616      2086
```

```
plot(recruits~spawners, dat_task2)
```



```
  data{
    int<lower=1> n;
    vector[n] x;
    vector[n] y;
  }

  parameters{
    real<lower=0> tau;
    real alpha;
    real beta;
  }

  transformed parameters{
    vector[n] mu;

    real<lower=0> sigma;
    real<lower=0> sigma2;
    sigma2 = 1/tau;
    sigma = sqrt(sigma2);
    mu = alpha + beta * x;

  }

  model{
    // likelihood
    for(i in 1:n){
    log(y[i]/x[i])  ~ normal(mu[i], sigma);
  }

    alpha ~ normal(0, 1000);
    beta ~ normal(0, 1000);
    tau ~ gamma(0.001, 0.001);
  }
```

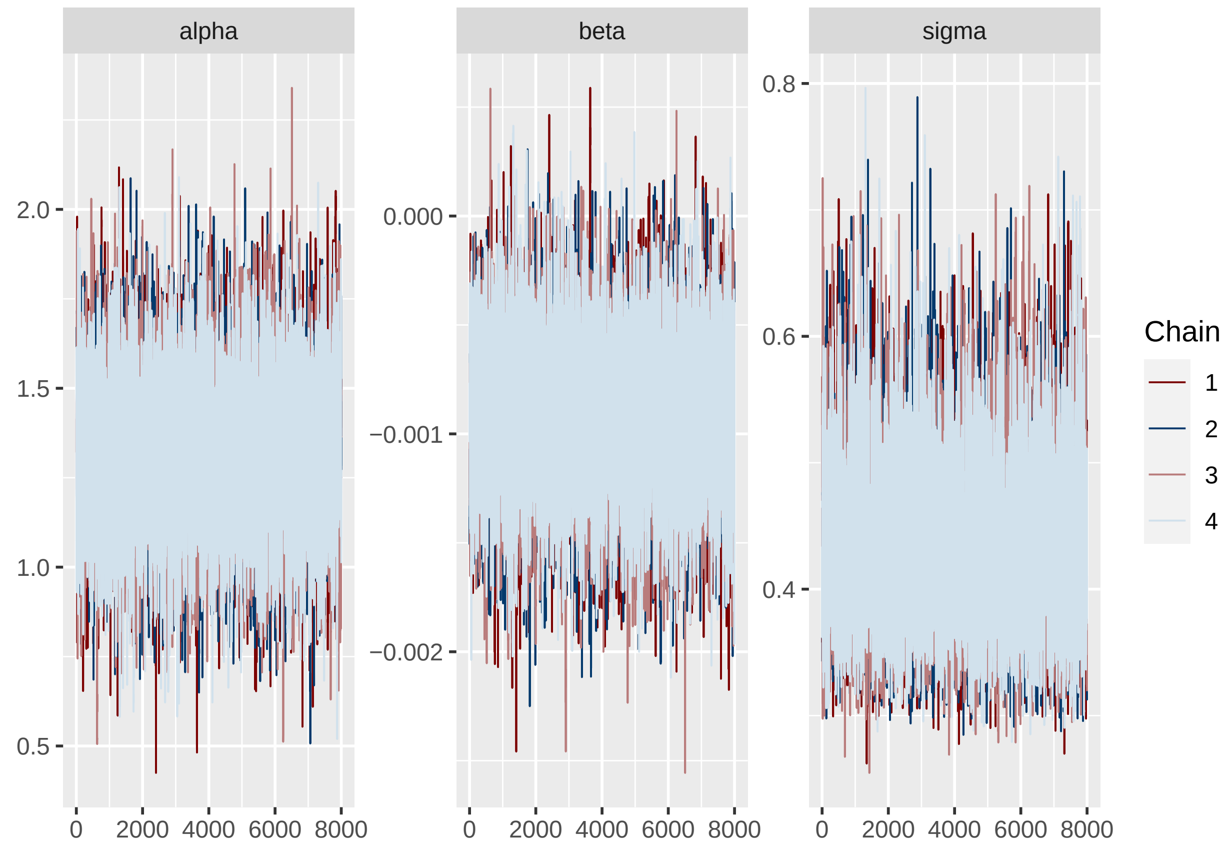
```
n <- NROW(dat_task2)
data.in2a <- list(x=dat_task2$spawners, y=dat_task2$recruits, n=n)
model.fit2a <- sampling(task_2a, data=data.in2a, iter=10000, warmup = 2000)
```

