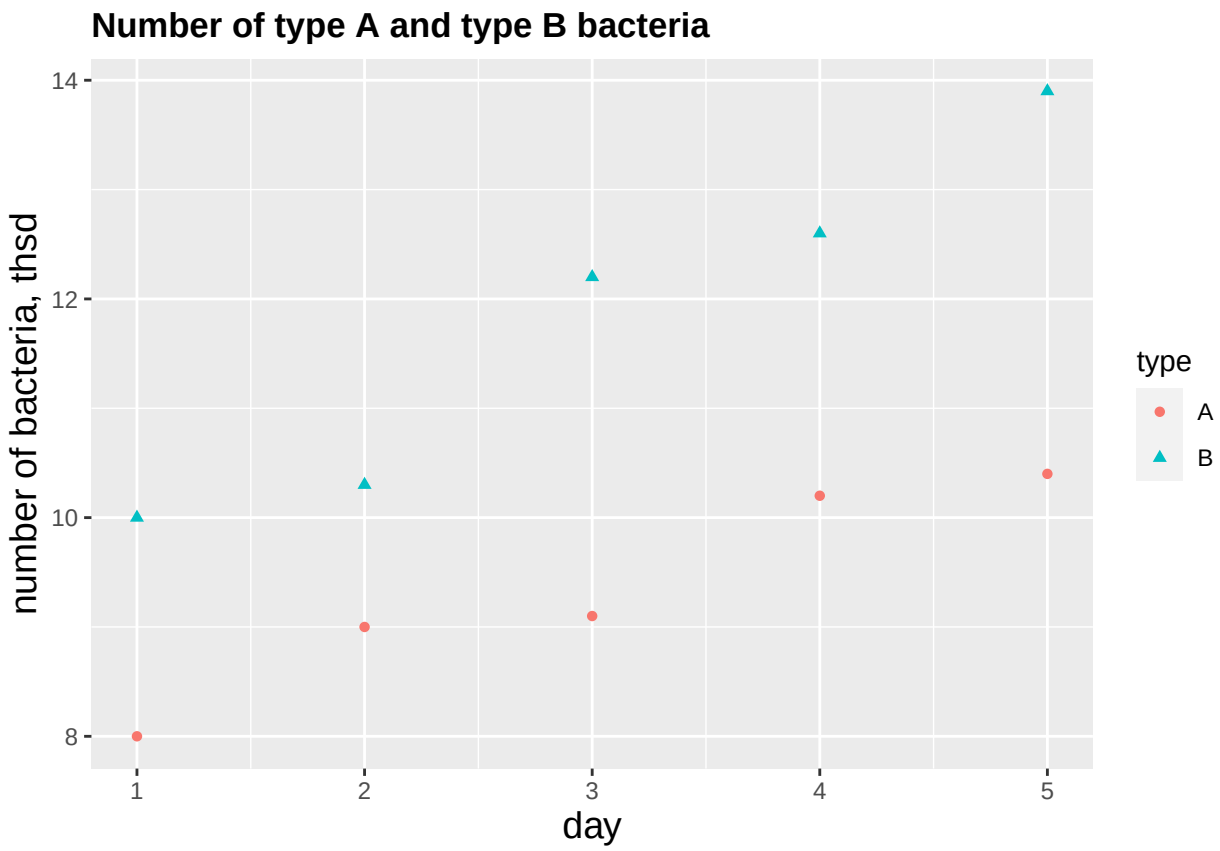
HW8_R_m

```
ggplot(data=bacteria, aes(x = day, y = number, colour = type, shape = type)) +
  geom_point() +
  labs(title = "Number of type A and type B bacteria",
       x = "day",
       y = "number of bacteria, thsd") +
  theme(plot.title = element_text(lineheight=.8, face="bold"),
          axis.title.x = element_text(size = 14),
          axis.title.y = element_text(size = 14))
```




```
####  Model 1: A one-way ANOVA using type = "A" as the reference category.

# Create a dummy variable
type_b = c(type == 'B')

# Create model
stanmod = "
data {
  int<lower=1> n; // number of observations
  vector[n] y; // data
  vector[n] type_b; //indicator for type b
```

```
  real<lower=0> v_sample; // sample variance of y
  real mu_sample; // sample mean of y

}

parameters {
  real mu_0;
  real alpha_b;
  real<lower=0> sigmasq;

}

transformed parameters {
  vector[n] mu;        // mean of observations
  mu = mu_0 + alpha_b*type_b;

}
model {

  // priors
  mu_0 ~ normal(mu_sample, 10 * sqrt(v_sample));
  alpha_b ~ normal(0.0, 100);
  sigmasq ~ inv_gamma(0.01, 0.01);

  // data distribution
  for(i in 1:n){
    y[i] ~ normal(mu[i], sqrt(sigmasq));
  }
}
generated quantities {
  real Rbsq;          // goodness-of-fit
  vector[n] log_lik;  // log likelihood of data
  Rbsq = 1 - sigmasq/v_sample;
  for (i in 1:n) log_lik[i] = normal_lpdf(y[i]|mu[i], sqrt(sigmasq));
}
"

stan_dat = list(n = length(number), y = number,
                type_b = type_b, v_sample = var(number), mu_sample = mean(number))

stan_fit = stan(model_code = stanmod, data = stan_dat, iter = 10^4)
```

```
##
## SAMPLING FOR MODEL '0287d11e8883d8736f3c1db3e8612913' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.29 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
```

```
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.084772 seconds (Warm-up)
## Chain 1:                0.097921 seconds (Sampling)
## Chain 1:                0.182693 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '0287d11e8883d8736f3c1db3e8612913' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.084844 seconds (Warm-up)
## Chain 2:                0.135715 seconds (Sampling)
## Chain 2:                0.220559 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '0287d11e8883d8736f3c1db3e8612913' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 3: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 3: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 3: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 3: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
```

```
## Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.151998 seconds (Warm-up)
## Chain 3:                0.153484 seconds (Sampling)
## Chain 3:                0.305482 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '0287d11e8883d8736f3c1db3e8612913' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 4e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.08946 seconds (Warm-up)
## Chain 4:                0.105642 seconds (Sampling)
## Chain 4:                0.195102 seconds (Total)
## Chain 4:
```

```r
#sso <- launch_shinystan(stan_fit)

# Mean, 0.025 quantile, and 0.975 quantile
summary(stan_fit)$summary[,c("mean", "2.5%", "97.5%")]
```

```
##                  mean        2.5%      97.5%
## mu_0         9.3537663   7.9847999 10.7733394
## alpha_b      2.4503629   0.5113444  4.4296609
## sigmasq      2.4431358   0.8256458  6.6802914
## mu[1]        9.3537663   7.9847999 10.7733394
## mu[2]        9.3537663   7.9847999 10.7733394
## mu[3]        9.3537663   7.9847999 10.7733394
## mu[4]        9.3537663   7.9847999 10.7733394
## mu[5]        9.3537663   7.9847999 10.7733394
## mu[6]       11.8041291  10.3986995 13.1969861
## mu[7]       11.8041291  10.3986995 13.1969861
## mu[8]       11.8041291  10.3986995 13.1969861
## mu[9]       11.8041291  10.3986995 13.1969861
## mu[10]      11.8041291  10.3986995 13.1969861
## Rbsq         0.2586824  -1.0269924  0.7494753
```

```
## log_lik[1]  -1.8861131  -3.0764890 -1.1880556
## log_lik[2]  -1.4218541  -2.1240366 -0.8986054
## log_lik[3]  -1.4055198  -2.0940038 -0.8905027
## log_lik[4]  -1.5869420  -2.4477667 -1.0193295
## log_lik[5]  -1.6910535  -2.6577844 -1.0823171
## log_lik[6]  -2.2782965  -3.9202123 -1.3907109
## log_lik[7]  -2.0064038  -3.3271232 -1.2640695
## log_lik[8]  -1.4278976  -2.1342194 -0.9000437
## log_lik[9]  -1.5577822  -2.3704013 -0.9918690
## log_lik[10] -2.5844747  -4.5077763 -1.5202002
## lp__        -8.6731331 -12.4080519 -7.0113908
```

```
summary(stan_fit)$summary
```

```
