final_project_code

2022-11-19

Code Appendix

1. EDA

2. Final Model

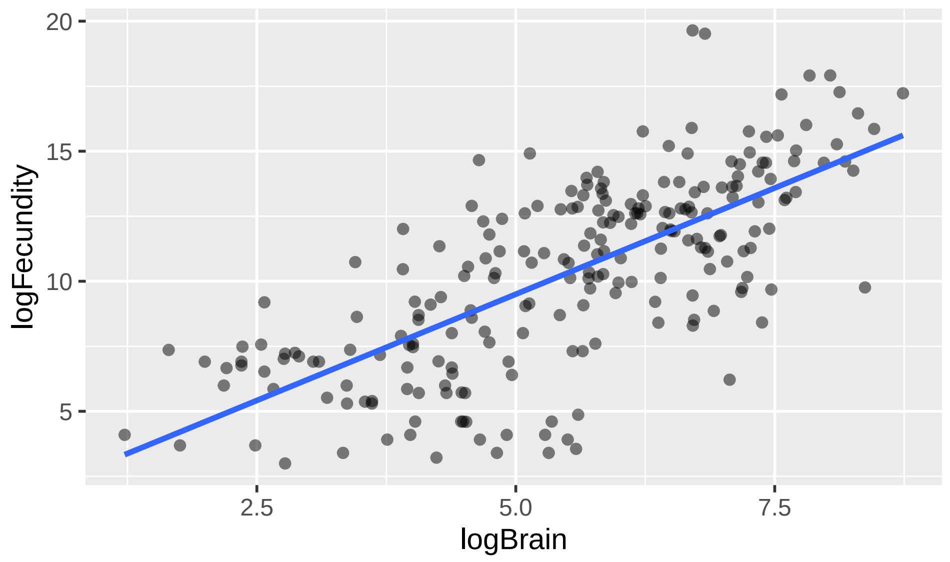
3. Test Models

1. EDA

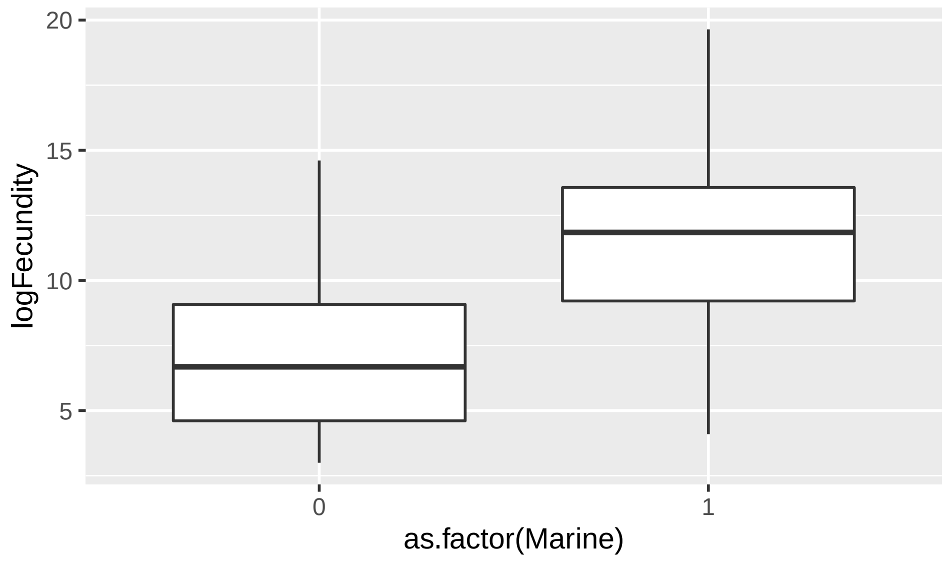
```
# load data
fish_clean <- read_csv("fish_clean.csv") %>%
  select(logBrain, Marine, logFecundity)

# EDA
summary(fish_clean)
```

```
##     logBrain         Marine        logFecundity
##  Min.   :1.224   Min.   :0.0000   Min.   : 2.996
##  1st Qu.:4.409   1st Qu.:0.2500   1st Qu.: 7.361
##  Median :5.705   Median :1.0000   Median :10.747
##  Mean   :5.517   Mean   :0.7478   Mean   :10.346
##  3rd Qu.:6.727   3rd Qu.:1.0000   3rd Qu.:12.899
##  Max.   :8.738   Max.   :1.0000   Max.   :19.644
```

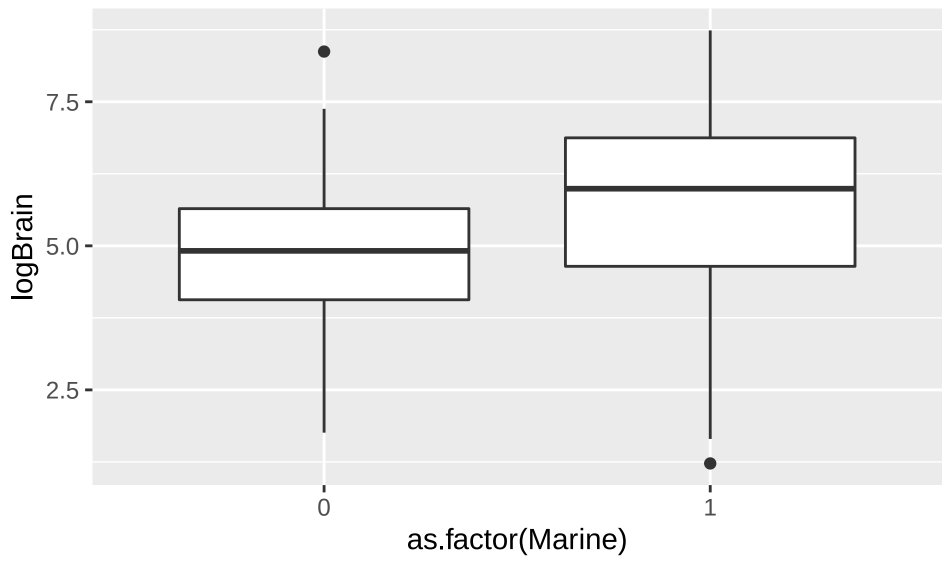
```
# logBrain vs. logFecundity
fish_clean %>%
  ggplot(aes(x=logBrain, y=logFecundity)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE)
```



```r
# Marine vs. logFecundity
fish_clean %>%
  ggplot(aes(x=as.factor(Marine), y=logFecundity)) + geom_boxplot()
```



```r
# Marine vs. logBrain
fish_clean %>%
  ggplot(aes(x=as.factor(Marine), y=logBrain)) + geom_boxplot()
```



2. Final Model

Multiple Linear Regression

```
# run jags

# model
modelString <-"
model {
  ## sampling
  for (i in 1:n){
     y[i] ~ dnorm(mu[i], phi)
     mu[i] <- beta0 + beta1 * x1[i] + beta2 * x2[i]
  }

  ## priors
  beta0 ~ dnorm(0, pow(10, -2))
  beta1 ~ dnorm(0, pow(10, -2))
  beta2 ~ dnorm(0, pow(10, -2))
  phi ~ dgamma(1,1)
  sigma <- sqrt(pow(phi ,-1))
}
"

# initial values
repro_inits <- list(
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95063432,
       beta0 = 0,  beta1 = 1, beta2 = -1, phi = 1),
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95073452,
       beta0 = -1, beta1 = 0, beta2 = 1, phi = 1),
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95086345,
       beta0 = 1, beta1 = -1, beta2 = 0, phi = 1)
)

fish_data <- list(
  n = nrow(fish_clean),
  x1 = fish_clean$logBrain - mean(fish_clean$logBrain), # centering method
  x2 = fish_clean$Marine,
  y = fish_clean$logFecundity
)

posterior <- run.jags(
  modelString,
  data = fish_data,
  inits = repro_inits,
  n.chains = 3,
  adapt = 1000,
  burnin = 5000,
  sample = 8000,
  thin = 5,
  monitor = c("beta0", "beta1", "beta2", "sigma"),
  silent.jags = TRUE
)
```

```
# summary
summary(posterior)
```

```
##        Lower95   Median  Upper95     Mean         SD Mode        MCerr MC%ofSD
## beta0 7.425285 7.996534 8.573723 7.998434 0.29424313   NA 0.0024468748     0.8
## beta1 1.260137 1.441735 1.625973 1.441796 0.09339034   NA 0.0006253054     0.7
## beta2 2.461870 3.139377 3.801130 3.135503 0.34110290   NA 0.0028460284     0.8
## sigma 1.971378 2.174588 2.375751 2.178795 0.10343166   NA 0.0006578496     0.6
##       SSeff       AC.50     psrf
## beta0 14461 0.016028586 1.000237
## beta1 22306 0.005600326 1.000100
## beta2 14365 0.012773278 1.000151
## sigma 24720 0.016376975 1.000148
```

```
summary(posterior$mcmc[[1]], digits = 3)
```

```
##
## Iterations = 6001:45996
## Thinning interval = 5
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##        Mean      SD Naive SE Time-series SE
## beta0 8.003 0.29496 0.003298       0.004286
## beta1 1.443 0.09342 0.001044       0.001069
## beta2 3.131 0.34293 0.003834       0.004973
## sigma 2.179 0.10434 0.001167       0.001136
##
## 2. Quantiles for each variable:
##
##        2.5%   25%   50%   75% 97.5%
## beta0 7.435 7.802 8.002 8.198 8.583
## beta1 1.263 1.379 1.442 1.506 1.628
## beta2 2.441 2.900 3.134 3.368 3.795
## sigma 1.989 2.106 2.175 2.246 2.398
```

```
summary(posterior$mcmc[[2]], digits = 3)
```

```
##
## Iterations = 6001:45996
## Thinning interval = 5
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean      SD Naive SE Time-series SE
## beta0 7.992 0.29577 0.003307       0.004240
## beta1 1.441 0.09439 0.001055       0.001125
## beta2 3.141 0.34242 0.003828       0.004953
## sigma 2.180 0.10320 0.001154       0.001154
##
## 2. Quantiles for each variable:
##
##        2.5%   25%   50%   75% 97.5%
## beta0 7.418 7.795 7.989 8.190 8.574
## beta1 1.253 1.379 1.442 1.506 1.624
## beta2 2.462 2.913 3.147 3.373 3.809
## sigma 1.990 2.108 2.177 2.249 2.395
```

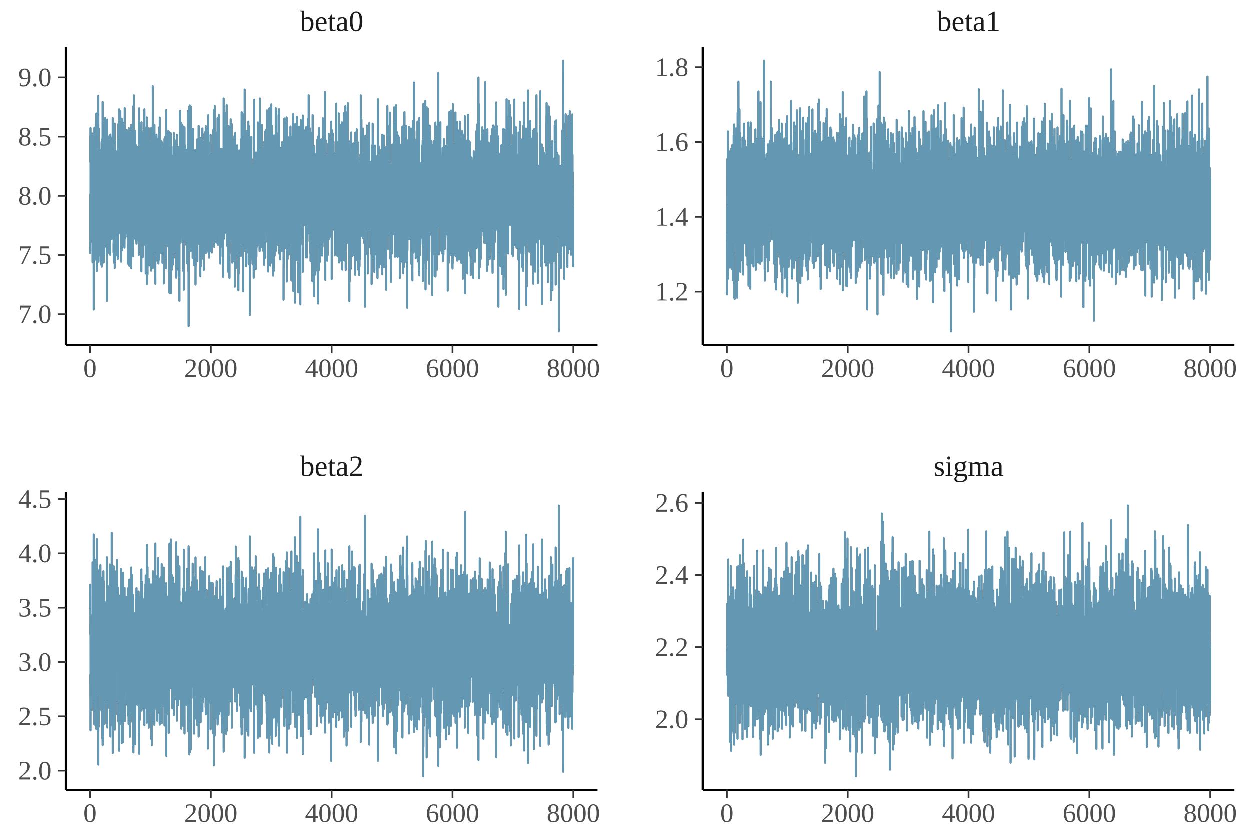
```
summary(posterior$mcmc[[3]], digits = 3)
```

```
##
## Iterations = 6001:45996
## Thinning interval = 5
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean      SD Naive SE Time-series SE
## beta0 8.001 0.29191 0.003264       0.004188
## beta1 1.441 0.09236 0.001033       0.001057
## beta2 3.134 0.33789 0.003778       0.004862
## sigma 2.177 0.10274 0.001149       0.001129
##
## 2. Quantiles for each variable:
##
##        2.5%   25%   50%   75% 97.5%
## beta0 7.435 7.807 7.998 8.198 8.575
## beta1 1.262 1.379 1.441 1.502 1.624
## beta2 2.476 2.907 3.137 3.361 3.804
## sigma 1.987 2.106 2.173 2.244 2.391
```