```
check_hmc_diagnostics(model.fit2a)
```

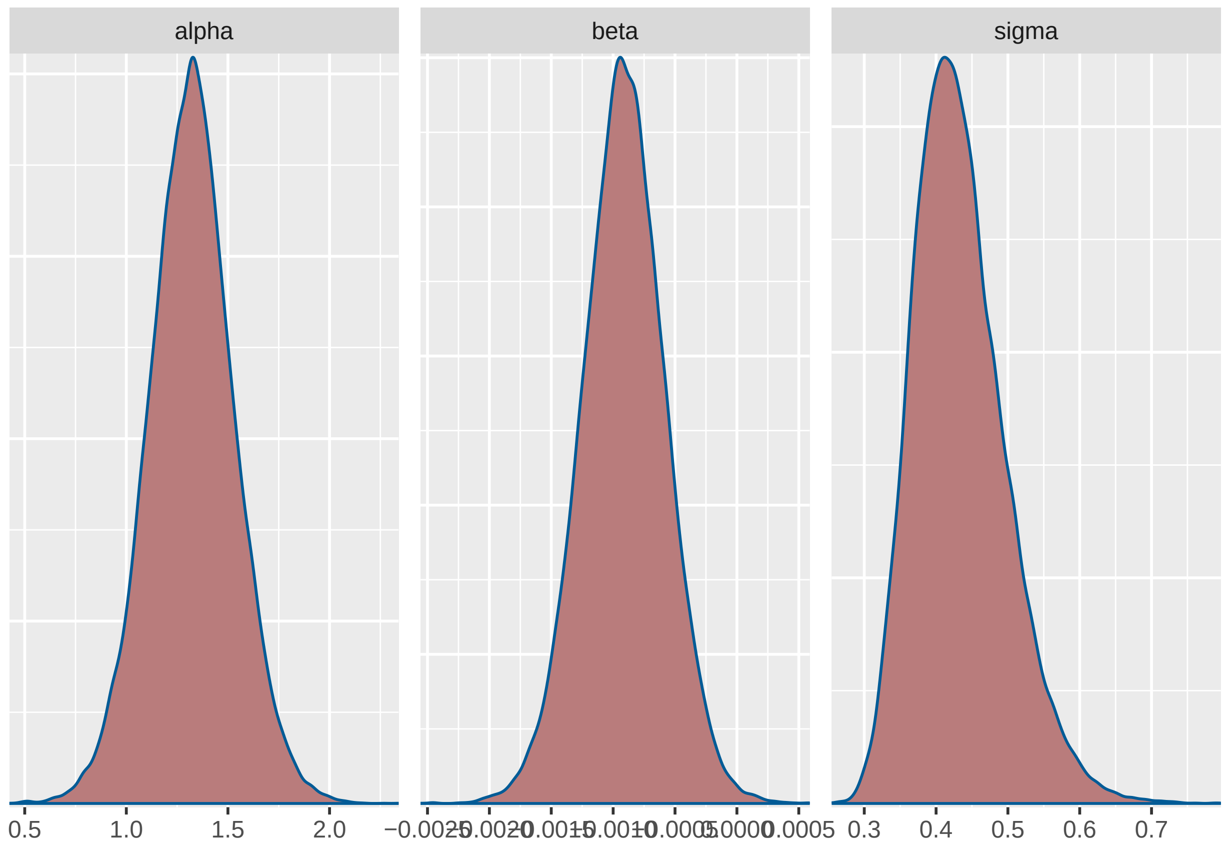
```
##
## Divergences:

## 0 of 32000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 32000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

```
posterior_2a <- as.array(model.fit2a)
```