##                  mean     se_mean        sd        2.5%        25%        50%
## mu_0         9.3537663 0.008315195 0.7124428   7.9847999  8.9161244  9.3446405
## alpha_b      2.4503629 0.011614262 0.9979804   0.5113444  1.8449374  2.4515091
## sigmasq      2.4431358 0.021015324 1.7449936   0.8256458  1.4324855  2.0040976
## mu[1]        9.3537663 0.008315195 0.7124428   7.9847999  8.9161244  9.3446405
## mu[2]        9.3537663 0.008315195 0.7124428   7.9847999  8.9161244  9.3446405
## mu[3]        9.3537663 0.008315195 0.7124428   7.9847999  8.9161244  9.3446405
## mu[4]        9.3537663 0.008315195 0.7124428   7.9847999  8.9161244  9.3446405
## mu[5]        9.3537663 0.008315195 0.7124428   7.9847999  8.9161244  9.3446405
## mu[6]       11.8041291 0.005530213 0.6993863  10.3986995 11.3753952 11.8067226
## mu[7]       11.8041291 0.005530213 0.6993863  10.3986995 11.3753952 11.8067226
## mu[8]       11.8041291 0.005530213 0.6993863  10.3986995 11.3753952 11.8067226
## mu[9]       11.8041291 0.005530213 0.6993863  10.3986995 11.3753952 11.8067226
## mu[10]      11.8041291 0.005530213 0.6993863  10.3986995 11.3753952 11.8067226
## Rbsq         0.2586824 0.006376653 0.5294812  -1.0269924  0.1234832  0.3918992
## log_lik[1]  -1.8861131 0.004778295 0.4824140  -3.0764890 -2.1310556 -1.7992577
## log_lik[2]  -1.4218541 0.003940954 0.3152575  -2.1240366 -1.6050882 -1.3893200
## log_lik[3]  -1.4055198 0.003929771 0.3106740  -2.0940038 -1.5857720 -1.3738064
## log_lik[4]  -1.5869420 0.004130271 0.3669738  -2.4477667 -1.7840048 -1.5372787
## log_lik[5]  -1.6910535 0.004334879 0.4061260  -2.6577844 -1.9006624 -1.6311969
## log_lik[6]  -2.2782965 0.004972322 0.6556241  -3.9202123 -2.5977432 -2.1450721
## log_lik[7]  -2.0064038 0.004018904 0.5311447  -3.3271232 -2.2696796 -1.9074162
## log_lik[8]  -1.4278976 0.003738536 0.3139515  -2.1342194 -1.6132889 -1.3964290
## log_lik[9]  -1.5577822 0.003484830 0.3511661  -2.3704013 -1.7549018 -1.5137079
## log_lik[10] -2.5844747 0.005874686 0.7785053  -4.5077763 -2.9916701 -2.4284548
## lp__        -8.6731331 0.020969563 1.4750683 -12.4080519 -9.3301096 -8.3005026
##                    75%      97.5%     n_eff      Rhat
## mu_0         9.7869395 10.7733394  7340.998 1.0004273
## alpha_b      3.0500572  4.4296609  7383.475 1.0001416
## sigmasq      2.8887071  6.6802914  6894.702 1.0004525
## mu[1]        9.7869395 10.7733394  7340.998 1.0004273
## mu[2]        9.7869395 10.7733394  7340.998 1.0004273
## mu[3]        9.7869395 10.7733394  7340.998 1.0004273
## mu[4]        9.7869395 10.7733394  7340.998 1.0004273
## mu[5]        9.7869395 10.7733394  7340.998 1.0004273
## mu[6]       12.2314477 13.1969861 15993.758 0.9999804
## mu[7]       12.2314477 13.1969861 15993.758 0.9999804
## mu[8]       12.2314477 13.1969861 15993.758 0.9999804
## mu[9]       12.2314477 13.1969861 15993.758 0.9999804
## mu[10]      12.2314477 13.1969861 15993.758 0.9999804
## Rbsq         0.5653427  0.7494753  6894.702 1.0004525
```

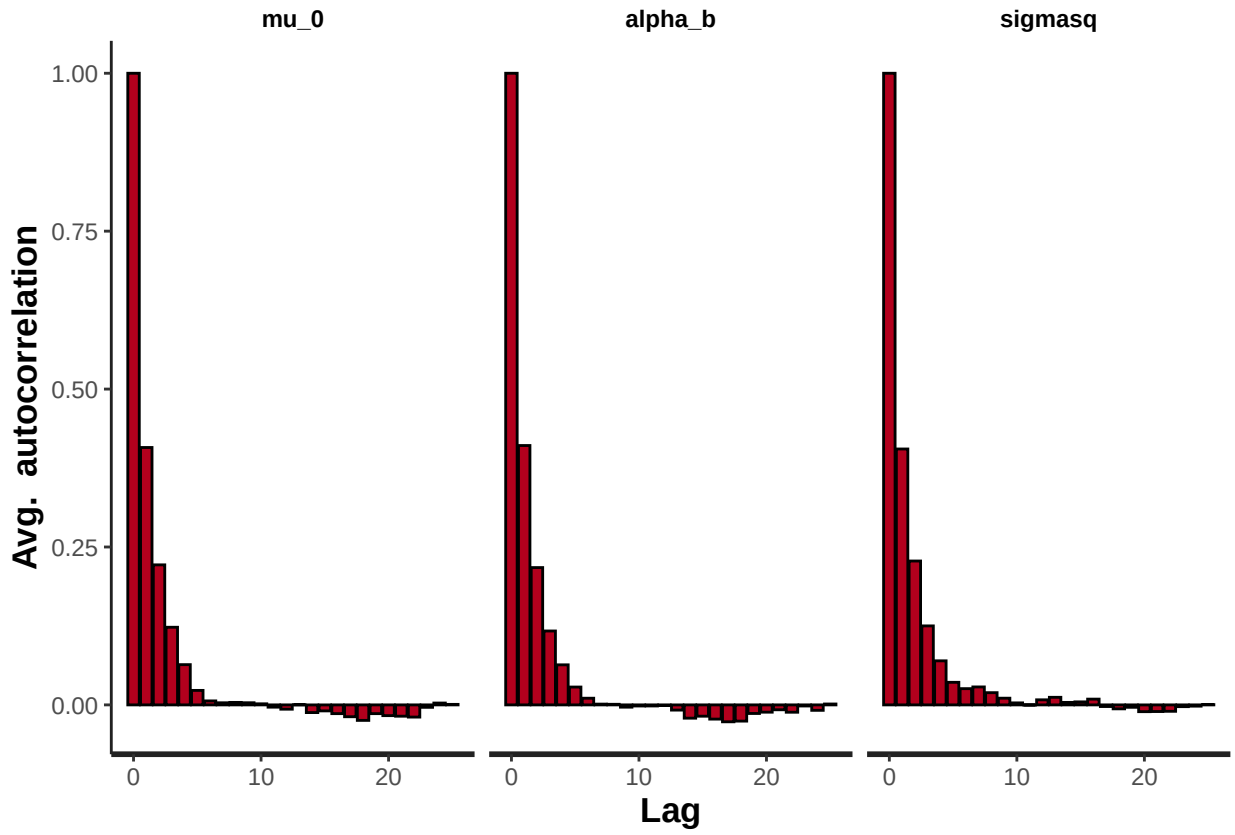
```
## log_lik[1]  -1.5494504 -1.1880556 10192.809 1.0000401
## log_lik[2]  -1.2023276 -0.8986054  6399.236 1.0005286
## log_lik[3]  -1.1917434 -0.8905027  6249.932 1.0005324
## log_lik[4]  -1.3343144 -1.0193295  7894.291 1.0001383
## log_lik[5]  -1.4103519 -1.0823171  8777.435 1.0001630
## log_lik[6]  -1.8096796 -1.3907109 17385.665 0.9998955
## log_lik[7]  -1.6351676 -1.2640695 17466.680 0.9998870
## log_lik[8]  -1.2092930 -0.9000437  7052.157 1.0007108
## log_lik[9]  -1.3122206 -0.9918690 10154.581 1.0003359
## log_lik[10] -2.0143089 -1.5202002 17561.187 1.0001483
## lp__        -7.6103898 -7.0113908  4948.180 1.0003612
```

```
# Gelman-Rubin statistic
summary(stan_fit)$summary[,"Rhat"]
```

```
##        mu_0     alpha_b     sigmasq       mu[1]       mu[2]       mu[3]
##   1.0004273   1.0001416   1.0004525   1.0004273   1.0004273   1.0004273
##       mu[4]       mu[5]       mu[6]       mu[7]       mu[8]       mu[9]
##   1.0004273   1.0004273   0.9999804   0.9999804   0.9999804   0.9999804
##      mu[10]        Rbsq  log_lik[1]  log_lik[2]  log_lik[3]  log_lik[4]
##   0.9999804   1.0004525   1.0000401   1.0005286   1.0005324   1.0001383
##  log_lik[5]  log_lik[6]  log_lik[7]  log_lik[8]  log_lik[9] log_lik[10]
##   1.0001630   0.9998955   0.9998870   1.0007108   1.0003359   1.0001483
##        lp__
##   1.0003612
```

```
# check convergence with trace plots
#stan_trace(stan_fit, c("mu_0", "alpha_b", "sigmasq"))

# Check the ACF of draws
stan_ac(stan_fit, c("mu_0", "alpha_b", "sigmasq"))
```




```
# WAIC and LOOC
log_lik = extract_log_lik(stan_fit, merge_chains = FALSE)
r_eff = exp(relative_eff(log_lik))

waic(log_lik)
```

```
## Warning:
## 2 (20.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 20000 by 10 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic    -19.2 1.6
## p_waic         2.3 0.5
## waic          38.3 3.2
##
## 2 (20.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
loo(log_lik, r_eff = r_eff)
```

```
##
## Computed from 20000 by 10 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -19.3 1.7
## p_loo         2.4 0.6
## looic        38.6 3.3
## ------
```