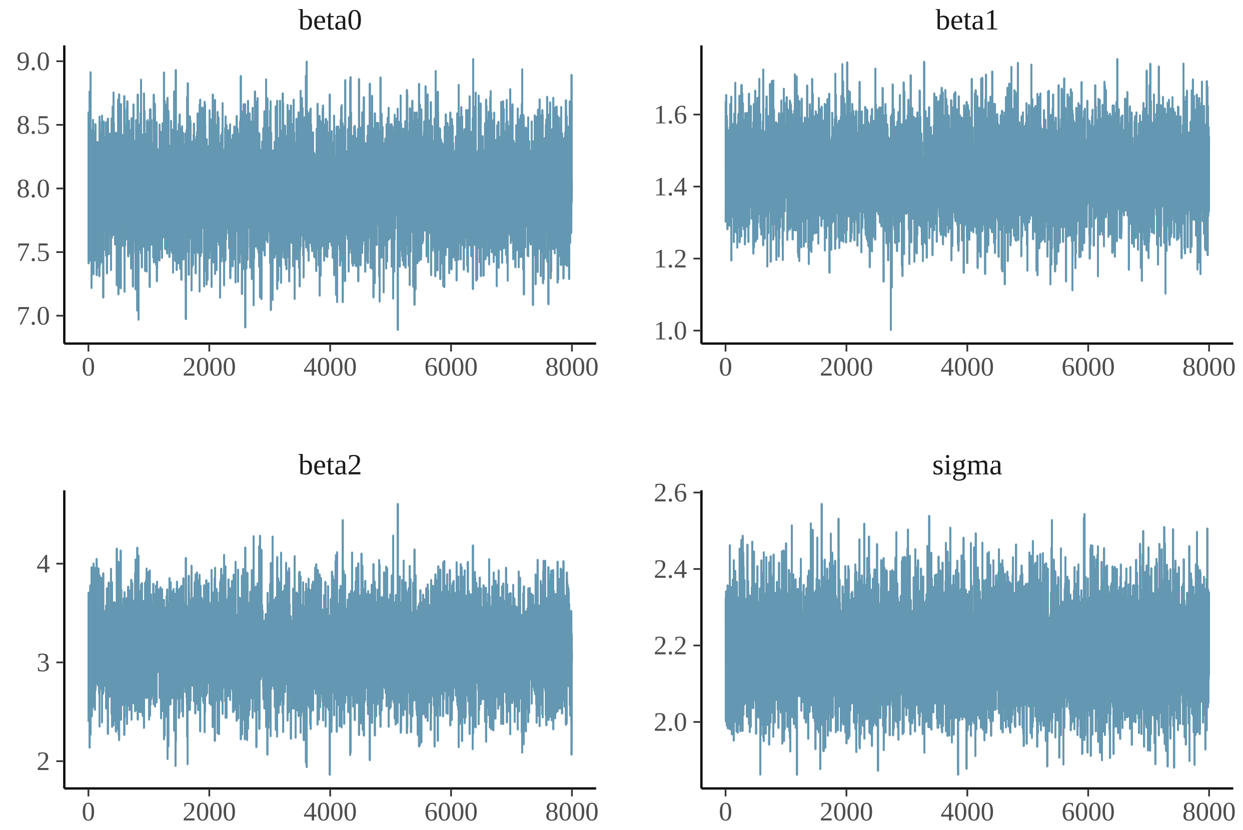
```
# summary table
df <- data.frame(summary(posterior), digits = 3) %>%
  mutate(across(where(is.numeric), ~ round(., 3))) %>%
  mutate(Param = c("beta0", "beta1", "beta2", "sigma")) %>%
  select(Param, Median, Mean, SD, Lower95, Upper95)
df %>%
  gt()
```

Param	Median	Mean	SD	Lower95	Upper95
beta0	7.997	7.998	0.294	7.425	8.574
beta1	1.442	1.442	0.093	1.260	1.626
beta2	3.139	3.136	0.341	2.462	3.801
sigma	2.175	2.179	0.103	1.971	2.376

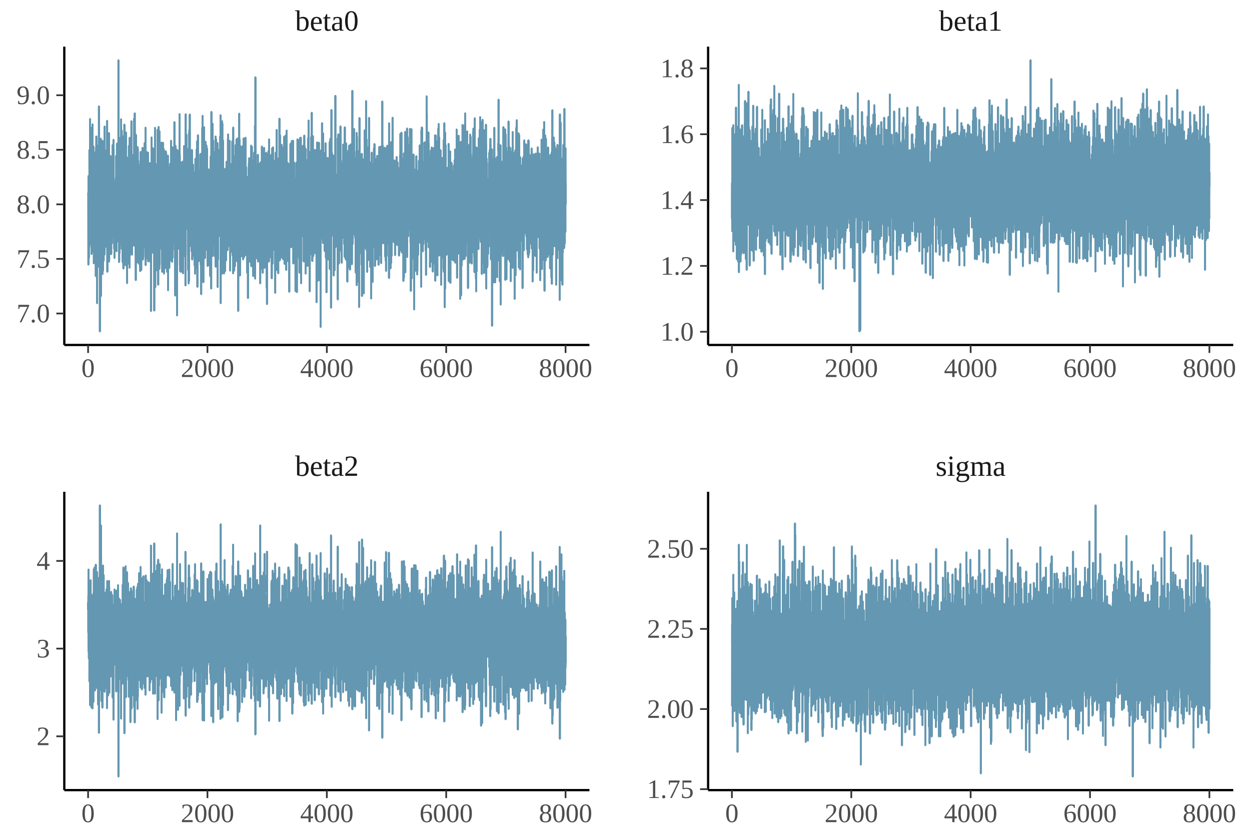
```
# convergence dsiagnostics

# convert to an mcmc object
post_mcmc1 <- as.mcmc(posterior$mcmc[[1]])
post_mcmc2 <- as.mcmc(posterior$mcmc[[2]])
post_mcmc3 <- as.mcmc(posterior$mcmc[[3]])

# trace plot
mcmc_trace(post_mcmc1)
```