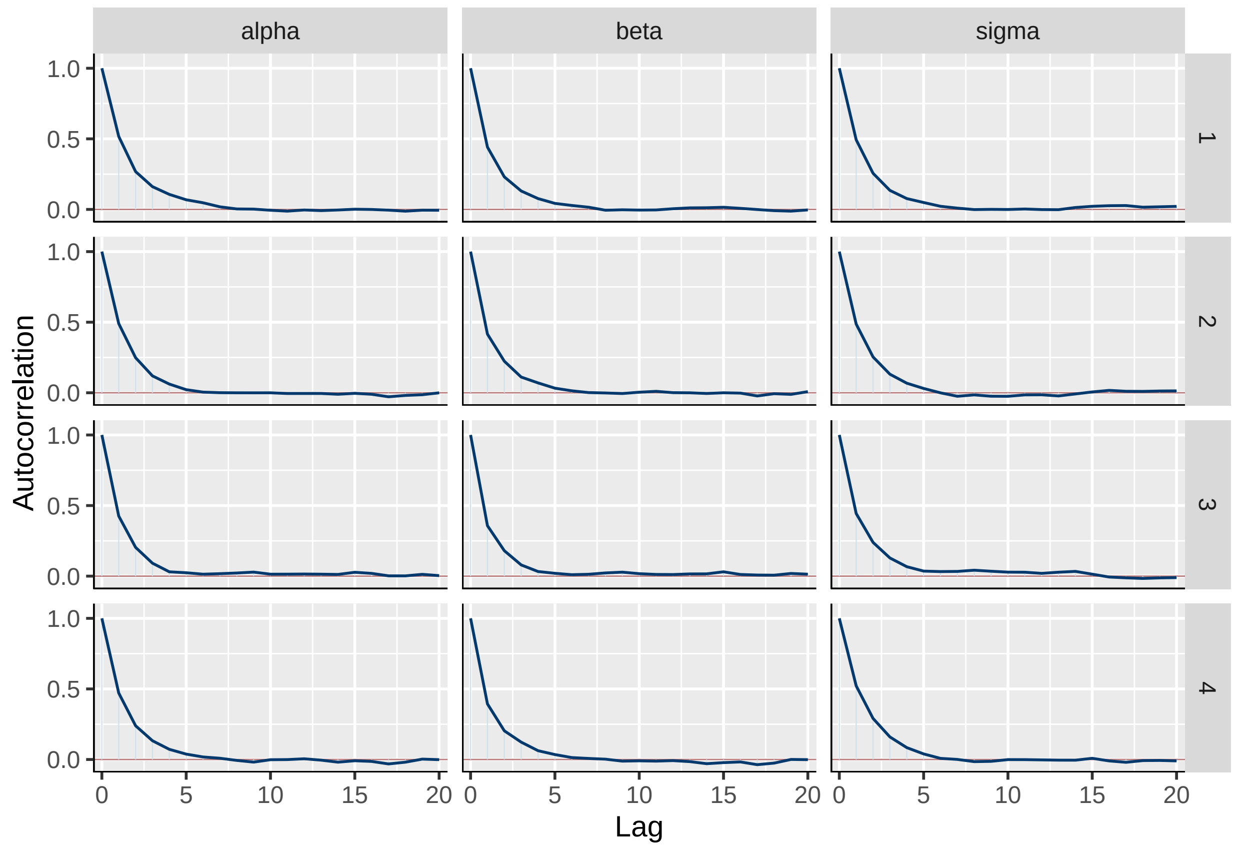
```
color_scheme_set("mix-blue-red")
mcmc_trace(posterior_2a, pars = c("alpha", "beta", "sigma"))
```



```r
mcmc_dens(posterior_2a, pars = c("alpha", "beta", "sigma"))
```



```r
mcmc_acf(posterior_2a, pars = c("alpha", "beta", "sigma"))
```



```r
print(model.fit2a, pars = c("alpha", "beta", "sigma"), digits=8)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=2000; thin=1;
## post-warmup draws per chain=8000, total post-warmup draws=32000.
##
##                mean    se_mean         sd        2.5%        25%         50%
## alpha   1.32170098 0.00201365 0.20856136  0.91155952  1.1860971  1.32100683
## beta   -0.00091416 0.00000308 0.00033383 -0.00157598 -0.0011312 -0.00091442
## sigma   0.43321218 0.00061930 0.06334119  0.32903972  0.3879801  0.42636611
##                 75%       97.5% n_eff     Rhat
## alpha   1.45543234  1.73988377 10728 1.000414
## beta   -0.00069506 -0.00025718 11760 1.000294
## sigma   0.47138218  0.57538081 10461 1.000250
##
## Samples were drawn using NUTS(diag_e) at Sun Oct 13 13:09:11 2024.
## 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).
```

```r
median(extract(model.fit2a)$alpha)
```

```
## [1] 1.321007
```

```
median(extract(model.fit2a)$beta)
```

```
## [1] -0.0009144234
```

```
median(extract(model.fit2a)$sigma)
```

```
## [1] 0.4263661
```