```
## 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.
```

```
####  Model 2: A simple linear regression model using day as the predictor.

# Create model
stanmod = "
data {
  int<lower=1> n; // number of observations
  vector[n] y; // data
  vector[n] day; //day
  real<lower=0> v_sample; // sample variance of y
  real mu_sample; // sample mean of y

}

parameters {
  real beta_0;
  real beta_1;
  real<lower=0> sigmasq;

}

transformed parameters {
  vector[n] mu;        // mean of observations
  mu = beta_0 + beta_1*day;

}
model {

  // priors
  beta_0 ~ normal(0, 100);
  beta_1 ~ normal(0 , 100);
  sigmasq ~ inv_gamma(0.01, 0.01);

  // data distribution
  for(i in 1:n){
    y[i] ~ normal(mu[i], sqrt(sigmasq));
  }
}
generated quantities {
  real Rbsq;          // goodness-of-fit
  vector[n] log_lik;  // log likelihood of data
  Rbsq = 1 - sigmasq/v_sample;
  for (i in 1:n) log_lik[i] = normal_lpdf(y[i]|mu[i], sqrt(sigmasq));
}
"

stan_dat = list(n = length(number), y = number,
                day = bacteria$day, v_sample = var(number), mu_sample = mean(number))

stan_fit = stan(model_code = stanmod, data = stan_dat, iter = 10^5)
```

```
##
## SAMPLING FOR MODEL '4096455c0fdc06fd8c683c6257402a0a' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 1: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 1: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 1: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 1: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 1: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 1: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 1: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 1: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 1: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 1: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 1: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.09239 seconds (Warm-up)
## Chain 1:                1.39013 seconds (Sampling)
## Chain 1:                2.48252 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '4096455c0fdc06fd8c683c6257402a0a' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 2: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 2: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 2: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 2: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 2: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 2: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 2: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 2: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 2: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 2: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 2: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.5766 seconds (Warm-up)
## Chain 2:                1.45506 seconds (Sampling)
## Chain 2:                3.03166 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '4096455c0fdc06fd8c683c6257402a0a' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
```

```
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 3: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 3: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 3: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 3: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 3: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 3: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 3: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 3: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 3: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 3: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 3: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.14015 seconds (Warm-up)
## Chain 3:                1.40372 seconds (Sampling)
## Chain 3:                2.54387 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4096455c0fdc06fd8c683c6257402a0a' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 4: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 4: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 4: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 4: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 4: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 4: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 4: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 4: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 4: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 4: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 4: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.10033 seconds (Warm-up)
## Chain 4:                1.5419 seconds (Sampling)
## Chain 4:                2.64223 seconds (Total)
## Chain 4:
```

```
#sso <- launch_shinystan(stan_fit)

# Mean, 0.025 quantile, and 0.975 quantile
pl_cpi = summary(stan_fit)$summary[,c("mean", "2.5%", "97.5%")]
print(pl_cpi, digits = 2)
```

```
##              mean    2.5% 97.5%
## beta_0       8.15   5.695 10.60
```

```
## beta_1       0.81   0.066  1.55
## sigmasq      2.77   0.947  7.58
## mu[1]        8.96   7.140 10.77
## mu[2]        9.77   8.480 11.05
## mu[3]       10.57   9.521 11.62
## mu[4]       11.38  10.094 12.66
## mu[5]       12.18  10.364 14.01
## mu[6]        8.96   7.140 10.77
## mu[7]        9.77   8.480 11.05
## mu[8]       10.57   9.521 11.62
## mu[9]       11.38  10.094 12.66
## mu[10]      12.18  10.364 14.01
## Rbsq         0.16  -1.299  0.71
## log_lik[1]  -1.72  -2.840 -1.08
## log_lik[2]  -1.57  -2.253 -1.06
## log_lik[3]  -1.92  -2.713 -1.38
## log_lik[4]  -1.76  -2.585 -1.19
## log_lik[5]  -2.26  -4.099 -1.32
## log_lik[6]  -1.76  -2.948 -1.10
## log_lik[7]  -1.49  -2.147 -1.00
## log_lik[8]  -2.04  -2.947 -1.45
## log_lik[9]  -1.78  -2.640 -1.21
## log_lik[10] -2.21  -3.983 -1.30
## lp__        -9.34 -13.076 -7.70
```

```
summary(stan_fit)$summary
```

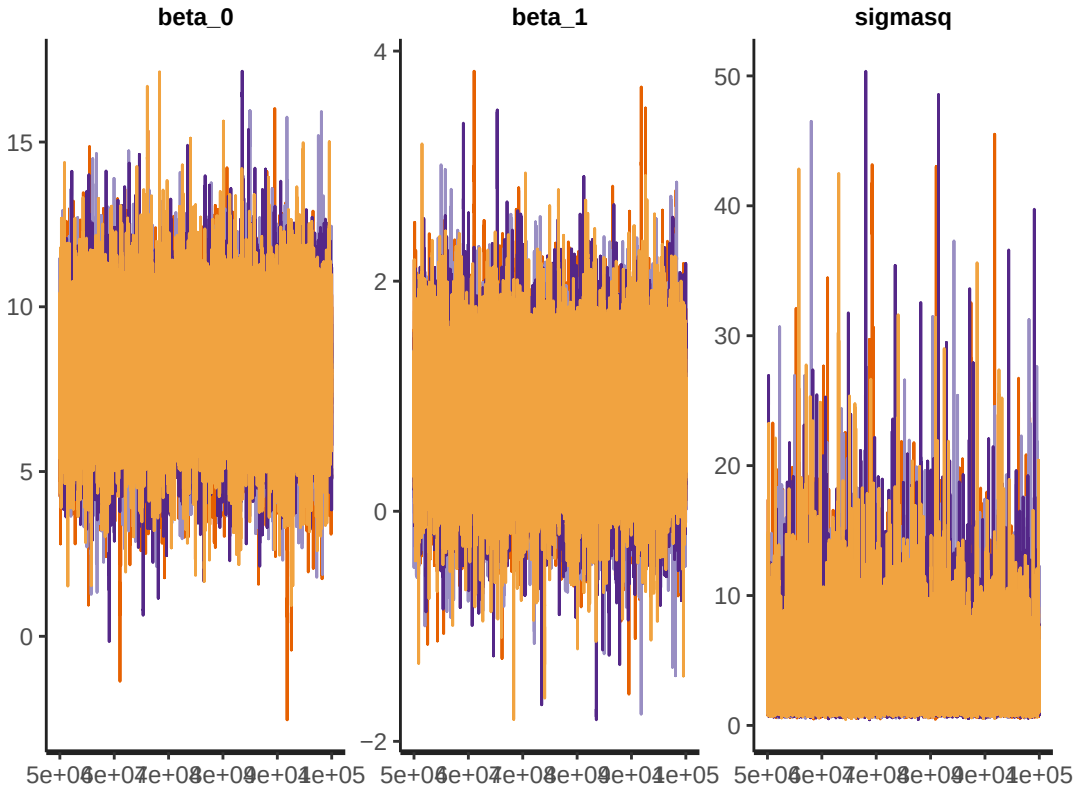
```