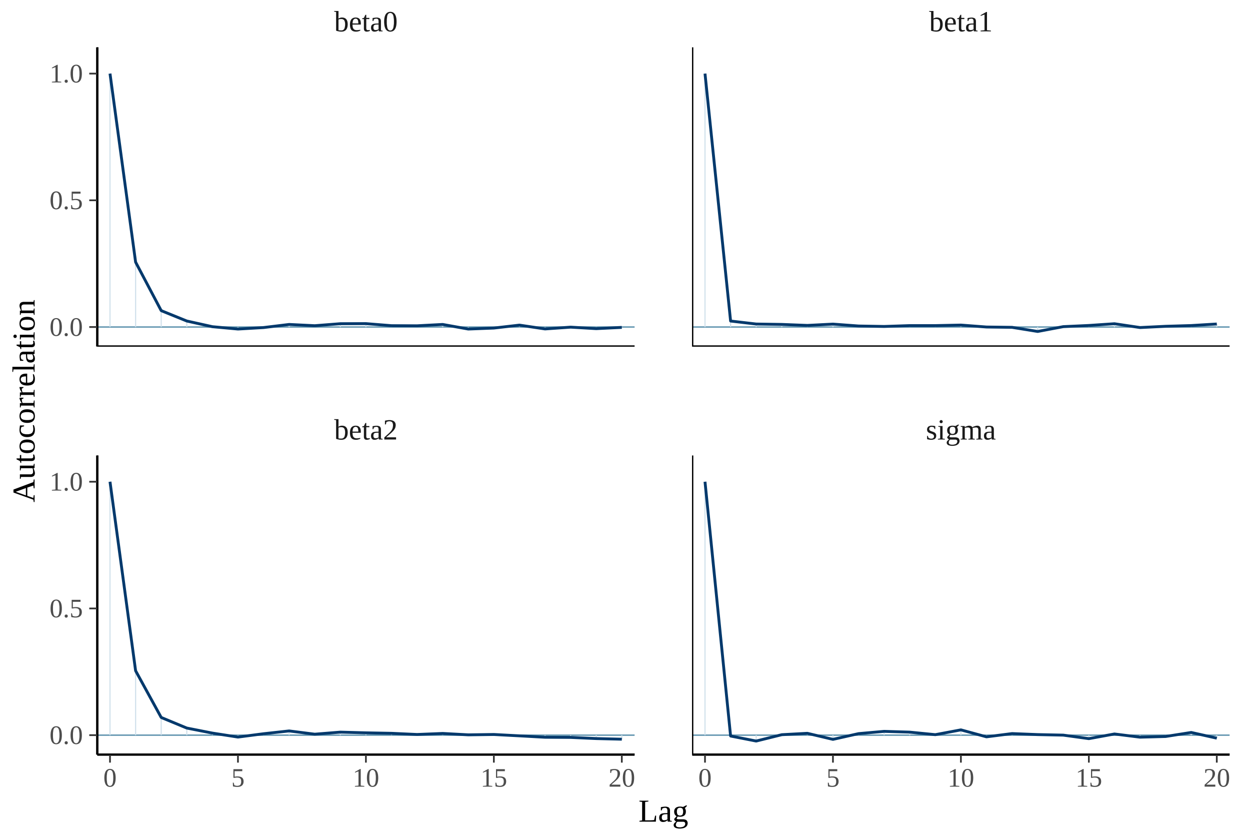
```
mcmc_trace(post_mcmc2)
```



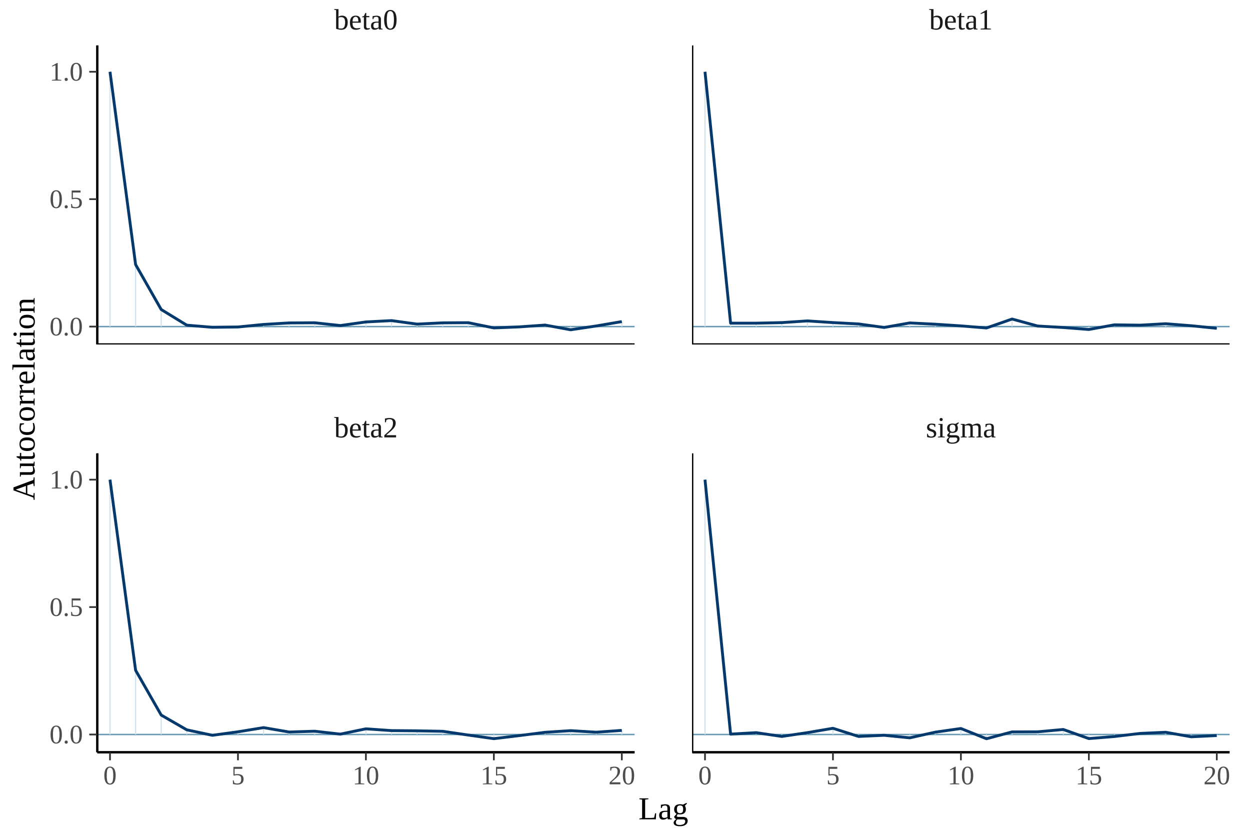
```
mcmc_trace(post_mcmc3)
```



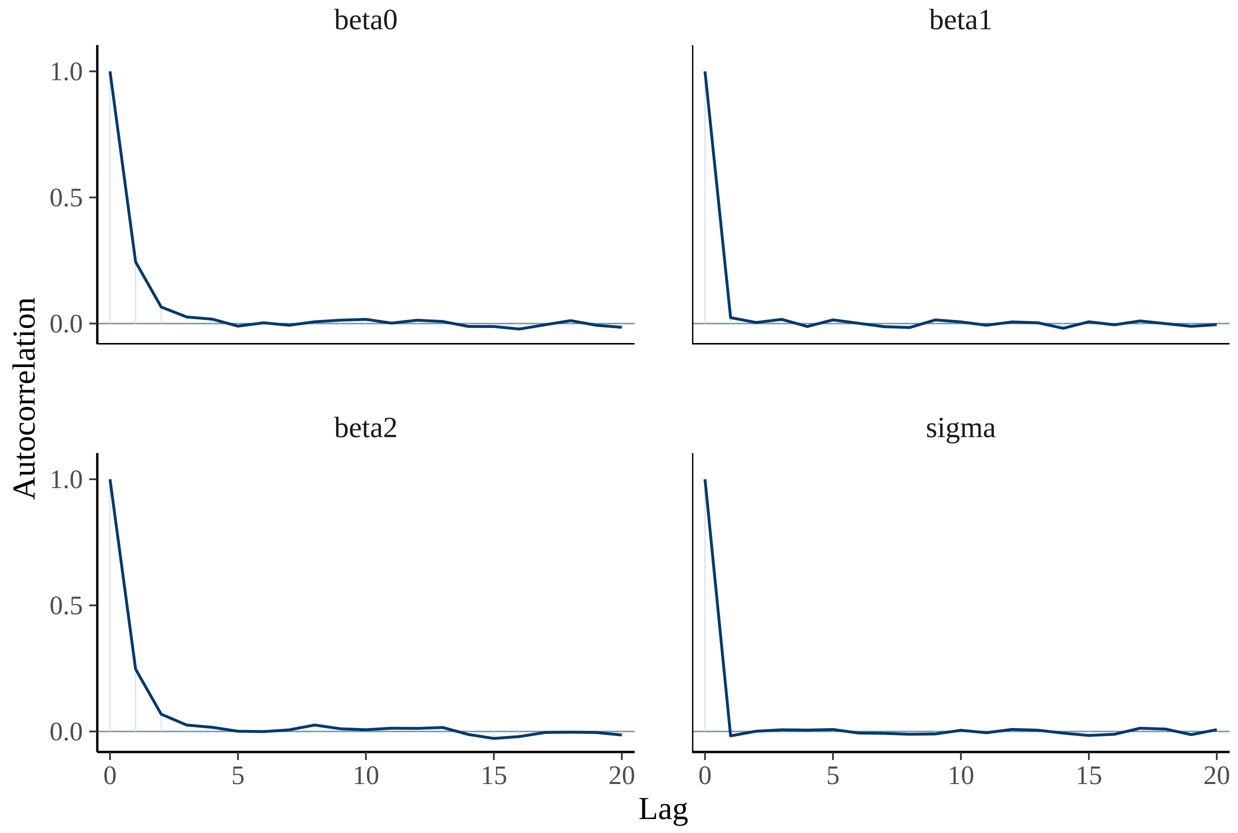
```
# ACF plot
mcmc_acf(post_mcmc1)
```



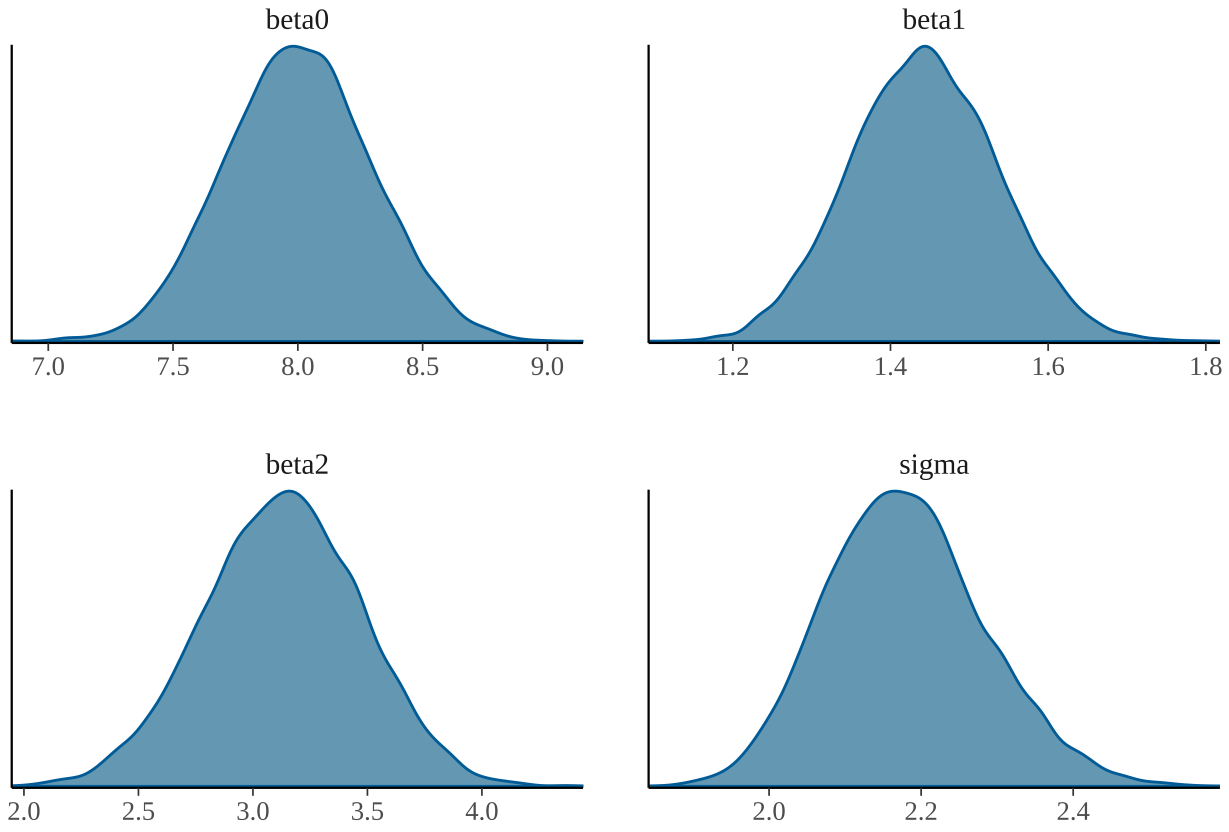
```
mcmc_acf(post_mcmc2)
```



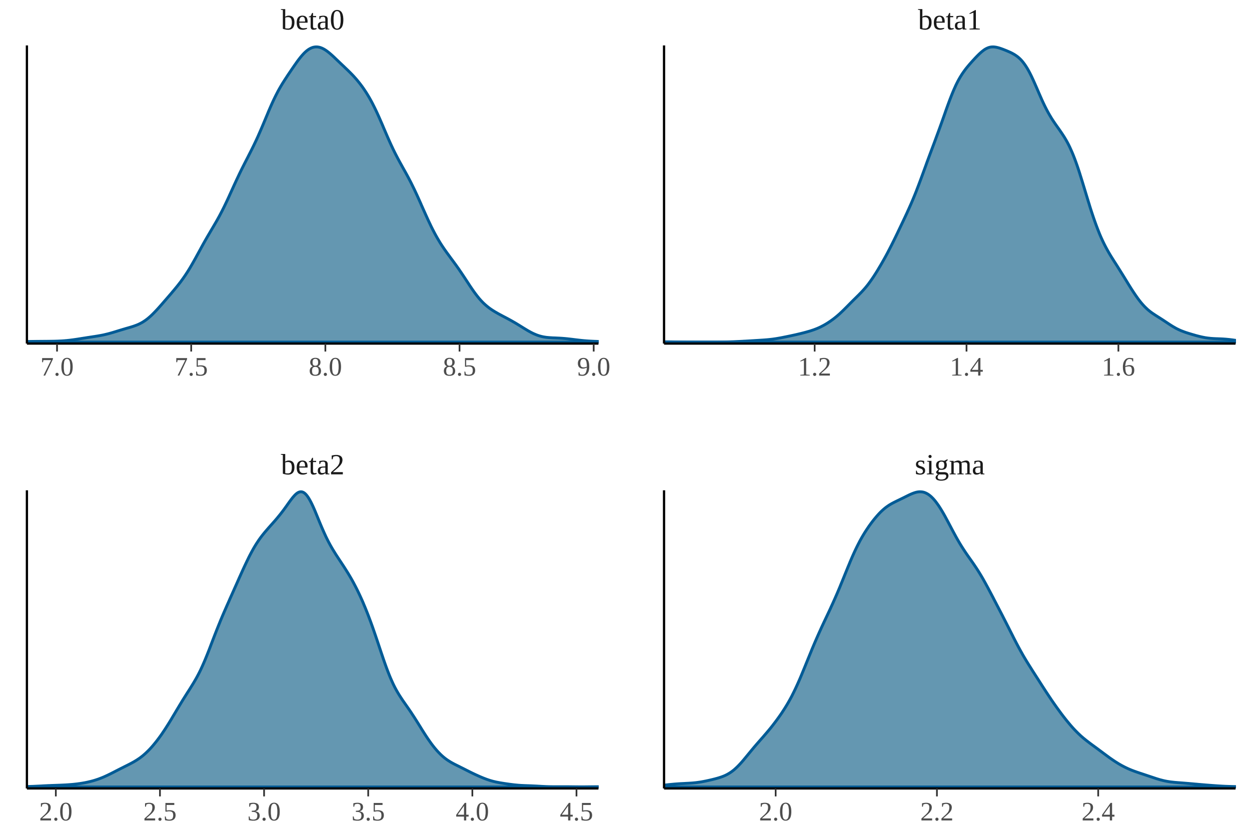
```
mcmc_acf(post_mcmc3)
```



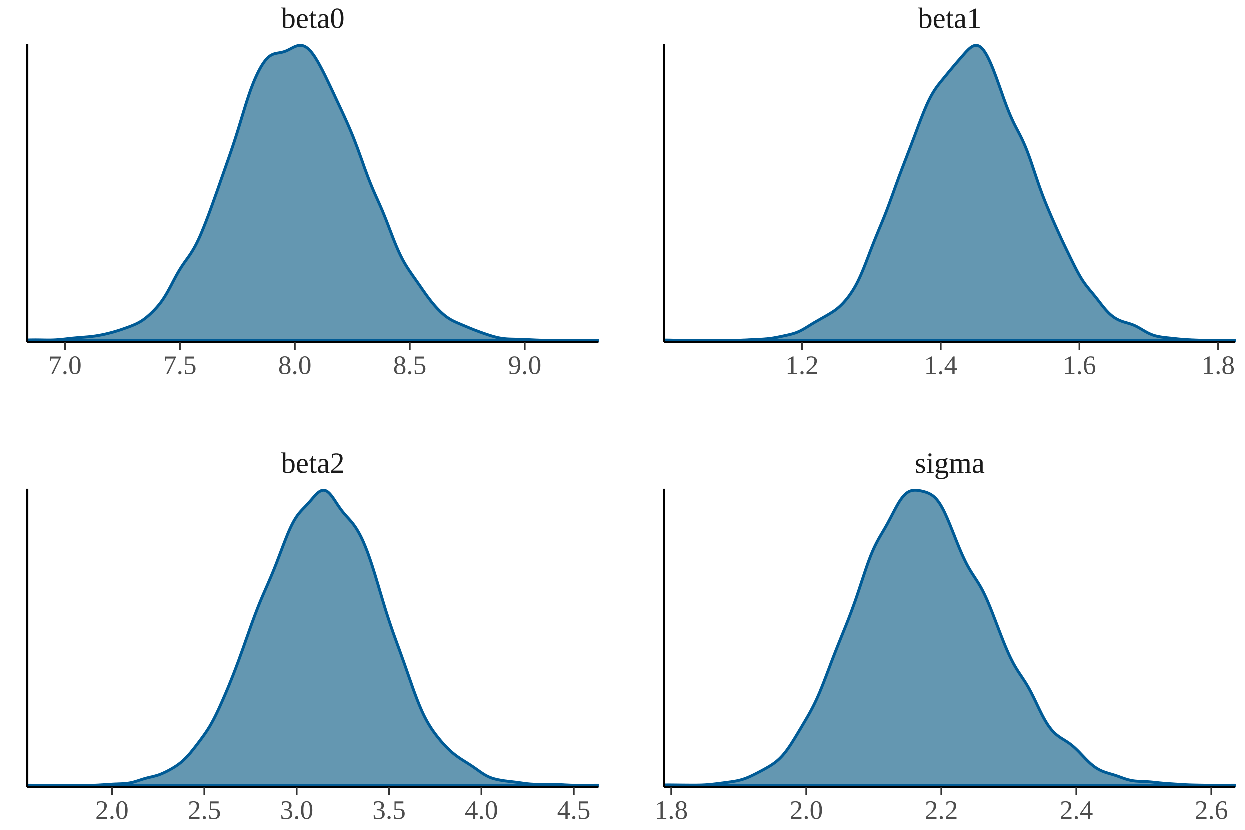
```
# posterior density plot
mcmc_dens(post_mcmc1)
```



```
mcmc_dens(post_mcmc2)
```



```
mcmc_dens(post_mcmc3)
```



```
# convergence diagnostics

# Geweke diagnostics
geweke.diag(posterior$mcmc[[1]])
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  beta0  beta1  beta2  sigma
##  1.310  1.035 -1.090 -0.469
```

```
geweke.diag(posterior$mcmc[[2]])
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   beta0   beta1   beta2   sigma
## -0.5976  0.7949  0.4103 -0.2792
```

```
geweke.diag(posterior$mcmc[[3]])
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     beta0     beta1     beta2     sigma
##  0.544486 -0.803371  0.002283 -1.181925
```

```
# Gelman-Rubin diagnostics
gelman.diag(posterior$mcmc)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1          1
## beta1          1          1
## beta2          1          1
## sigma          1          1
##
## Multivariate psrf
##
## 1
```

```