Prior Choices:

Alpha : Normal distribution with mean of 0 and variance 1000^2 Beta : Normal distribution with mean of 0 and variance 1000^2 Sigma : I have transformed the standard deviation into a precious parameter Tao, and used normal gamma prior alpha = 0.001, beta = 0.001 on the precision parameter Tao. Then transformed back to SD. sigma = sqrt(1/tau);

With 4 chains of 10000 Iterations (2000 warm-ups) each, my Bayesian estimate for parameters are:

Estimated Posterior Mean : Alpha: 1.32 Beta: -0.00091 Sigma: 0.43

Estimated Posterior Median : Alpha: 1.32 Beta: -0.00091 Sigma: 0.42

50% Credible Interval : Alpha: 1.18, 1.46 Beta: -0.001,-0.0006 Sigma: 0.38,0.47

```
data_to_plot <- data.frame(x = dat_task2$spawners, log_y_x = log(dat_task2$recruits/dat_task2$spawners)
fit2b <- extract(model.fit2a,  c("alpha", "beta", "sigma"))
fit2b_mean <- c(mean(fit2b$alpha),mean(fit2b$beta),mean(fit2b$sigma))
names(fit2b_mean) <- c("alpha_mean", "beta_mean" , "sigma_mean" )
fit2b_median <- c(median(fit2b$alpha),median(fit2b$beta),median(fit2b$sigma))
names(fit2b_median) <- c("alpha_median", "beta_median" , "sigma_median" )
```

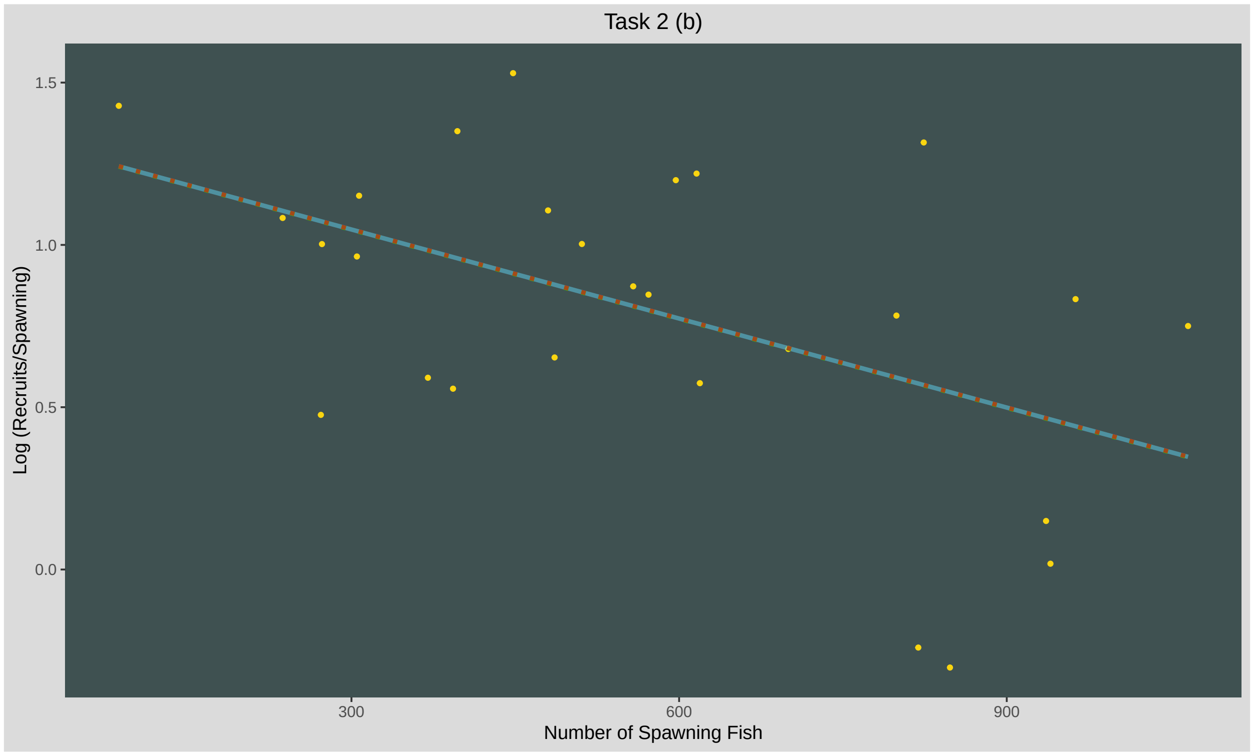
```
xgr <- with(dat_task2, seq(from = 87, to = 1066, length = 601))
line_to_plot_mean <- data.frame(x = xgr, y = fit2b_mean["alpha_mean"]+ fit2b_mean["beta_mean"]*xgr)
line_to_plot_median <- data.frame(x = xgr, y = fit2b_median["alpha_median"]+ fit2b_median["beta_median"
posterior_mean_given_data <- function(xgr) {
  sapply(xgr, function(x) {
    mean(fit2b$alpha + x*fit2b$beta)
  })
}
line_to_plot_posterior_mean_given_data <- data.frame(x = xgr, y = posterior_mean_given_data(xgr))
```