##                  mean      se_mean        sd         2.5%          25%
## beta_0       8.1535493 0.0051006377 1.2319106   5.69487845  7.402185225
## beta_1       0.8059512 0.0015358180 0.3717824   0.06649899  0.577899878
## sigmasq      2.7687380 0.0079842217 1.9136748   0.94682921  1.625783737
## mu[1]        8.9595005 0.0036161639 0.9092609   7.13971061  8.405488549
## mu[2]        9.7654517 0.0022029832 0.6421460   8.47958030  9.373373170
## mu[3]       10.5714030 0.0011600703 0.5238242   9.52083327 10.251765307
## mu[4]       11.3773542 0.0016028402 0.6425547  10.09402865 10.984570946
## mu[5]       12.1833054 0.0029070508 0.9098382  10.36438946 11.627041732
## mu[6]        8.9595005 0.0036161639 0.9092609   7.13971061  8.405488549
## mu[7]        9.7654517 0.0022029832 0.6421460   8.47958030  9.373373170
## mu[8]       10.5714030 0.0011600703 0.5238242   9.52083327 10.251765307
## mu[9]       11.3773542 0.0016028402 0.6425547  10.09402865 10.984570946
## mu[10]      12.1833054 0.0029070508 0.9098382  10.36438946 11.627041732
## Rbsq         0.1598853 0.0024226424 0.5806639  -1.29939249  0.002163375
## log_lik[1]  -1.7204854 0.0016999611 0.4513468  -2.83991507 -1.937500978
## log_lik[2]  -1.5651596 0.0011386990 0.3061229  -2.25251300 -1.745880071
## log_lik[3]  -1.9199495 0.0008299200 0.3417446  -2.71282187 -2.108052694
## log_lik[4]  -1.7571224 0.0009919773 0.3549763  -2.58491095 -1.954289961
## log_lik[5]  -2.2625291 0.0020862077 0.7258750  -4.09933195 -2.610829265
## log_lik[6]  -1.7589434 0.0017629995 0.4720028  -2.94806643 -1.982244883
## log_lik[7]  -1.4929758 0.0011702205 0.2941767  -2.14676298 -1.670141455
## log_lik[8]  -2.0383728 0.0009168273 0.3808982  -2.94728467 -2.241213005
## log_lik[9]  -1.7849236 0.0009936302 0.3627878  -2.64005334 -1.984100996
## log_lik[10] -2.2099122 0.0019982605 0.6983265  -3.98270041 -2.540998777
## lp__        -9.3378596 0.0065161092 1.4424241 -13.07629141 -9.994234066
##                   50%        75%      97.5%     n_eff      Rhat
## beta_0       8.1568212  8.9071550 10.6009761  58332.35 1.0000463
```

```
## beta_1       0.8050938  1.0325749  1.5482468  58600.08 1.0000227
## sigmasq      2.2649719  3.2885369  7.5780312  57447.49 1.0000445
## mu[1]        8.9614813  9.5140446 10.7740653  63223.84 1.0000514
## mu[2]        9.7656511 10.1567063 11.0503819  84966.00 1.0000512
## mu[3]       10.5705221 10.8903066 11.6246884 203893.11 1.0000218
## mu[4]       11.3741469 11.7687399 12.6628404 160708.83 0.9999930
## mu[5]       12.1795173 12.7356796 14.0058847  97954.21 0.9999933
## mu[6]        8.9614813  9.5140446 10.7740653  63223.84 1.0000514
## mu[7]        9.7656511 10.1567063 11.0503819  84966.00 1.0000512
## mu[8]       10.5705221 10.8903066 11.6246884 203893.11 1.0000218
## mu[9]       11.3741469 11.7687399 12.6628404 160708.83 0.9999930
## mu[10]      12.1795173 12.7356796 14.0058847  97954.21 0.9999933
## Rbsq         0.3127424  0.5066905  0.7127048  57447.49 1.0000445
## log_lik[1]  -1.6408396 -1.4116816 -1.0759096  70492.47 1.0000577
## log_lik[2]  -1.5345834 -1.3498427 -1.0555926  72272.62 1.0000524
## log_lik[3]  -1.8776701 -1.6819739 -1.3771363 169562.94 1.0000124
## log_lik[4]  -1.7117245 -1.5093356 -1.1945966 128054.60 0.9999952
## log_lik[5]  -2.0991659 -1.7491337 -1.3151218 121062.20 0.9999912
## log_lik[6]  -1.6723904 -1.4348555 -1.0963531  71677.83 1.0000491
## log_lik[7]  -1.4657849 -1.2854992 -0.9970162  63194.79 1.0000559
## log_lik[8]  -1.9827047 -1.7732980 -1.4541906 172600.80 1.0000041
## log_lik[9]  -1.7371994 -1.5313845 -1.2126996 133307.85 1.0000078
## log_lik[10] -2.0540944 -1.7175549 -1.2982023 122127.32 1.0000103
## lp__        -8.9618218 -8.2855117 -7.7038194  49001.48 1.0000654
```

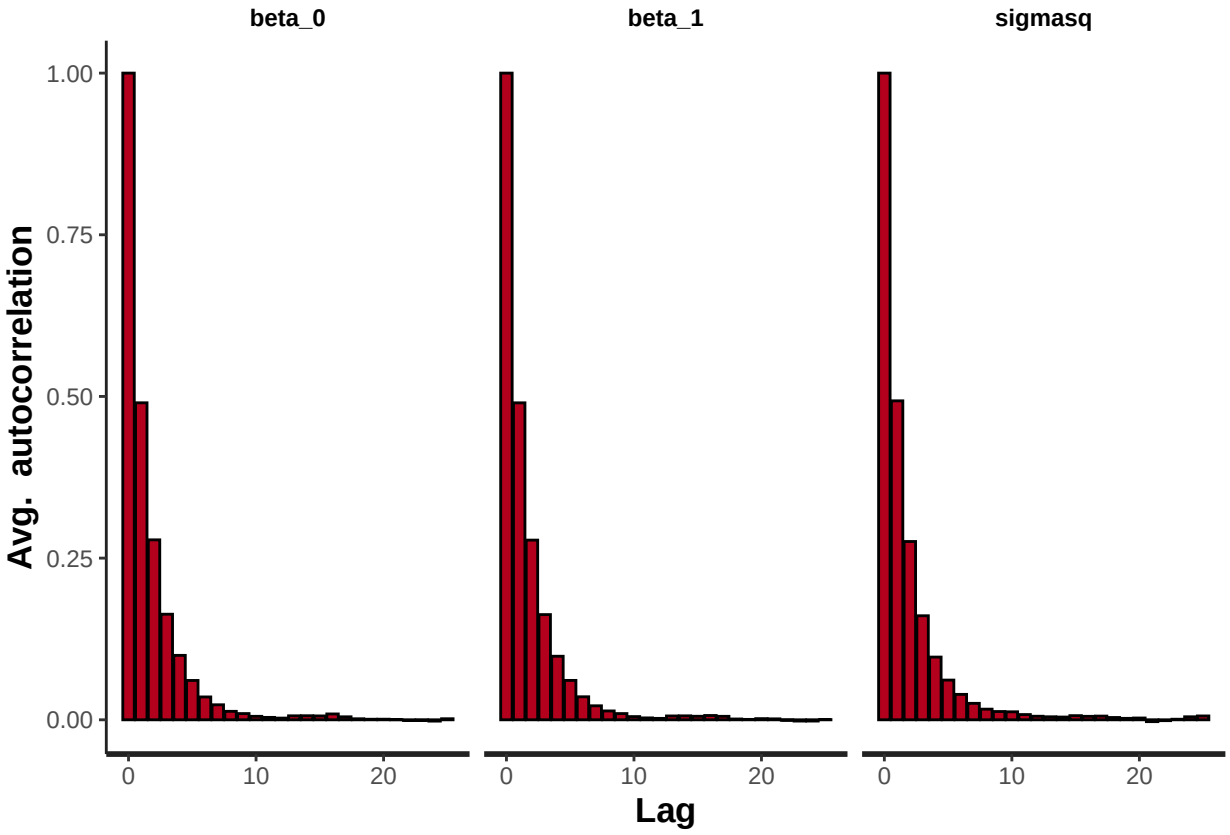
```
# Gelman-Rubin statistic
summary(stan_fit)$summary[,"Rhat"]
```

```
##      beta_0      beta_1     sigmasq       mu[1]       mu[2]       mu[3]
##   1.0000463   1.0000227   1.0000445   1.0000514   1.0000512   1.0000218
##       mu[4]       mu[5]       mu[6]       mu[7]       mu[8]       mu[9]
##   0.9999930   0.9999933   1.0000514   1.0000512   1.0000218   0.9999930
##      mu[10]        Rbsq  log_lik[1]  log_lik[2]  log_lik[3]  log_lik[4]
##   0.9999933   1.0000445   1.0000577   1.0000524   1.0000124   0.9999952
##  log_lik[5]  log_lik[6]  log_lik[7]  log_lik[8]  log_lik[9] log_lik[10]
##   0.9999912   1.0000491   1.0000559   1.0000041   1.0000078   1.0000103
##        lp__
##   1.0000654
```

```
# check convergence with trace plots
stan_trace(stan_fit, c("beta_0", "beta_1", "sigmasq"))
```




```
# Check the ACF of draws
stan_ac(stan_fit, c("beta_0", "beta_1", "sigmasq"))
```



```r
# WAIC and LOOC
log_lik = extract_log_lik(stan_fit, merge_chains = FALSE)
r_eff = exp(relative_eff(log_lik))

waic(log_lik)
```

```
## Warning:
## 2 (20.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 200000 by 10 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic    -19.8 1.1
## p_waic         2.1 0.5
## waic          39.5 2.2
##
## 2 (20.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```r
loo(log_lik, r_eff = r_eff)
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
##
## Computed from 200000 by 10 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -20.0 1.2
## p_loo         2.3 0.6
## looic        39.9 2.4
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     8     80.0%   159104
##  (0.5, 0.7]   (ok)       2     20.0%   30950
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
### A parallel lines (main effects) model using day as the quantitative predictor and using type = "A"
type_b = c(type == 'B')

# Create model
stanmod = "
data {
  int<lower=1> n; // number of observations
  vector[n] y; // data
  vector[n] day; //day
  vector[n] type_b; //
  real<lower=0> v_sample; // sample variance of y
  real mu_sample; // sample mean of y
```

```
}

parameters {
  real beta_0;
  real beta_1;
  real alpha_b;
  real<lower=0> sigmasq;

}

transformed parameters {
  vector[n] mu;        // mean of observations
  mu = beta_0 + beta_1 * day + alpha_b * type_b;