# effective sample size
effectiveSize(posterior$mcmc[[1]])
```

```
##    beta0    beta1    beta2    sigma
## 4736.556 7630.744 4756.158 8436.253
```

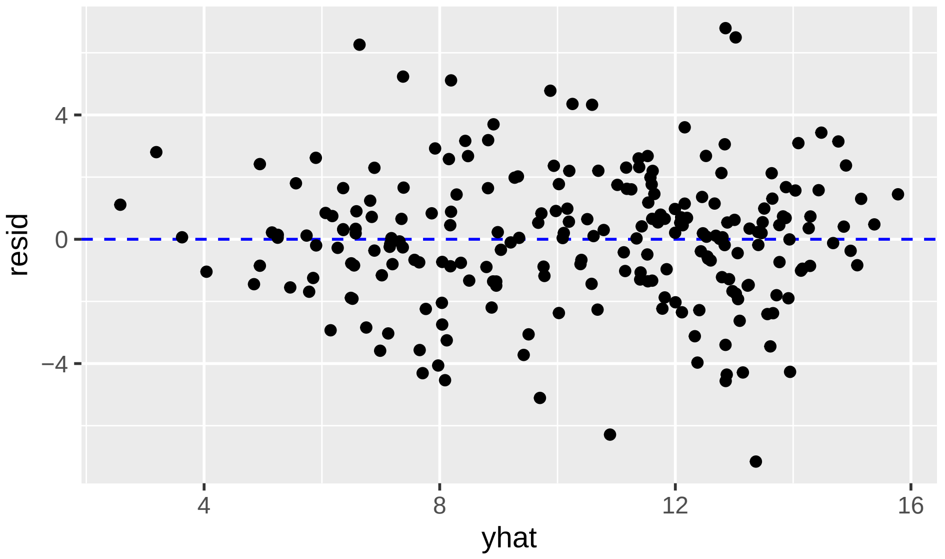
```
effectiveSize(posterior$mcmc[[2]])
```

```
##    beta0    beta1    beta2    sigma
## 4866.770 7040.562 4779.163 8000.000
```

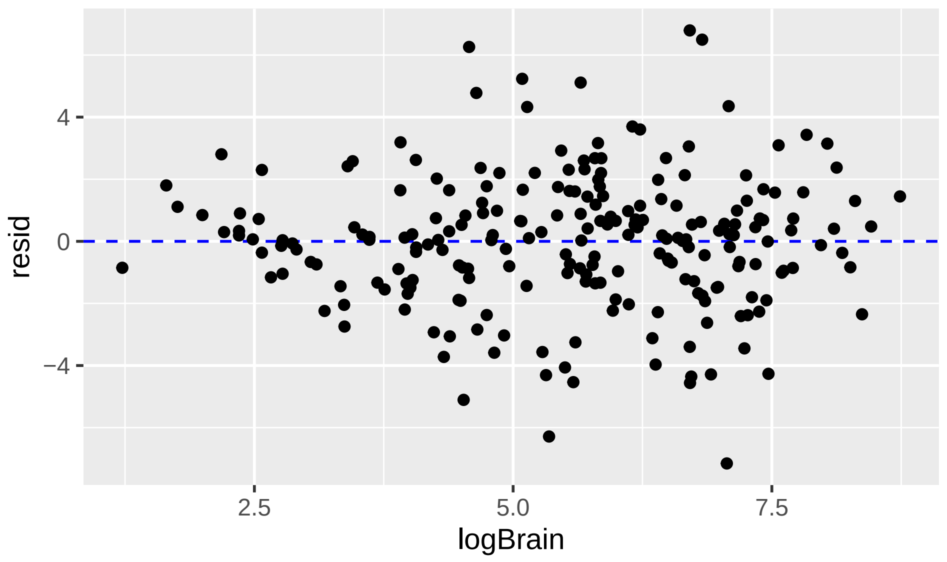
```
effectiveSize(posterior$mcmc[[3]])
```

```
##    beta0    beta1    beta2    sigma
## 4857.370 7634.582 4829.241 8284.047
```

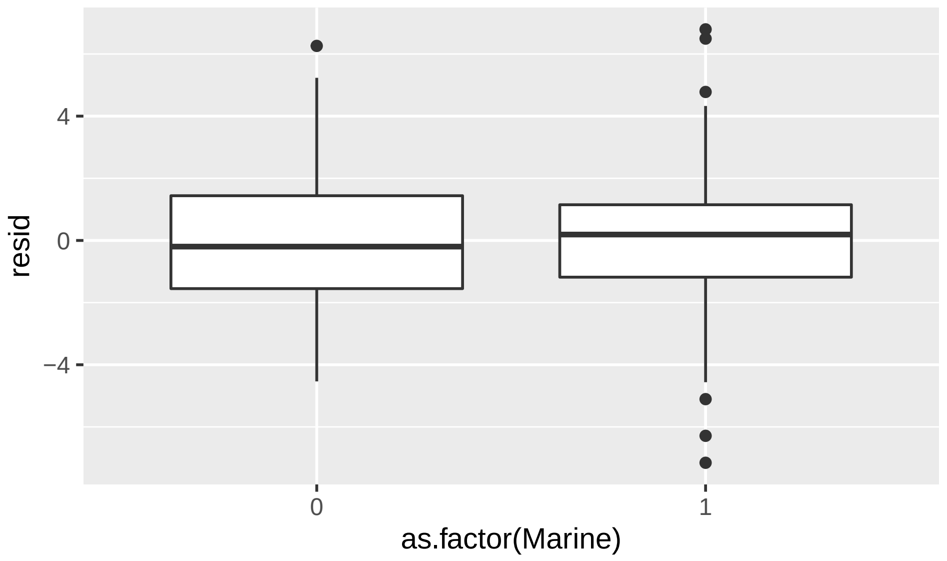
```
# MLR assumptions diagnostics

post_draws <- tidybayes::tidy_draws(posterior$mcmc)

# calculate residuals
resids <- fish_clean %>%
  mutate(
  beta0 = median(post_draws$beta0),
  beta1 = median(post_draws$beta1),
  beta2 = median(post_draws$beta2),
  yhat = beta0 + beta1 * (logBrain-mean(logBrain)) + beta2 * Marine,
  resid = logFecundity - yhat
)

# residuals plot for fitted y
ggplot(data = resids, aes(x = yhat, y = resid)) +
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point()
```



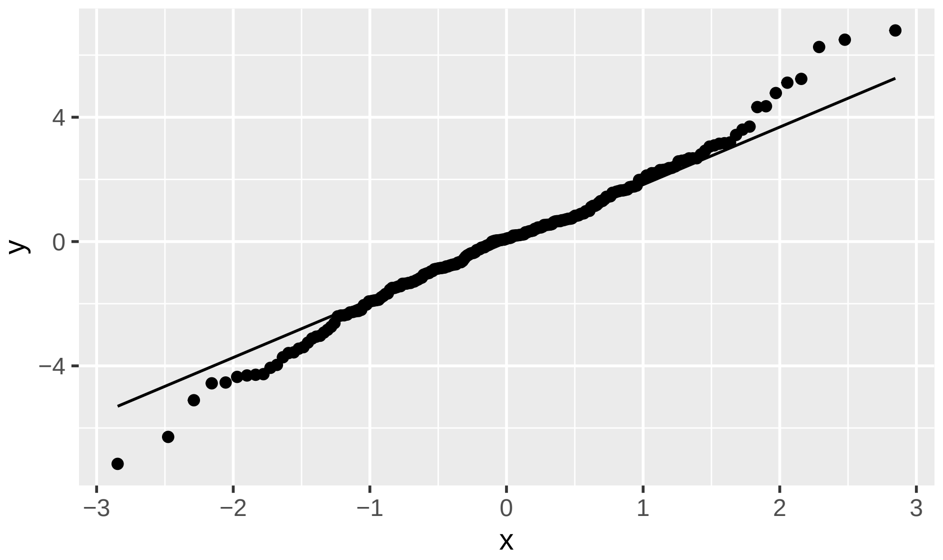
```
# residuals plot for logBrain
ggplot(data = resids, aes(x = logBrain, y = resid)) +
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point()
```



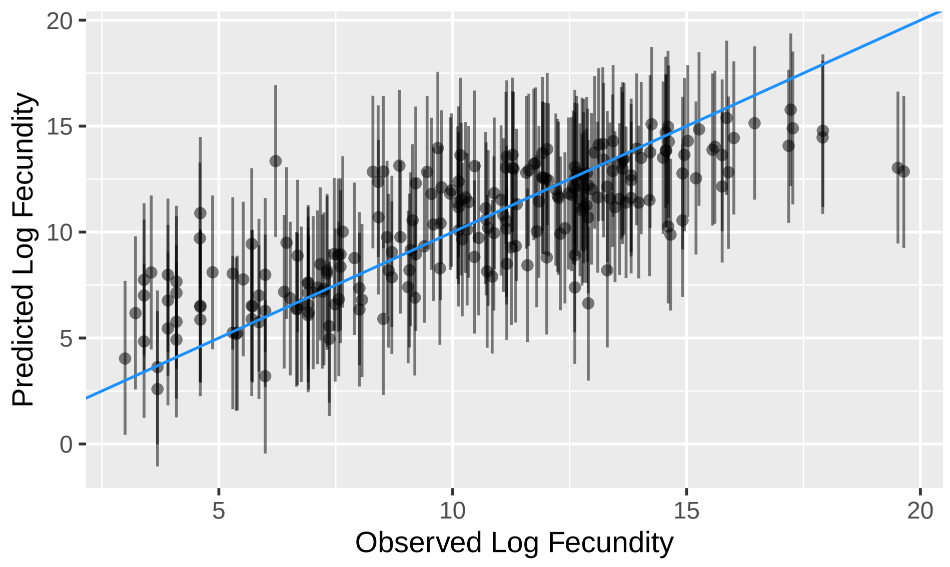
```
# residuals plot for Marine
ggplot(data = resids, aes(x = as.factor(Marine), y = resid)) +
  geom_boxplot()
```