Just an extra step to ensure above calculation of Mu is correct by using original data, the returned the result should be the same with Stanfit Mu[i] estimates

```
posterior_mean_given_data(data_to_plot$x)
```

```
##  [1] 0.4413650 0.7988015 1.0428822 1.0730495 0.5684332 0.4623907 0.8774193
##  [8] 1.0410539 0.3472065 0.8829042 0.9624361 1.1050451 0.6817890 0.8545653
## [15] 1.2421691 0.9834618 0.9121573 0.5730040 0.5912872 1.0721353 0.4660473
## [22] 0.8115997 0.7759475 0.5464934 0.7558360 0.9587795 0.7585785
```

```
#data_to_plot
ggplot() +
  geom_point(data = data_to_plot, aes(x = x, y= log_y_x), color = "#FAD510", size =1) +
  geom_line(data = line_to_plot_mean, aes(x=x, y =y ), color = "#5BBCD6", size = 1.2, alpha= 0.6) +
  geom_line(data = line_to_plot_median, aes(x=x, y =y ), color = "green", size = 1.2, alpha= 0.8,linetyp
  geom_line(data = line_to_plot_posterior_mean_given_data, aes(x=x, y =y ), color = "#CB2314", size = 1
  ggtitle("Task 2 (b)") +
  labs(x = "Number of Spawning Fish", y = "Log (Recruits/Spawning)") +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "#3F5151"),
        plot.background = element_rect(fill = "gray86"))
```