}
model {

  // priors
  beta_0 ~ normal(0, 100);
  beta_1 ~ normal(0 , 100);
  alpha_b ~ normal(0, 100);
  sigmasq ~ inv_gamma(0.01, 0.01);

  // data distribution
  for(i in 1:n){
    y[i] ~ normal(mu[i], sqrt(sigmasq));
  }
}
generated quantities {
  real Rbsq;          // goodness-of-fit
  vector[n] log_lik;  // log likelihood of data
  Rbsq = 1 - sigmasq/v_sample;
  for (i in 1:n) log_lik[i] = normal_lpdf(y[i]|mu[i], sqrt(sigmasq));
}
"

stan_dat = list(n = length(number), y = number, type_b = type_b,
                day = bacteria$day, v_sample = var(number), mu_sample = mean(number))

stan_fit = stan(model_code = stanmod, data = stan_dat, iter = 10^5)
```

```
##
## SAMPLING FOR MODEL 'fd2ad801503c3ccaa7fc58f840ea75e1' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.8e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 1: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 1: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 1: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 1: Iteration: 40000 / 100000 [ 40%]  (Warmup)
```

```
## Chain 1: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 1: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 1: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 1: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 1: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 1: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 1: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.62084 seconds (Warm-up)
## Chain 1:                2.30199 seconds (Sampling)
## Chain 1:                3.92282 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'fd2ad801503c3ccaa7fc58f840ea75e1' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 2: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 2: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 2: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 2: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 2: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 2: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 2: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 2: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 2: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 2: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 2: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.60756 seconds (Warm-up)
## Chain 2:                2.14932 seconds (Sampling)
## Chain 2:                3.75688 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'fd2ad801503c3ccaa7fc58f840ea75e1' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 3: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 3: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 3: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 3: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 3: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 3: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 3: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 3: Iteration: 70000 / 100000 [ 70%]  (Sampling)
```

```
## Chain 3: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 3: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 3: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.62707 seconds (Warm-up)
## Chain 3:                1.95422 seconds (Sampling)
## Chain 3:                3.58128 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'fd2ad801503c3ccaa7fc58f840ea75e1' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 4: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 4: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 4: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 4: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 4: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 4: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 4: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 4: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 4: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 4: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 4: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.67857 seconds (Warm-up)
## Chain 4:                1.90955 seconds (Sampling)
## Chain 4:                3.58812 seconds (Total)
## Chain 4:
```

```r
#sso <- launch_shinystan(stan_fit)

# Mean, 0.025 quantile, and 0.975 quantile
pl_cpi = summary(stan_fit)$summary[,c("mean", "2.5%", "97.5%")]
print(pl_cpi, digits = 2)
```

```
##              mean   2.5%  97.5%
## beta_0       6.92  6.024  7.827
## beta_1       0.80  0.551  1.055
## alpha_b      2.46  1.753  3.172
## sigmasq      0.32  0.099  0.937
## mu[1]        7.73  7.024  8.439
## mu[2]        8.53  7.974  9.094
## mu[3]        9.34  8.834  9.841
## mu[4]       10.14  9.577 10.704
## mu[5]       10.95 10.232 11.658
## mu[6]       10.19  9.480 10.905
## mu[7]       11.00 10.433 11.556
## mu[8]       11.80 11.299 12.304
## mu[9]       12.61 12.043 13.168
## mu[10]      13.41 12.699 14.117
```

```
## Rbsq         0.90  0.716  0.970
## log_lik[1]  -0.61 -1.881  0.081
## log_lik[2]  -0.85 -2.169 -0.109
## log_lik[3]  -0.48 -1.275  0.084
## log_lik[4]  -0.38 -1.157  0.169
## log_lik[5]  -1.12 -3.160 -0.124
## log_lik[6]  -0.53 -1.654  0.116
## log_lik[7]  -1.44 -3.498 -0.369
## log_lik[8]  -0.70 -1.713 -0.064
## log_lik[9]  -0.38 -1.140  0.173
## log_lik[10] -0.98 -2.812 -0.071
## lp__         1.68 -2.848  3.898
```

```
summary(stan_fit)$summary
```

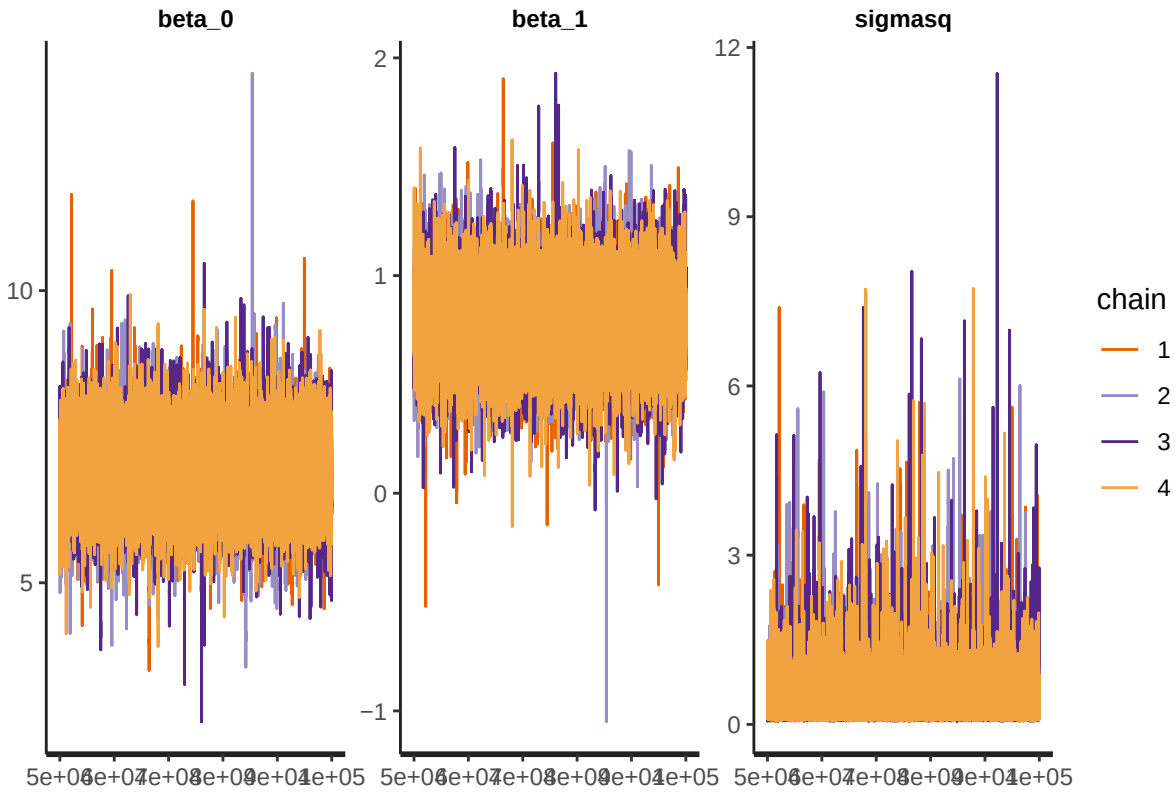
```