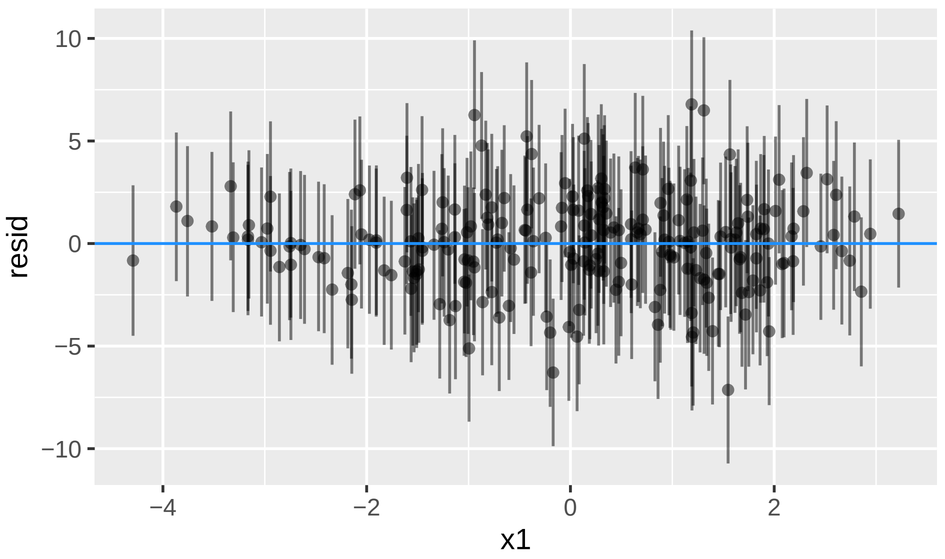
```
# normal QQ plot
ggplot(resids, aes(sample=resid)) +
  stat_qq() +
  stat_qq_line()
```



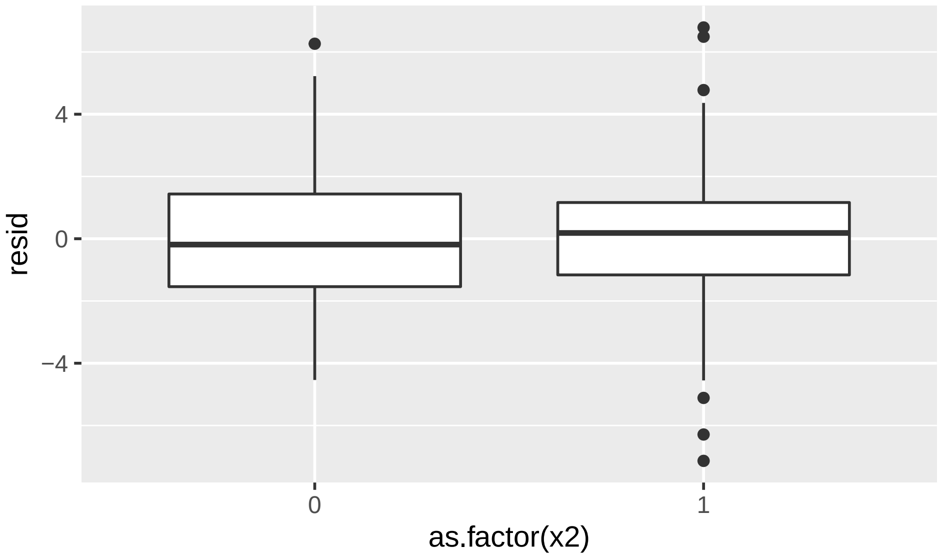
```r
# posterior prediction plot

set.seed(10357)

# calculate mu given x
mu_link <- function(x1, x2) {
  post_draws[["beta0"]] + post_draws[["beta1"]] * x1 + post_draws[["beta2"]] * x2
}

# calculate S mu for each data point
mu_draws <- mapply(mu_link, (fish_clean$logBrain - mean(fish_clean$logBrain)), (fish_clean$Marine))

# simulate one y for each mu in each column
S <- nrow(post_draws)
y_draws <- apply(mu_draws, 2, function(x) rnorm(S, x, post_draws[["sigma"]]))

# calculate the mean and PI for each column
y_means <- colMeans(y_draws)
y_pis   <- apply(y_draws, 2, quantile, probs = c(0.05, 0.95))

post_pred_data <- data.frame(
  y = fish_clean$logFecundity,     # original y
  y_pred = y_means,                # avg. predicted response
  y_lo = y_pis[1,],                # lower bound of predicted response
  y_hi = y_pis[2,]                 # upper bound of predicted response
)

# render plot
ggplot(post_pred_data, aes(x = y)) +
  geom_point(aes(y = y_means), alpha = 0.5) +
  geom_linerange(aes(ymin = y_lo, ymax = y_hi), alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "dodgerblue") +
  labs(x="Observed Log Fecundity", y="Predicted Log Fecundity")
```



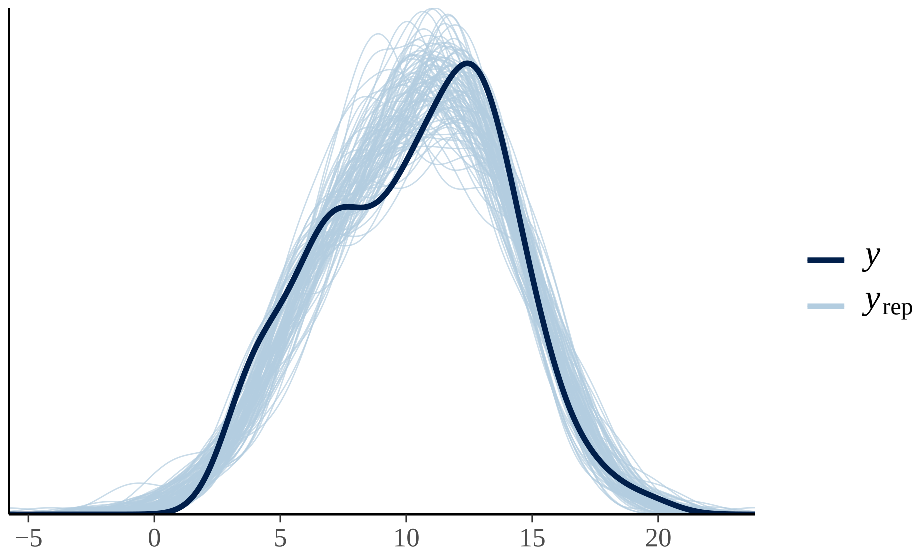
```
# predictive residuals plot

pred_resids <- post_pred_data %>%
  mutate(
    resid = y - y_pred,
    resid_lo = y - y_lo,
    resid_hi = y - y_hi,
    x1 = jitter((fish_clean$logBrain - mean(fish_clean$logBrain)), factor = 2.5), # avoiding some overl
    x2 = fish_clean$Marine
)

# predictive residuals plot for logBrain
ggplot(pred_resids, aes(x = x1)) +
  geom_point(aes(y = resid), alpha = 0.5) +
  geom_linerange(aes(ymin = resid_lo, ymax = resid_hi), alpha = 0.5) +
  geom_hline(yintercept = 0, color = "dodgerblue")
```



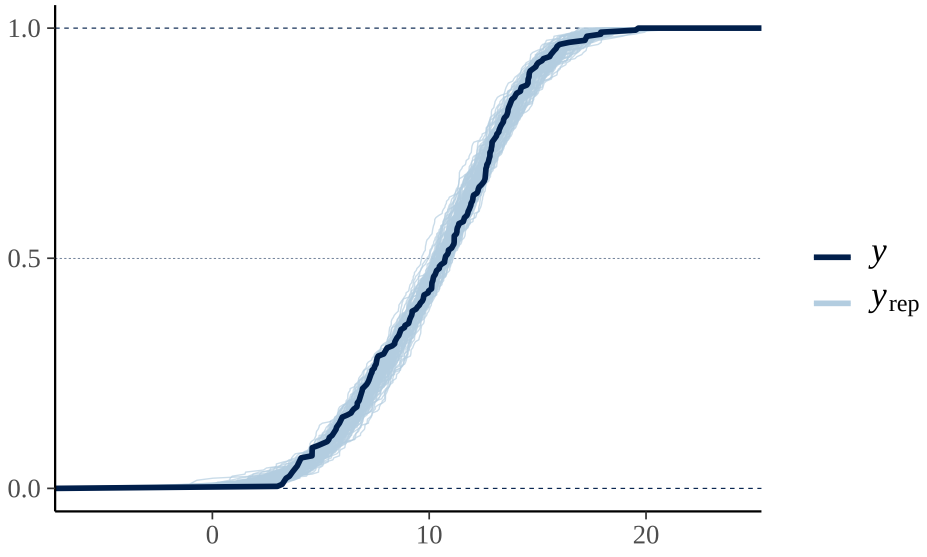
```
# predictive residuals plot for Marine
ggplot(pred_resids, aes(x = as.factor(x2))) +
  geom_boxplot(aes(y = resid))
```



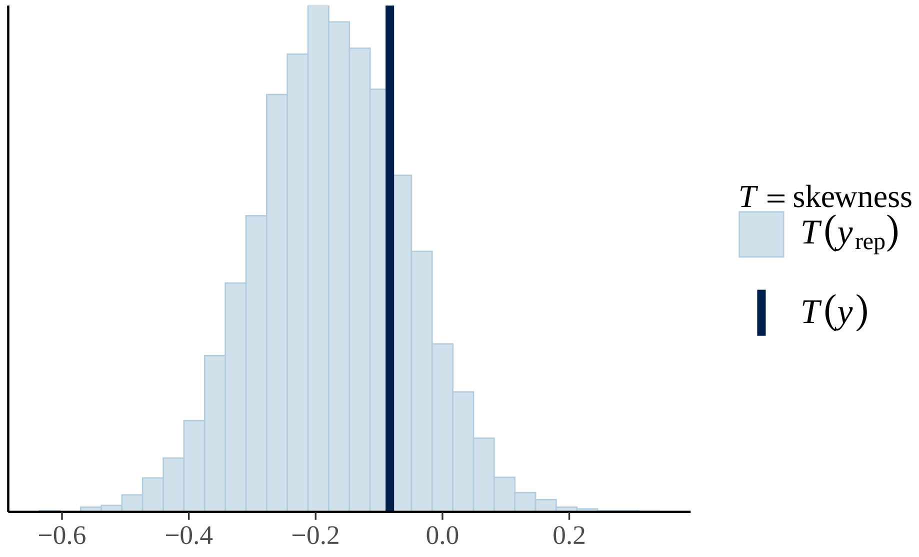
```
# posterior predictive check

# randomly draw 100 from all the simulated data
set.seed(170357)
c_sample <- sample(1:24000, 100)

# overlayed density curves
ppc_dens_overlay(y = fish_clean$logFecundity,
                 yrep = y_draws[c_sample,])
```



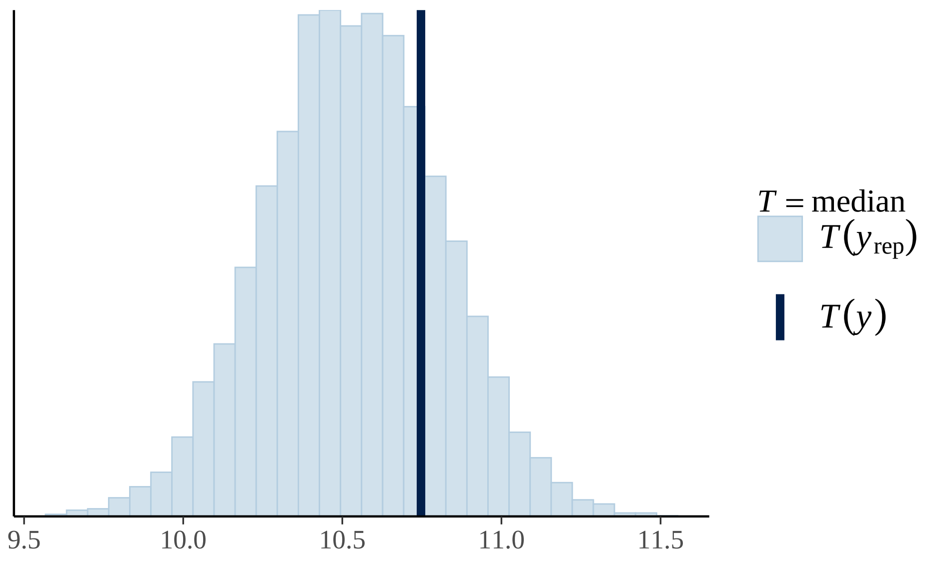
```
# overlayed empirical CDFs
ppc_ecdf_overlay(fish_clean$logFecundity,
                 y_draws[c_sample,])
```



```
# investigating skew
ppc_stat(y = fish_clean$logFecundity,       # observed y
         yrep = y_draws[1:8000,],           # simulated y
         stat = "skewness"                  # name of function
)
```



```
# investigating median
ppc_stat(y = fish_clean$logFecundity,       # observed y
         yrep = y_draws[1:8000,],           # simulated y
         stat = "median"                    # name of function
)
```