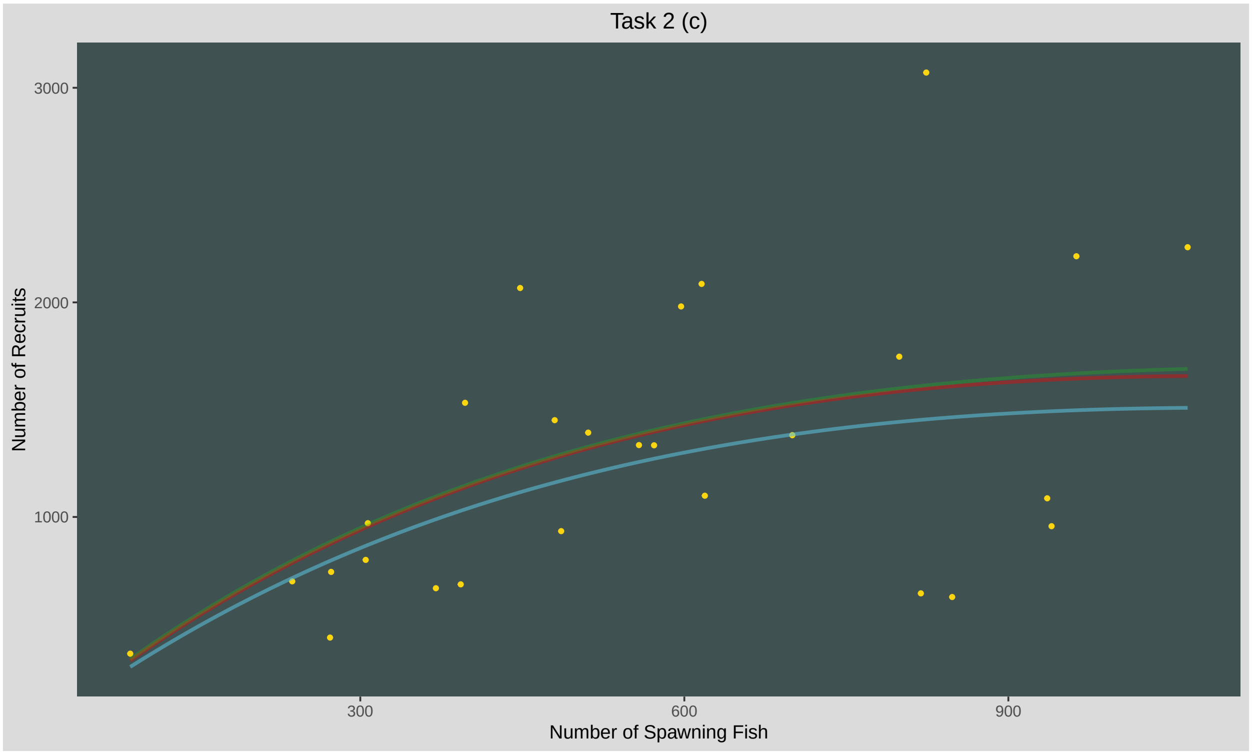
Overall the model seems dealt with some heteroscedasticity issues but still not ideally. There are still increasing variability along with increasing in number of Spawning fish, particularly around 800 Range, the data points year 1958 and 1964 were the only two years, where the number of recruits were less than number of Spawning. which directly lead to the log result in negative range. (1958: 819 vs 644; 1964: 848 vs 627), and in the year of 1944, the measurement was 824 vs 3071.

For this particular task, by using either mean or median estimate of alpha / beta returns very similar result. But if the posterior distribution of Alpha or Beta are clearly not symmetric, Median may be better option to depicting the response, in order to reduce effect of outliers if there were any.

```
sigma_mean <- mean(extract(model.fit2a)$sigma)
posterior_mean_given_data_2 <- function(xgr) {sapply(xgr, function(x) {mean(exp(fit2b$alpha + x*fit2b$b
  })
}
data_to_plot_c <- data.frame(x = dat_task2$spawners, y = dat_task2$recruits)
line_to_plot_c1 <- data.frame(x = xgr, y = xgr * exp(posterior_mean_given_data(xgr)))
```

```
line_to_plot_c2 <- data.frame(x = xgr, y = xgr * exp(posterior_mean_given_data(xgr) + 1/2 * (sigma_mean
line_to_plot_c3 <- data.frame(x = xgr, y = xgr * posterior_mean_given_data_2(xgr))
```

```
#data_to_plot
ggplot() +
  geom_point(data = data_to_plot_c, aes(x = x, y= y), color = "#FAD510", size =1) +
  geom_line(data = line_to_plot_c1, aes(x=x, y =y ), color = "#5BBCD6", size = 1, alpha= 0.6) +
  geom_line(data = line_to_plot_c2, aes(x=x, y =y ), color = "red", size = 1, alpha= 0.4) +
  geom_line(data = line_to_plot_c3, aes(x=x, y =y ), color = "green", size = 1, alpha= 0.2) +
  ggtitle("Task 2 (c)") +
  labs(x = "Number of Spawning Fish", y = "Number of Recruits") +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "#3F5151"),
        plot.background = element_rect(fill = "gray86"))
```



On the original scale of the data , these lines estimate the number of recruits (in thousands) based on number of Spawning Fish (in thousands) :

The blue line is making estimate based on the posterior mean of m(x;alpha, beta) given the data, from the log(y/x) ~ x linear model. The red line is adding additional uncertainty 'directly' onto the previous estimate (blue line), by incorporating posterior mean of sigma, essentially adding variance directly to the blue line estimate to better deal with heterosexuality. The green line may be the most suitable for our data, by taking into the account that the increasing variance seems to be proportional to the number of Spawning Fish, therefore the estimates are considering the effects of variability first, then calculate the mean for the response : number of Recruits.