##                  mean      se_mean         sd        2.5%        25%
## beta_0       6.9242391 0.0017001561 0.45127495  6.02366701  6.6518122
## beta_1       0.8048263 0.0004570070 0.12573857  0.55088303  0.7297262
## alpha_b      2.4621605 0.0011853039 0.35484728  1.75310955  2.2484734
## sigmasq      0.3164940 0.0010434357 0.25589681  0.09936629  0.1754526
## mu[1]        7.7290654 0.0013007819 0.35376272  7.02414053  7.5152485
## mu[2]        8.5338916 0.0009545228 0.27976315  7.97391839  8.3644483
## mu[3]        9.3387179 0.0007410366 0.25101225  8.83367242  9.1875690
## mu[4]       10.1435442 0.0007782520 0.28172195  9.57706080  9.9760325
## mu[5]       10.9483705 0.0010363085 0.35685815 10.23182204 10.7345817
## mu[6]       10.1912259 0.0011390829 0.35642266  9.47966268  9.9766008
## mu[7]       10.9960522 0.0008123998 0.28164827 10.43282350 10.8274441
## mu[8]       11.8008784 0.0006648674 0.25146522 11.29885519 11.6509043
## mu[9]       12.6057047 0.0008002212 0.28064950 12.04282911 12.4376493
## mu[10]      13.4105310 0.0011195666 0.35484348 12.69861839 13.1973658
## Rbsq         0.9039666 0.0003166084 0.07764645  0.71564115  0.8871157
## log_lik[1]  -0.6122468 0.0018470509 0.50199133 -1.88092748 -0.8402726
## log_lik[2]  -0.8548820 0.0016256767 0.52962009 -2.16926597 -1.1154398
## log_lik[3]  -0.4764661 0.0012135726 0.34698149 -1.27461494 -0.6755707
## log_lik[4]  -0.3830440 0.0013410197 0.33954674 -1.15651389 -0.5787716
## log_lik[5]  -1.1167602 0.0020360665 0.79246567 -3.15952674 -1.4735925
## log_lik[6]  -0.5314455 0.0016497646 0.45295334 -1.65369120 -0.7477121
## log_lik[7]  -1.4437971 0.0020755494 0.81623353 -3.49840372 -1.8444348
## log_lik[8]  -0.7021873 0.0011492779 0.42073015 -1.71283258 -0.9248132
## log_lik[9]  -0.3753134 0.0013367926 0.33659279 -1.14023712 -0.5698352
## log_lik[10] -0.9775445 0.0020370136 0.71446925 -2.81204127 -1.2912632
## lp__         1.6821810 0.0083201353 1.79894878 -2.84786900  0.8026131
##                  50%        75%       97.5%     n_eff      Rhat
## beta_0       6.9229089  7.1956037  7.82687369  70453.88 1.0000716
## beta_1       0.8050270  0.8806989  1.05498998  75699.19 1.0000785
## alpha_b      2.4620035  2.6747009  3.17234521  89623.85 1.0000012
## sigmasq      0.2497361  0.3720290  0.93715198  60144.84 1.0001289
## mu[1]        7.7276522  7.9420707  8.43883718  73963.11 1.0000605
## mu[2]        8.5329498  8.7030505  9.09374046  85903.01 1.0000389
## mu[3]        9.3386893  9.4906626  9.84072316 114738.84 1.0000158
## mu[4]       10.1437370 10.3128635 10.70408193 131039.09 1.0000181
## mu[5]       10.9493581 11.1637521 11.65794747 118580.44 1.0000342
## mu[6]       10.1906525 10.4053683 10.90492288  97908.39 1.0000504
## mu[7]       10.9956861 11.1652944 11.55587699 120191.71 1.0000204
## mu[8]       11.8007050 11.9518535 12.30448470 143049.33 0.9999891
```

```
## mu[9]       12.6056997 12.7739138 13.16819064 123000.95 0.9999935
## mu[10]      13.4108945 13.6251933 14.11731971 100455.52 1.0000170
## Rbsq         0.9242229  0.9467626  0.96984941  60144.84 1.0001289
## log_lik[1]  -0.5180743 -0.2725923  0.08066680  73864.32 1.0001199
## log_lik[2]  -0.7569457 -0.4841466 -0.10904758 106135.46 1.0000574
## log_lik[3]  -0.4361592 -0.2322268  0.08384973  81748.74 1.0000469
## log_lik[4]  -0.3441464 -0.1435465  0.16865835  64110.45 1.0001230
## log_lik[5]  -0.9251523 -0.5611909 -0.12444571 151487.57 1.0000012
## log_lik[6]  -0.4534338 -0.2228401  0.11640038  75381.19 1.0000834
## log_lik[7]  -1.2672836 -0.8595943 -0.36924983 154654.55 1.0000106
## log_lik[8]  -0.6394095 -0.4089768 -0.06439201 134016.16 0.9999996
## log_lik[9]  -0.3365713 -0.1382332  0.17269060  63398.88 1.0001046
## log_lik[10] -0.8083176 -0.4840577 -0.07147446 123020.99 1.0000175
## lp__         2.0955643  3.0038543  3.89845869  46749.48 1.0001721
```

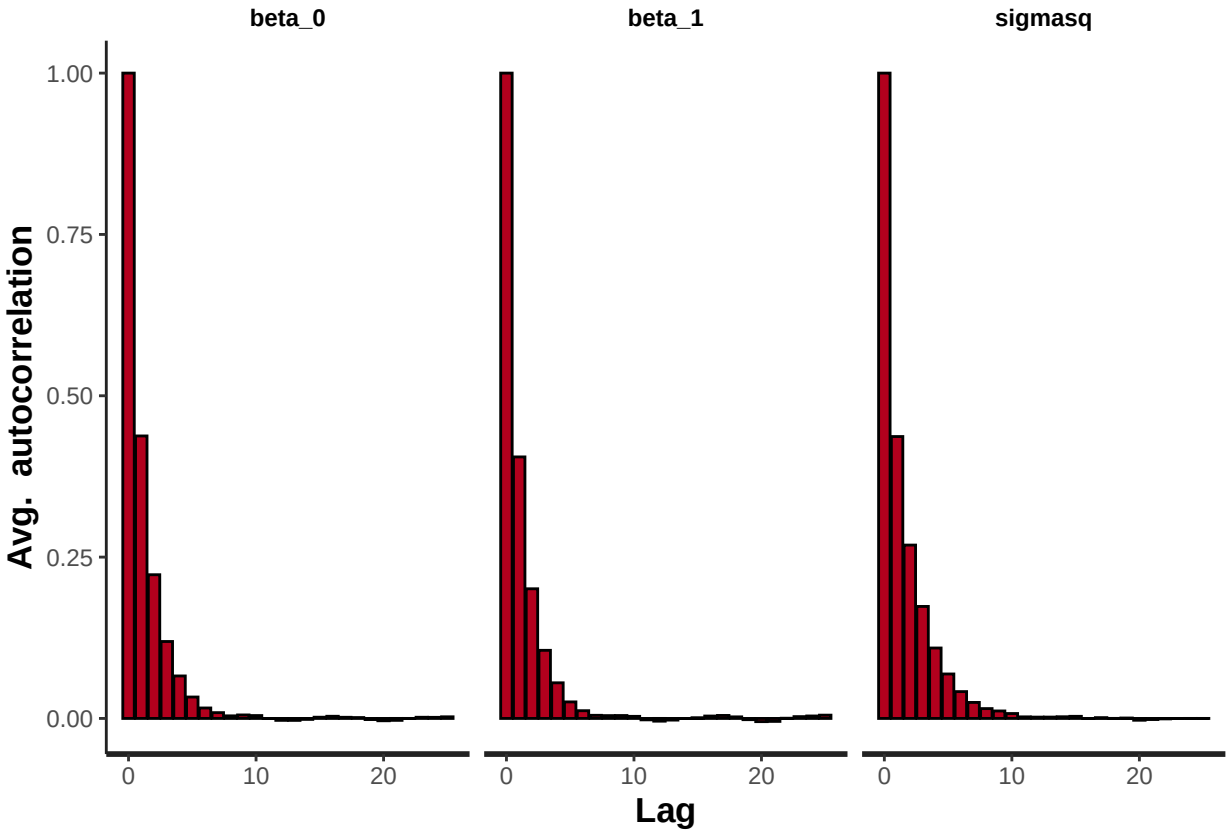
```
# Gelman-Rubin statistic
summary(stan_fit)$summary[,"Rhat"]
```

```
##      beta_0      beta_1     alpha_b     sigmasq       mu[1]       mu[2]
##   1.0000716   1.0000785   1.0000012   1.0001289   1.0000605   1.0000389
##       mu[3]       mu[4]       mu[5]       mu[6]       mu[7]       mu[8]
##   1.0000158   1.0000181   1.0000342   1.0000504   1.0000204   0.9999891
##       mu[9]      mu[10]        Rbsq  log_lik[1]  log_lik[2]  log_lik[3]
##   0.9999935   1.0000170   1.0001289   1.0001199   1.0000574   1.0000469
##  log_lik[4]  log_lik[5]  log_lik[6]  log_lik[7]  log_lik[8]  log_lik[9]
##   1.0001230   1.0000012   1.0000834   1.0000106   0.9999996   1.0001046
## log_lik[10]        lp__
##   1.0000175   1.0001721
```

```
# check convergence with trace plots
stan_trace(stan_fit, c("beta_0", "beta_1", "sigmasq"))
```




```
# Check the ACF of draws
stan_ac(stan_fit, c("beta_0", "beta_1", "sigmasq"))
```



```r
# WAIC and LOOC
log_lik = extract_log_lik(stan_fit, merge_chains = FALSE)
r_eff = exp(relative_eff(log_lik))

waic(log_lik)
```

```
## Warning:
## 3 (30.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 200000 by 10 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic     -9.3 1.5
## p_waic         3.1 0.7
## waic          18.7 3.1
##
## 3 (30.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```r
loo(log_lik, r_eff = r_eff)
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
##
## Computed from 200000 by 10 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     -9.8 1.7
## p_loo         3.5 0.9
## looic        19.6 3.4
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     7     70.0%   102513
##  (0.5, 0.7]   (ok)       3     30.0%   11987
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
#### A separate lines (interaction) model using day as the quantitative predictor and using type = "A"

type_b = c(type == 'B')

# Create model
stanmod = "
data {
  int<lower=1> n; // number of observations
  vector[n] y; // data
  vector[n] day; //day
  vector[n] type_b; //
  real<lower=0> v_sample; // sample variance of y
  real mu_sample; // sample mean of y
```

```
}

parameters {
  real beta_0;
  real beta_1;
  real alpha_b;
  real delta_b;
  real<lower=0> sigmasq;

}

transformed parameters {
  vector[n] mu;        // mean of observations
  for(i in 1:n){
    mu[i] = beta_0 + alpha_b*type_b[i] + beta_1*day[i] + delta_b*day[i]*type_b[i];
  }
}
model {

  // priors
  beta_0 ~ normal(0.0, 100);
  beta_1 ~ normal(0.0, 100);
  delta_b ~ normal(0.0, 100);
  alpha_b ~ normal(0.0, 100);

  sigmasq ~ inv_gamma(0.01, 0.01);

  // data distribution