3. Test Models

Simple Linear Regression

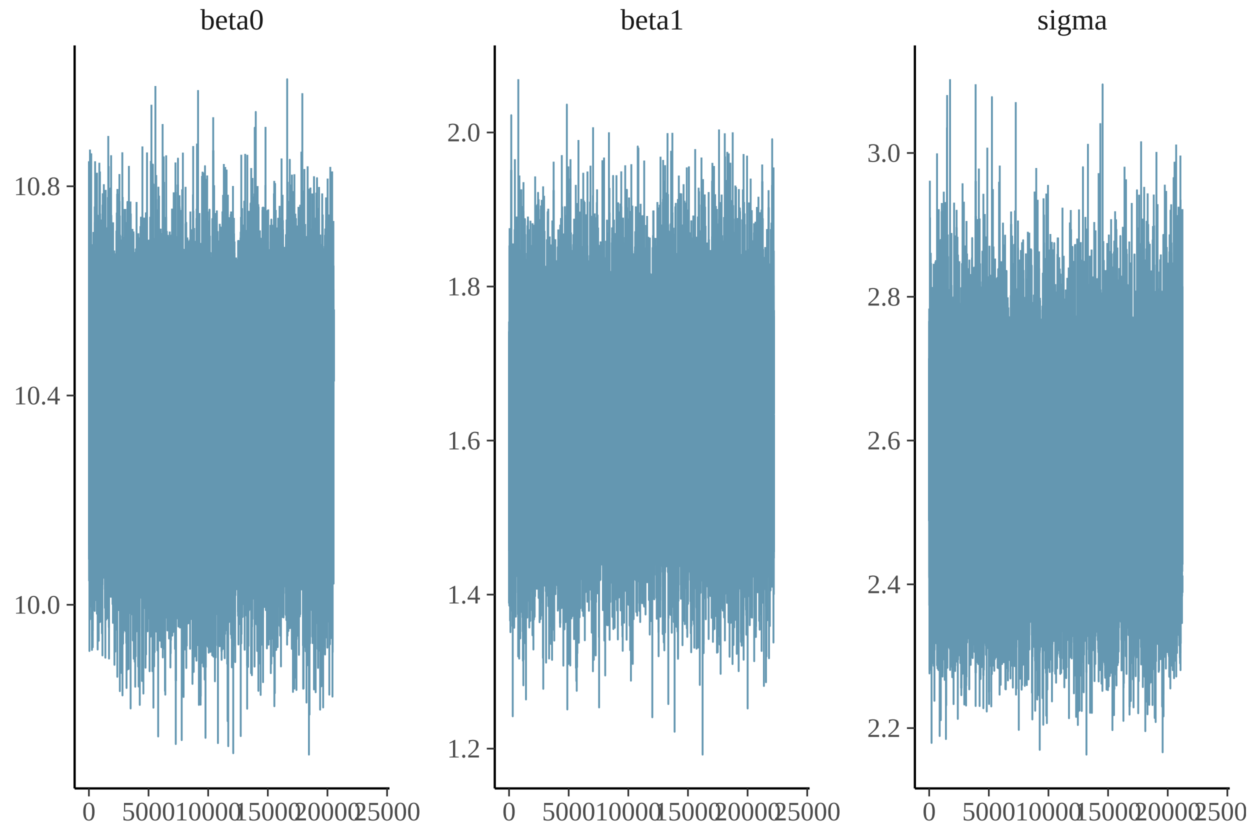
```
# run jags

modelString_test1 <-"
model {
  ## sampling
  for (i in 1:n){
     y[i] ~ dnorm(mu[i], phi)
     mu[i] <- beta0 + beta1 * x[i]
  }

  ## priors
  beta0 ~ dnorm(0, pow(10, -2))
  beta1 ~ dnorm(0, pow(10, -2))
  phi ~ dgamma(1,1)
  sigma <- sqrt(pow(phi ,-1))
}
"

repro_inits_test1 <- list(
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95063432,
       beta0 = 0,  beta1 = 1, phi = 1),
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95073452,
       beta0 = -1, beta1 = 0, phi = 1),
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95086345,
       beta0 = 1, beta1 = -1, phi = 1)
)

fish_data_test <- list(
  n = nrow(fish_clean),
  x = fish_clean$logBrain - mean(fish_clean$logBrain), # centering method
  y = fish_clean$logFecundity
)
```

```r
posterior_test1 <- run.jags(
  modelString_test1,
  data = fish_data_test,
  inits = repro_inits_test1,
  n.chains = 3,
  adapt = 1000,
  burnin = 5000,
  sample = 8000,
  monitor = c("beta0", "beta1", "sigma"),
  silent.jags = TRUE
)
```

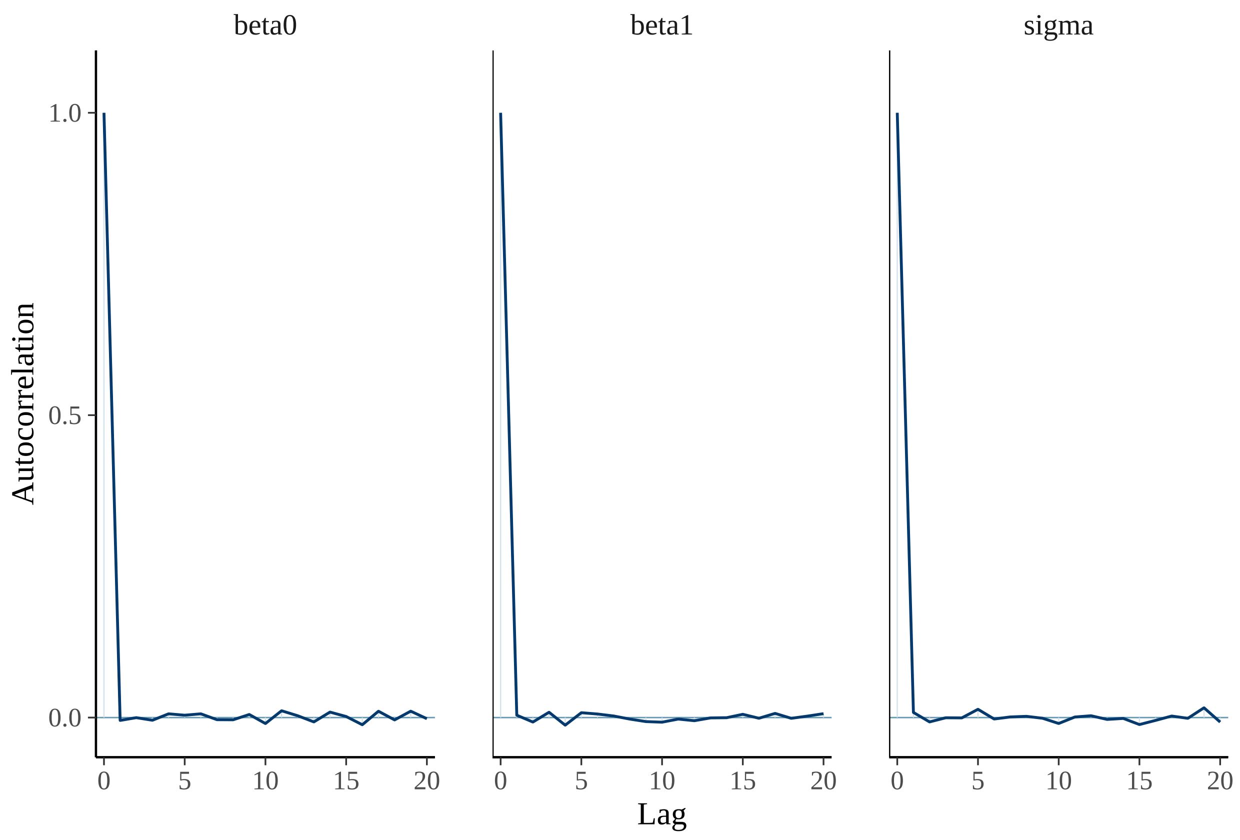
```r
# summary
summary(posterior_test1)
```

```
##        Lower95    Median   Upper95      Mean        SD Mode        MCerr
## beta0 10.02324 10.344424 10.692331 10.343632 0.1703942   NA 0.0010927424
## beta1  1.42258  1.632173  1.841073  1.632100 0.1069391   NA 0.0006906650
## sigma  2.31293  2.544499  2.784278  2.548816 0.1204624   NA 0.0007748662
##       MC%ofSD SSeff        AC.10     psrf
## beta0     0.6 24315 -0.010234030 1.000046
## beta1     0.6 23974 -0.007849681 1.000034
## sigma     0.6 24168 -0.009920752 1.000152
```

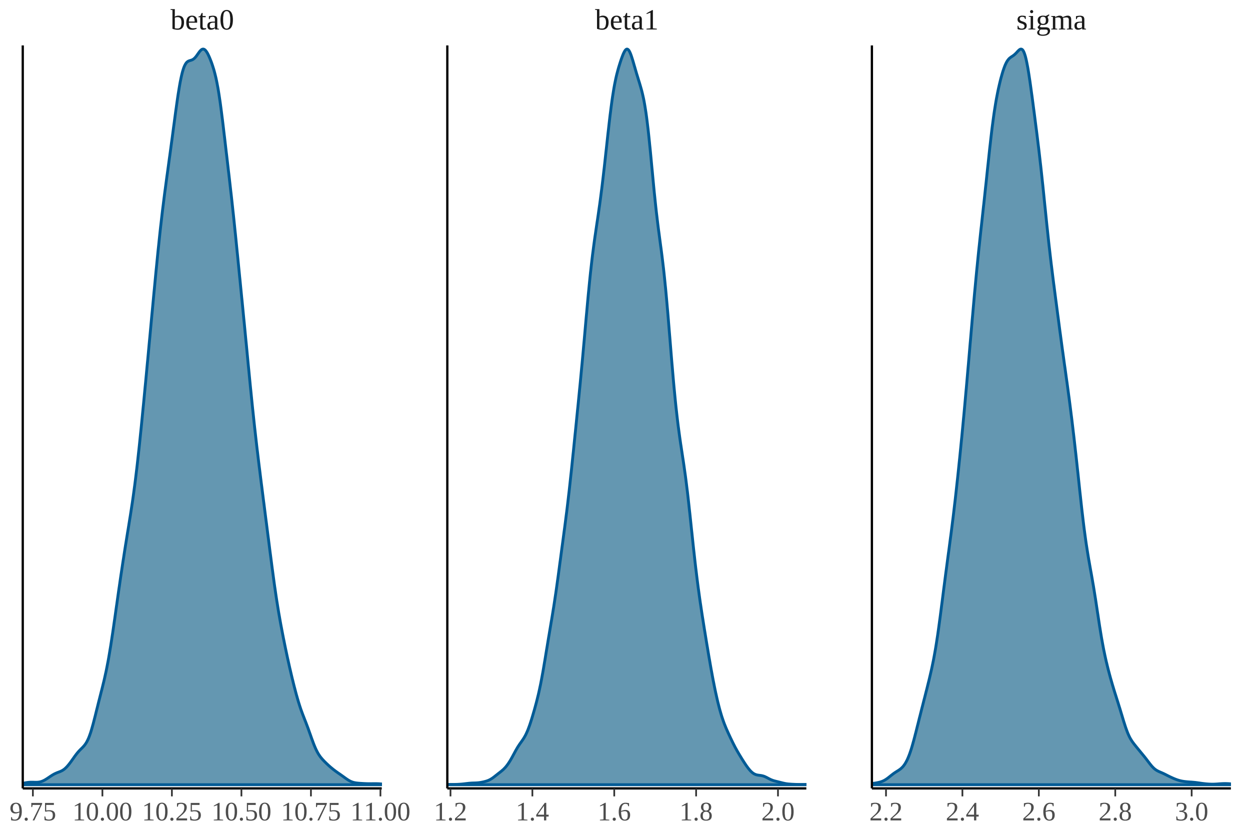
```r
# convergence dsiagnostics

# convert to an mcmc object
post_mcmc_test1 <- as.mcmc(posterior_test1)

# trace plot
mcmc_trace(post_mcmc_test1)
```



```
# ACF plot
mcmc_acf(post_mcmc_test1)
```



```
# posterior density plot
mcmc_dens(post_mcmc_test1)
```



```
# convergence diagnostics

# Geweke diagnostics
geweke.diag(posterior_test1)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   beta0   beta1   sigma
## 1.59870 0.04736 1.20729
```