  for(i in 1:n){
    y[i] ~ normal(mu[i], sqrt(sigmasq));
  }
}
generated quantities {
  real Rbsq;          // goodness-of-fit
  vector[n] log_lik;  // log likelihood of data
  Rbsq = 1 - sigmasq/v_sample;
  for (i in 1:n) log_lik[i] = normal_lpdf(y[i]|mu[i], sqrt(sigmasq));
}
"

stan_dat = list(n = length(number), y = number, type_b = type_b,
                day = bacteria$day, v_sample = var(number), mu_sample = mean(number))

stan_fit = stan(model_code = stanmod, data = stan_dat, iter = 10^5)
```

```
##
## SAMPLING FOR MODEL '2cb7b11a309d42c18b2bbbac59d205cd' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
```

```
## Chain 1: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 1: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 1: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 1: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 1: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 1: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 1: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 1: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 1: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 1: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 1: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 1: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.04025 seconds (Warm-up)
## Chain 1:                3.19825 seconds (Sampling)
## Chain 1:                5.23849 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '2cb7b11a309d42c18b2bbbac59d205cd' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 2: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 2: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 2: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 2: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 2: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 2: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 2: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 2: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 2: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 2: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 2: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 2.22512 seconds (Warm-up)
## Chain 2:                3.01111 seconds (Sampling)
## Chain 2:                5.23623 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '2cb7b11a309d42c18b2bbbac59d205cd' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 3: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 3: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 3: Iteration: 30000 / 100000 [ 30%]  (Warmup)
```

```
## Chain 3: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 3: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 3: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 3: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 3: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 3: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 3: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 3: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 2.22964 seconds (Warm-up)
## Chain 3:                2.67515 seconds (Sampling)
## Chain 3:                4.90479 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '2cb7b11a309d42c18b2bbbac59d205cd' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 4: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 4: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 4: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 4: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 4: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 4: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 4: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 4: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 4: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 4: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 4: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 2.21032 seconds (Warm-up)
## Chain 4:                3.04794 seconds (Sampling)
## Chain 4:                5.25826 seconds (Total)
## Chain 4:
```

```
## Warning: There were 2 divergent transitions after warmup. See
## http://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
```

```
#sso <- launch_shinystan(stan_fit)

# Mean, 0.025 quantile, and 0.975 quantile
pl_cpi = summary(stan_fit)$summary[,c("mean", "2.5%", "97.5%")]
print(pl_cpi, digits = )
```

```
##                  mean        2.5%        97.5%
## beta_0       7.5394508  6.62943677  8.444606677
## beta_1       0.6003055  0.32805284  0.874204553
## alpha_b      1.2313326 -0.04995776  2.511540214
## delta_b      0.4094209  0.02427005  0.796998630
```

```
## sigmasq      0.1877189  0.05177792  0.607139586
## mu[1]        8.1397563  7.46838428  8.809037871
## mu[2]        8.7400617  8.26676118  9.215613726
## mu[3]        9.3403672  8.95393097  9.728217174
## mu[4]        9.9406727  9.46658263 10.417101992
## mu[5]       10.5409781  9.86978053 11.214749172
## mu[6]        9.7805097  9.11150772 10.448595359
## mu[7]       10.7902361 10.31493317 11.262417692
## mu[8]       11.7999625 11.41156611 12.187495803
## mu[9]       12.8096889 12.33275308 13.284724741
## mu[10]      13.8194153 13.14714121 14.493797691
## Rbsq         0.9430407  0.81577640  0.984289092
## log_lik[1]  -0.3450512 -1.87033762  0.433183704
## log_lik[2]  -0.3873963 -1.59695856  0.325574651
## log_lik[3]  -0.2972894 -1.20192427  0.323296994
## log_lik[4]  -0.3870813 -1.59447311  0.326307449
## log_lik[5]  -0.3448608 -1.86878659  0.432076722
## log_lik[6]  -0.4585920 -2.23112311  0.389948124
## log_lik[7]  -1.0812089 -3.21694431  0.004016203
## log_lik[8]  -0.7078445 -2.06071618  0.070683032
## log_lik[9]  -0.2933975 -1.37441564  0.380364023
## log_lik[10] -0.2932854 -1.69497452  0.453635906
## lp__         4.5292153 -1.03327188  7.403082234
```

```
summary(stan_fit)$summary
```

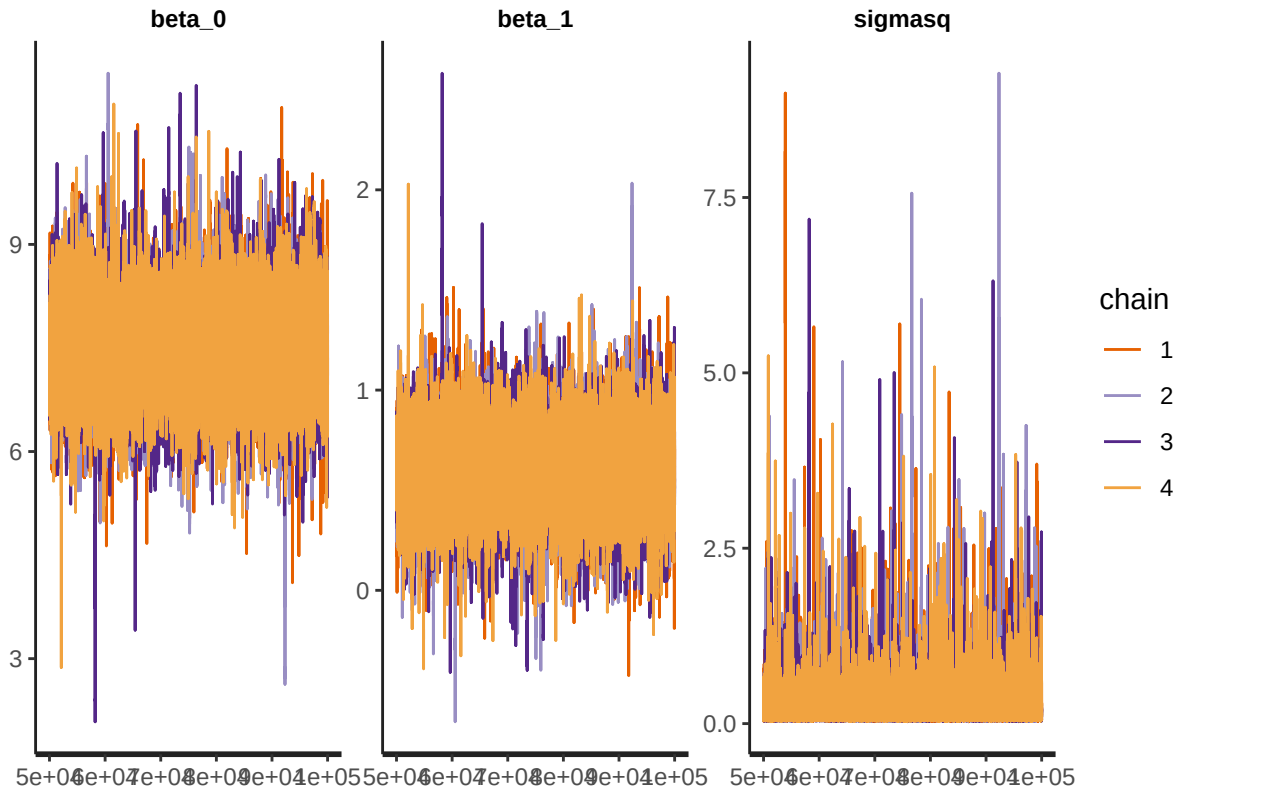
```