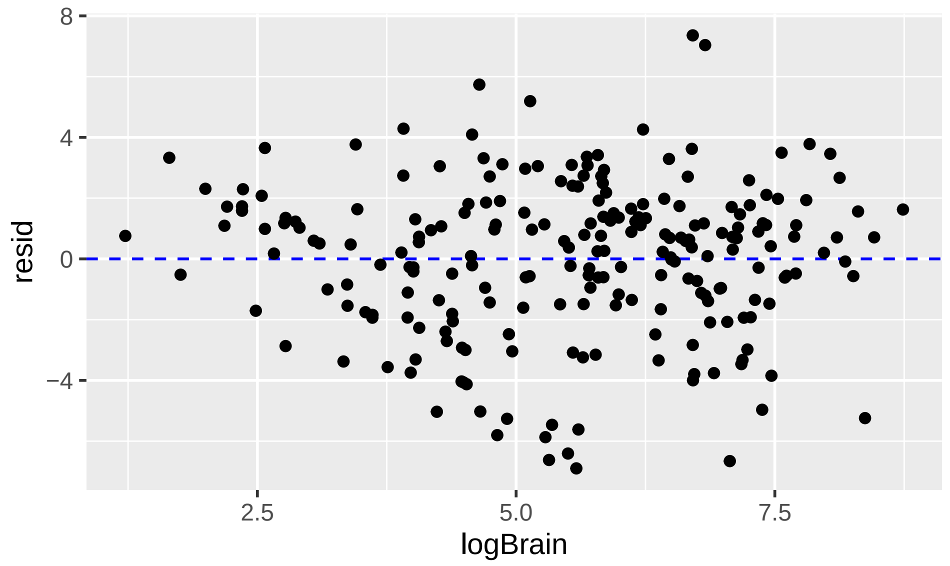
```
# Gelman-Rubin diagnostics
gelman.diag(posterior_test1)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1          1
## beta1          1          1
## sigma          1          1
##
## Multivariate psrf
##
## 1
```

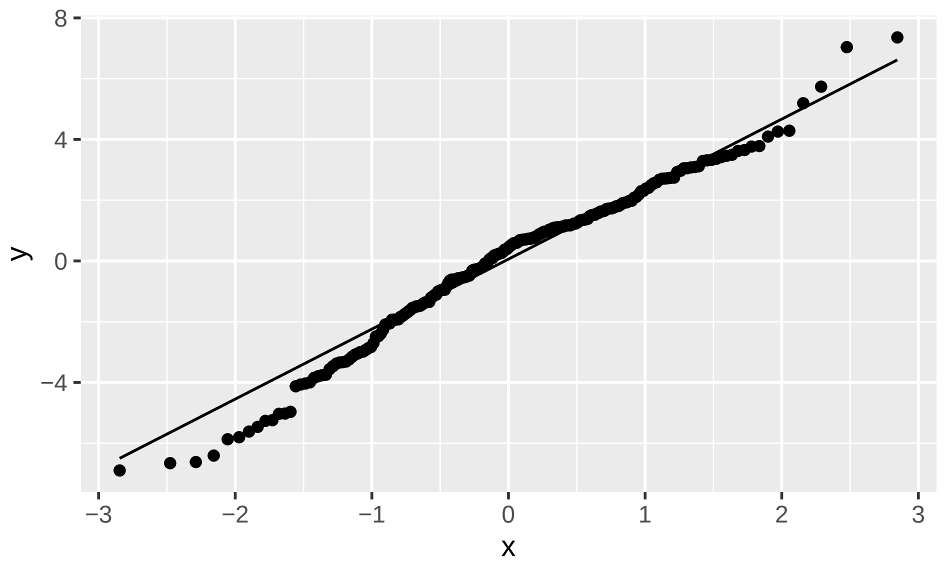
```
# effective sample size
effectiveSize(posterior_test1)
```

```
## beta0 beta1 sigma
## 24000 24000 24000
```

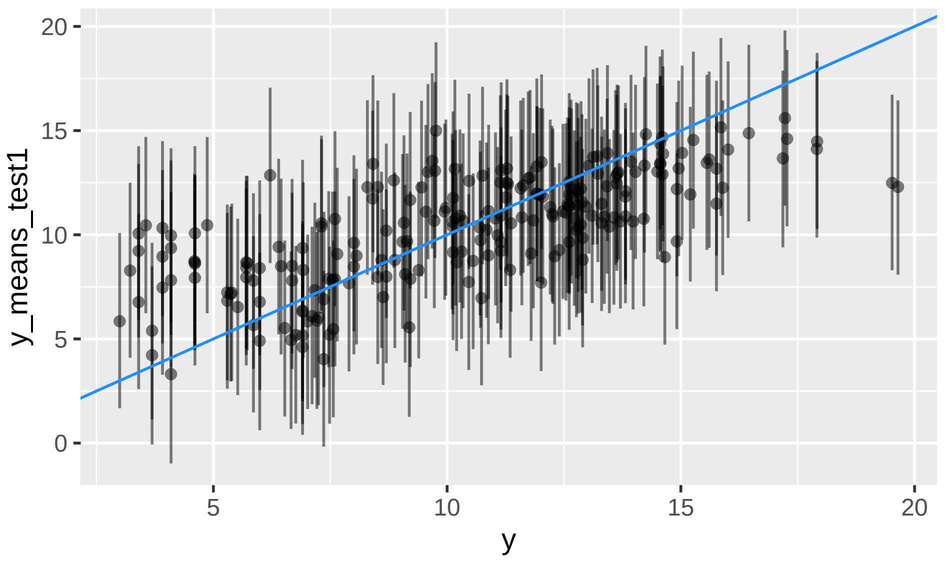
```
# SLR assumptions diagnostics

post_draws_test1 <- tidybayes::tidy_draws(posterior_test1$mcmc)

# calculate residuals
resids_test1 <- fish_clean %>%
  mutate(
  beta0 = median(post_draws_test1$beta0),
  beta1 = median(post_draws_test1$beta1),
  yhat = beta0 + beta1 * (logBrain-mean(logBrain)),
  resid = logFecundity - yhat
)

# residuals plot for logBrain
ggplot(data = resids_test1, aes(x = logBrain, y = resid)) +
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point()
```



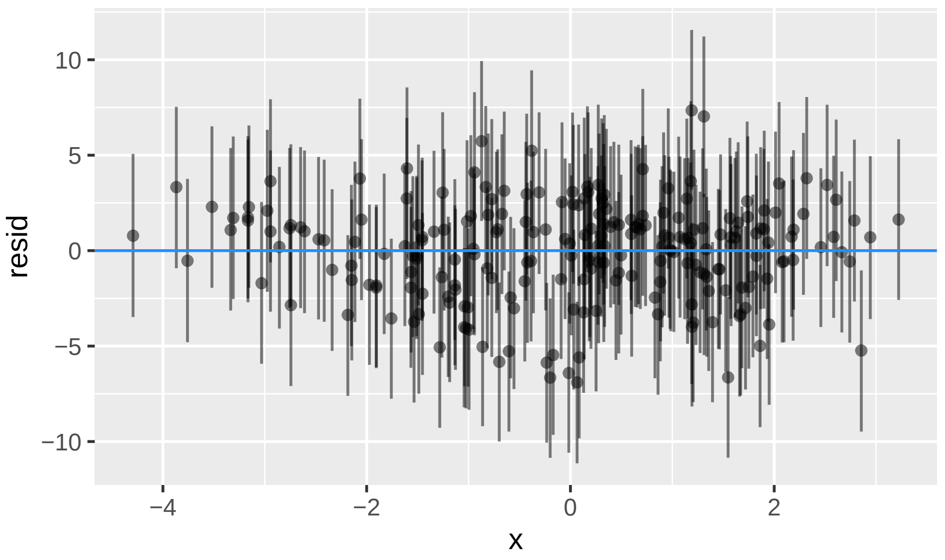
```
# normal QQ plot
ggplot(resids_test1, aes(sample=resid)) +
  stat_qq() +
  stat_qq_line()
```



```
# posterior prediction plot

set.seed(10357)

# calculate mu given x
mu_link_test1 <- function(x) {
  post_draws_test1[["beta0"]] + post_draws_test1[["beta1"]] * x
}

# calculate S mu for each data point
mu_draws_test1 <- sapply((fish_clean$logBrain - mean(fish_clean$logBrain)), mu_link_test1)

# simulate one y for each mu in each column
S_test1 <- nrow(post_draws_test1)
y_draws_test1 <- apply(mu_draws_test1, 2, function(x) rnorm(S_test1, x, post_draws_test1[["sigma"]]))

# calculate the mean and PI for each column
y_means_test1 <- colMeans(y_draws_test1)
y_pis_test1   <- apply(y_draws_test1, 2, quantile, probs = c(0.05, 0.95))

post_pred_data_test1 <- data.frame(
  y = fish_clean$logFecundity,      # original y
  y_pred = y_means_test1,           # avg. predicted response
  y_lo = y_pis_test1[1,],           # lower bound of predicted response
  y_hi = y_pis_test1[2,]            # upper bound of predicted response
)

# render plot
ggplot(post_pred_data_test1, aes(x = y)) +
  geom_point(aes(y = y_means_test1), alpha = 0.5) +
  geom_linerange(aes(ymin = y_lo, ymax = y_hi), alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "dodgerblue")
```

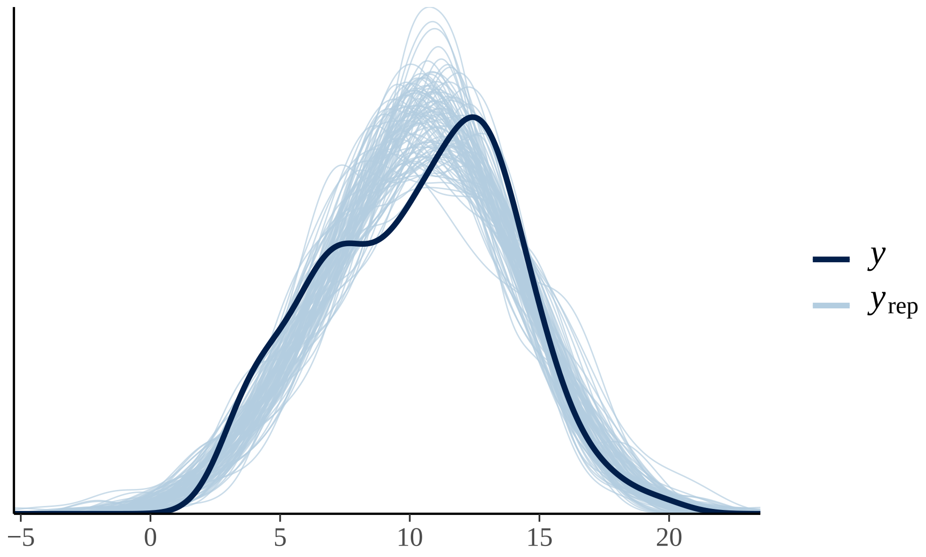


```
# predictive residuals plot

pred_resids_test1 <- post_pred_data_test1 %>%
  mutate(
    resid = y - y_pred,
    resid_lo = y - y_lo,
    resid_hi = y - y_hi,
    x = jitter((fish_clean$logBrain - mean(fish_clean$logBrain)), factor = 2.5), # avoiding some overla
)

# predictive residuals plot for logBrain
ggplot(pred_resids_test1, aes(x = x)) +
  geom_point(aes(y = resid), alpha = 0.5) +
  geom_linerange(aes(ymin = resid_lo, ymax = resid_hi), alpha = 0.5) +
  geom_hline(yintercept = 0, color = "dodgerblue")
```

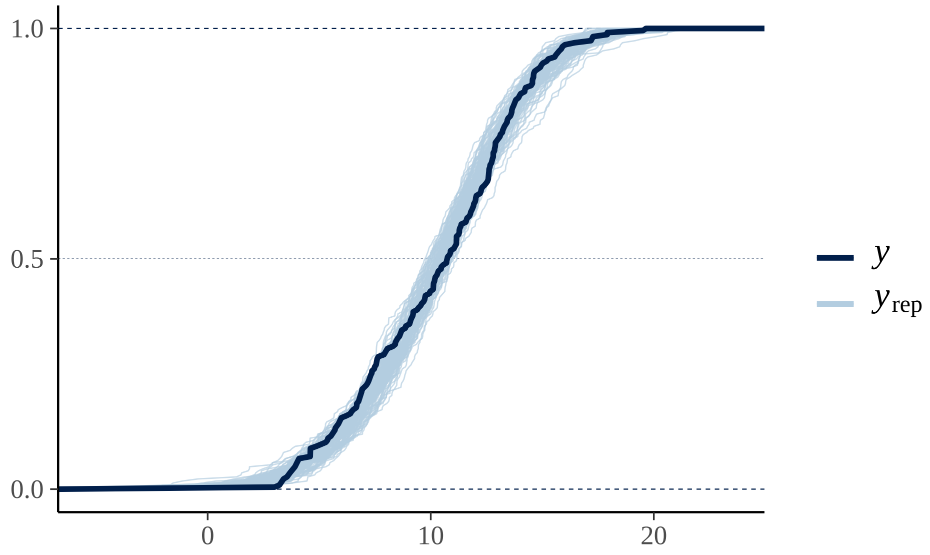


```
# posterior predictive check

# randomly draw 100 from all the simulated data
```