##                  mean      se_mean         sd        2.5%         25%
## beta_0       7.5394508 0.0021822971 0.45631273  6.62943677  7.27338023
## beta_1       0.6003055 0.0006559149 0.13742768  0.32805284  0.51966341
## alpha_b      1.2313326 0.0030721649 0.64340429 -0.04995776  0.85678781
## delta_b      0.4094209 0.0009253358 0.19407186  0.02427005  0.29586569
## sigmasq      0.1877189 0.0009625100 0.18749483  0.05177792  0.09519973
## mu[1]        8.1397563 0.0015459920 0.33711726  7.46838428  7.94393510
## mu[2]        8.7400617 0.0009379885 0.23842720  8.26676118  8.60123855
## mu[3]        9.3403672 0.0004744185 0.19447299  8.95393097  9.22699612
## mu[4]        9.9406727 0.0006596919 0.23783334  9.46658263  9.80231944
## mu[5]       10.5409781 0.0012268740 0.33627724  9.86978053 10.34439788
## mu[6]        9.7805097 0.0011377681 0.33577054  9.11150772  9.58435718
## mu[7]       10.7902361 0.0007109502 0.23733467 10.31493317 10.65191001
## mu[8]       11.7999625 0.0004205578 0.19407625 11.41156611 11.68666081
## mu[9]       12.8096889 0.0005638618 0.23823392 12.33275308 12.67058653
## mu[10]      13.8194153 0.0009584812 0.33704178 13.14714121 13.62211690
## Rbsq         0.9430407 0.0002920532 0.05689132  0.81577640  0.93417320
## log_lik[1]  -0.3450512 0.0026077835 0.59254909 -1.87033762 -0.59176569
## log_lik[2]  -0.3873963 0.0018578663 0.49135885 -1.59695856 -0.63236445
## log_lik[3]  -0.2972894 0.0013642125 0.39045291 -1.20192427 -0.51525836
## log_lik[4]  -0.3870813 0.0015821082 0.49214609 -1.59447311 -0.63062866
## log_lik[5]  -0.3448608 0.0023369425 0.59229149 -1.86878659 -0.59124549
## log_lik[6]  -0.4585920 0.0023347723 0.67657033 -2.23112311 -0.72609795
## log_lik[7]  -1.0812089 0.0022313100 0.84247111 -3.21694431 -1.47952092
## log_lik[8]  -0.7078445 0.0012913838 0.54748559 -2.06071618 -0.97829560
## log_lik[9]  -0.2933975 0.0015835907 0.45040292 -1.37441564 -0.52616541
## log_lik[10] -0.2932854 0.0021914196 0.55223638 -1.69497452 -0.53384640
## lp__         4.5292153 0.0121219573 2.22746496 -1.03327188  3.42177666
##                   50%          75%        97.5%     n_eff      Rhat
```

```
## beta_0       7.5397155  7.805468850  8.444606677  43721.74 1.0001123
## beta_1       0.6001441  0.680427012  0.874204553  43898.88 1.0000901
## alpha_b      1.2330785  1.606465983  2.511540214  43861.03 1.0000901
## delta_b      0.4093292  0.522698746  0.796998630  43987.21 1.0000640
## sigmasq      0.1396896  0.216943206  0.607139586  37946.18 1.0002067
## mu[1]        8.1397138  8.336414440  8.809037871  47549.67 1.0001091
## mu[2]        8.7404238  8.879118240  9.215613726  64612.50 1.0000846
## mu[3]        9.3406223  9.453238299  9.728217174 168033.28 1.0000165
## mu[4]        9.9397868 10.079421326 10.417101992 129976.01 0.9999972
## mu[5]       10.5399294 10.736736057 11.214749172  75126.82 1.0000218
## mu[6]        9.7802386  9.977564188 10.448595359  87091.89 1.0000159
## mu[7]       10.7902416 10.929753625 11.262417692 111440.74 1.0000047
## mu[8]       11.8002209 11.913731668 12.187495803 212957.72 0.9999878
## mu[9]       12.8097866 12.948915355 13.284724741 178509.73 0.9999931
## mu[10]      13.8189014 14.016188792 14.493797691 123651.77 1.0000041
## Rbsq         0.9576142  0.971113666  0.984289092  37946.18 1.0002067
## log_lik[1]  -0.2235920  0.049454887  0.433183704  51630.41 1.0000897
## log_lik[2]  -0.3038450 -0.046190130  0.325574651  69946.93 1.0001040
## log_lik[3]  -0.2476474 -0.023531238  0.323296994  81916.85 1.0000637
## log_lik[4]  -0.3040297 -0.048372428  0.326307449  96764.42 1.0000397
## log_lik[5]  -0.2230574  0.050424195  0.432076722  64235.47 1.0000581
## log_lik[6]  -0.3047642 -0.009906988  0.389948124  83972.45 1.0001002
## log_lik[7]  -0.8878597 -0.485245000  0.004016203 142557.54 1.0000135
## log_lik[8]  -0.6080690 -0.326028033  0.070683032 179735.85 0.9999994
## log_lik[9]  -0.2246243  0.019051615  0.380364023  80894.04 1.0000727
## log_lik[10] -0.1864679  0.076661357  0.453635906  63503.69 1.0001258
## lp__         5.0029386  6.151372989  7.403082234  33765.74 1.0002561
```

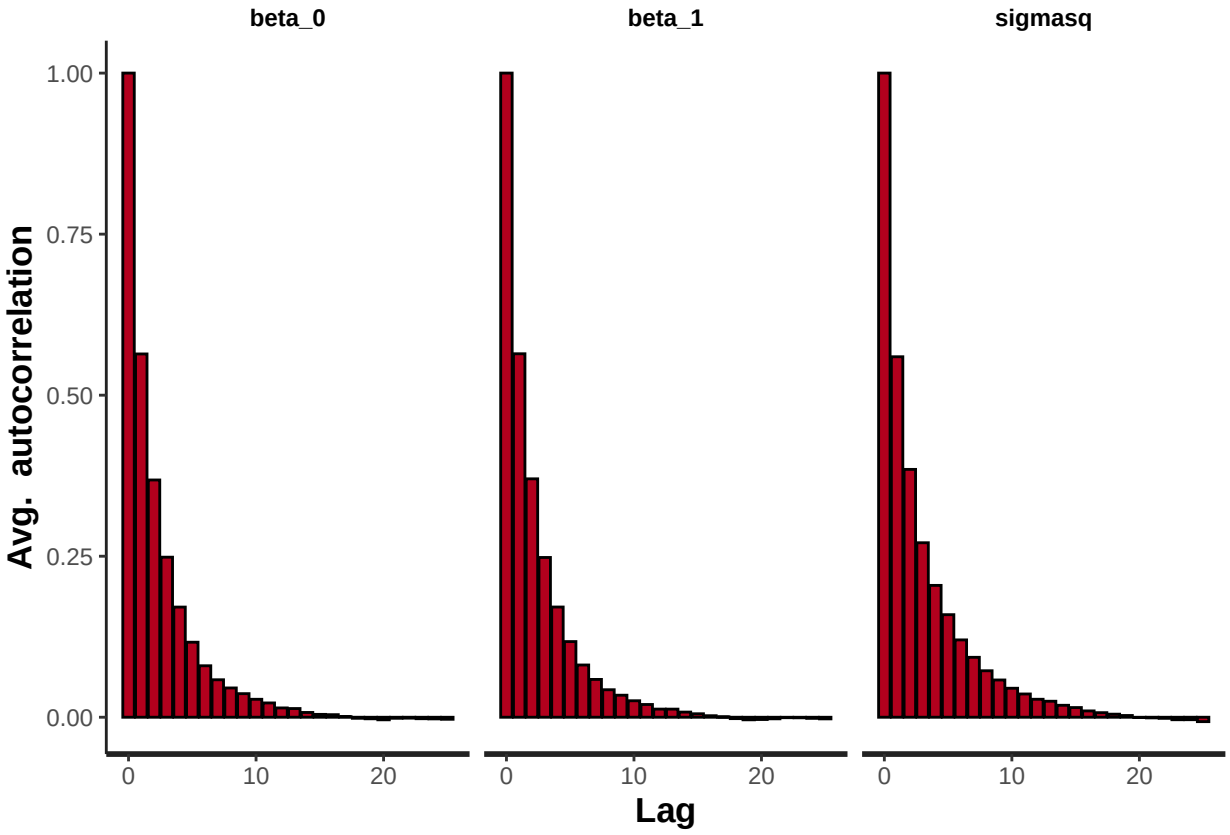
```
# Gelman-Rubin statistic
summary(stan_fit)$summary[,"Rhat"]
```

```
##      beta_0      beta_1     alpha_b     delta_b     sigmasq       mu[1]
##   1.0001123   1.0000901   1.0000901   1.0000640   1.0002067   1.0001091
##       mu[2]       mu[3]       mu[4]       mu[5]       mu[6]       mu[7]
##   1.0000846   1.0000165   0.9999972   1.0000218   1.0000159   1.0000047
##       mu[8]       mu[9]      mu[10]        Rbsq  log_lik[1]  log_lik[2]
##   0.9999878   0.9999931   1.0000041   1.0002067   1.0000897   1.0001040
##  log_lik[3]  log_lik[4]  log_lik[5]  log_lik[6]  log_lik[7]  log_lik[8]
##   1.0000637   1.0000397   1.0000581   1.0001002   1.0000135   0.9999994
##  log_lik[9] log_lik[10]        lp__
##   1.0000727   1.0001258   1.0002561
```

```
# check convergence with trace plots
stan_trace(stan_fit, c("beta_0", "beta_1", "sigmasq"))
```




```
# Check the ACF of draws
stan_ac(stan_fit, c("beta_0", "beta_1", "sigmasq"))
```



```r
# WAIC & LOOIC
log_lik = extract_log_lik(stan_fit, merge_chains = FALSE)
r_eff = exp(relative_eff(log_lik))

waic(log_lik)
```

```
## Warning:
## 2 (20.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 200000 by 10 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic     -6.6 1.1
## p_waic         3.3 0.5
## waic          13.2 2.2
##
## 2 (20.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```r
loo(log_lik, r_eff = r_eff)
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
##
## Computed from 200000 by 10 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     -7.3 1.2
## p_loo         4.0 0.7
## looic        14.5 2.5
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     5     50.0%   155128
##  (0.5, 0.7]   (ok)       2     20.0%   13112
##    (0.7, 1]   (bad)      3     30.0%   7011
##    (1, Inf)   (very bad) 0      0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```r
#sso <- launch_shinystan(stan_fit)
```