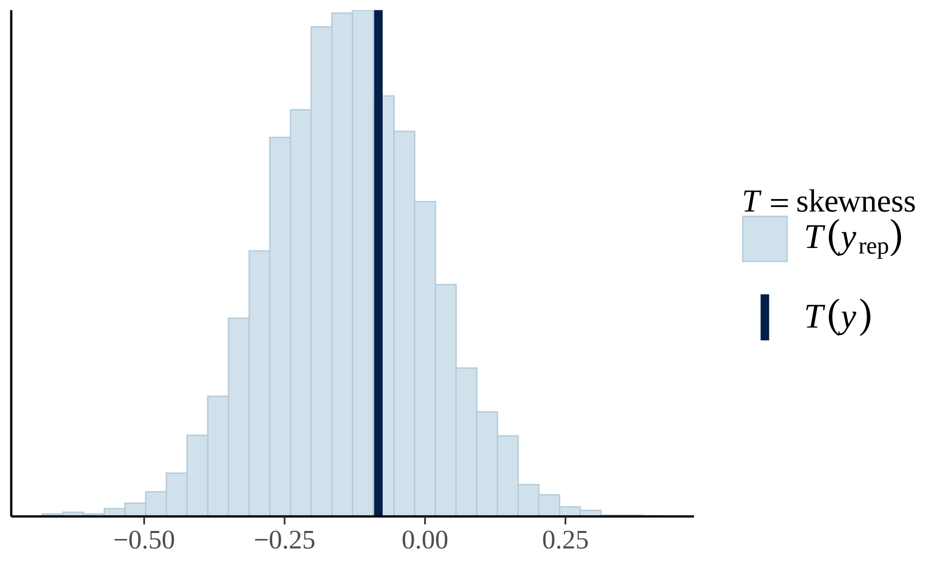
```r
set.seed(170357)
c_sample <- sample(1:24000, 100)

# overlayed density curves
ppc_dens_overlay(y = fish_clean$logFecundity,
                 yrep = y_draws_test1[c_sample,])
```



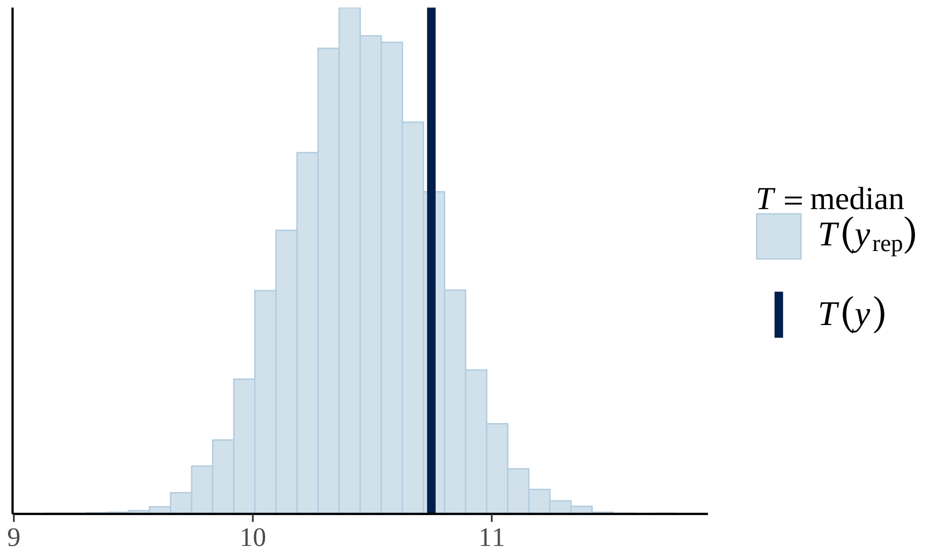
```r
# overlayed empirical CDFs
ppc_ecdf_overlay(fish_clean$logFecundity,
                 y_draws_test1[c_sample,])
```



```r
# investigating skew
ppc_stat(y = fish_clean$logFecundity,      # observed y
         yrep = y_draws_test1[1:8000,],    # simulated y
         stat = "skewness"                 # name of function
)
```



```
# investigating median
ppc_stat(y = fish_clean$logFecundity,       # observed y
         yrep = y_draws_test1[1:8000,],     # simulated y
         stat = "median"                    # name of function
)
```



Polynomial Regression

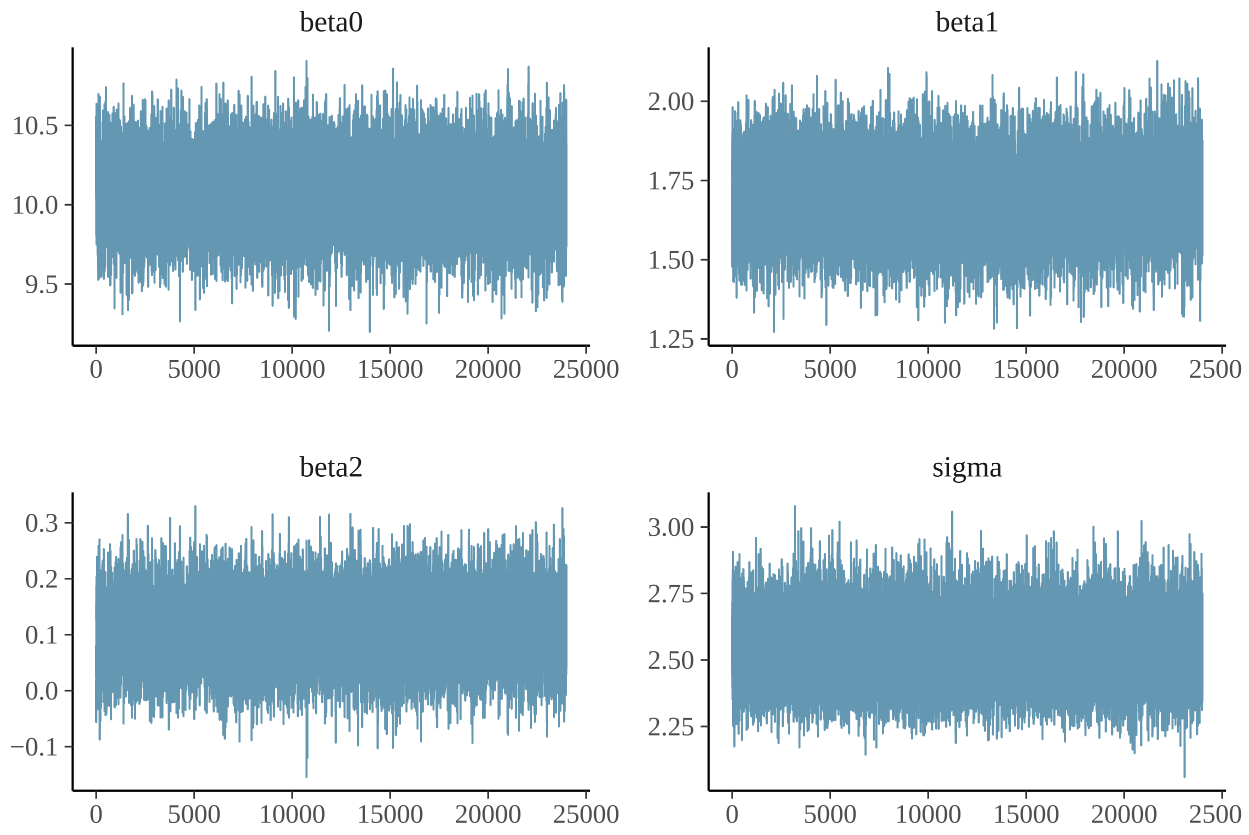
```
# run jags

modelString_test2 <-"
model {
## sampling
for (i in 1:n){
   y[i] ~ dnorm(mu[i], phi)
   mu[i] <- beta0 + beta1 * x[i] + beta2 * pow(x[i], 2)
}
## priors
beta0 ~ dnorm(0, pow(10, -2))
beta1 ~ dnorm(0, pow(10, -2))
beta2 ~ dnorm(0, pow(10, -2))
phi ~ dgamma(1,1)
sigma <- sqrt(pow(phi ,-1))
}
"

repro_inits_test2 <- list(
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95063435,
       beta0 = 0,  beta1 = 1, beta2 = -1, phi = 1),
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95074029,
       beta0 = -1, beta1 = 0, beta2 = 1, phi = 1),
  list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 95086349,
       beta0 = 1, beta1 = -1, beta2 = 0, phi = 1)
)

fish_data_test <- list(
  n = nrow(fish_clean),
  x = fish_clean$logBrain - mean(fish_clean$logBrain), # centering method
  y = fish_clean$logFecundity
)

posterior_test2 <- run.jags(
  modelString_test2,
  data = fish_data_test,
  inits = repro_inits_test2,
  n.chains = 3,
  adapt = 1000,
  burnin = 5000,
  sample = 8000,
  monitor = c("beta0", "beta1", "beta2", "sigma"),
  silent.jags = TRUE
)
```

```
# summary
summary(posterior_test2)
```

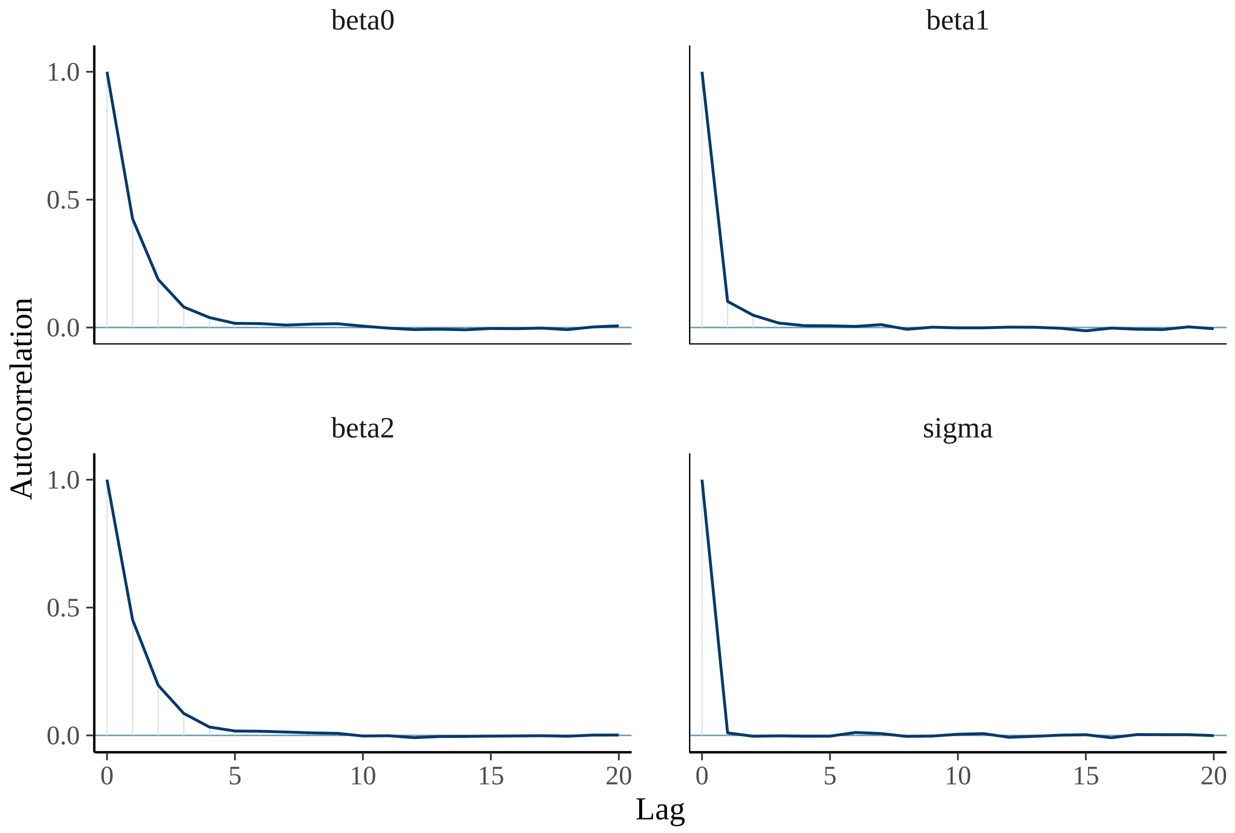
```
##             Lower95     Median    Upper95       Mean         SD Mode
## beta0   9.641764450 10.0693183 10.5193687 10.0700725 0.22398334   NA
## beta1   1.477320242  1.7007853  1.9140907  1.7002472 0.11207995   NA
## beta2 -0.005372704  0.1090232  0.2199391  0.1086761 0.05743789   NA
## sigma   2.306067919  2.5295095  2.7750915  2.5341113 0.12045843   NA
##               MCerr MC%ofSD SSeff        AC.10      psrf
## beta0 0.0023396792     1.0  9165  0.005604557 1.0001426
## beta1 0.0008324736     0.7 18127 -0.001505015 1.0000122
## beta2 0.0006060548     1.1  8982 -0.002368720 1.0005216
## sigma 0.0007775558     0.6 24000  0.004961971 0.9999651
```

```
# convergence dsiagnostics

# convert to an mcmc object
post_mcmc_test2 <- as.mcmc(posterior_test2)
```

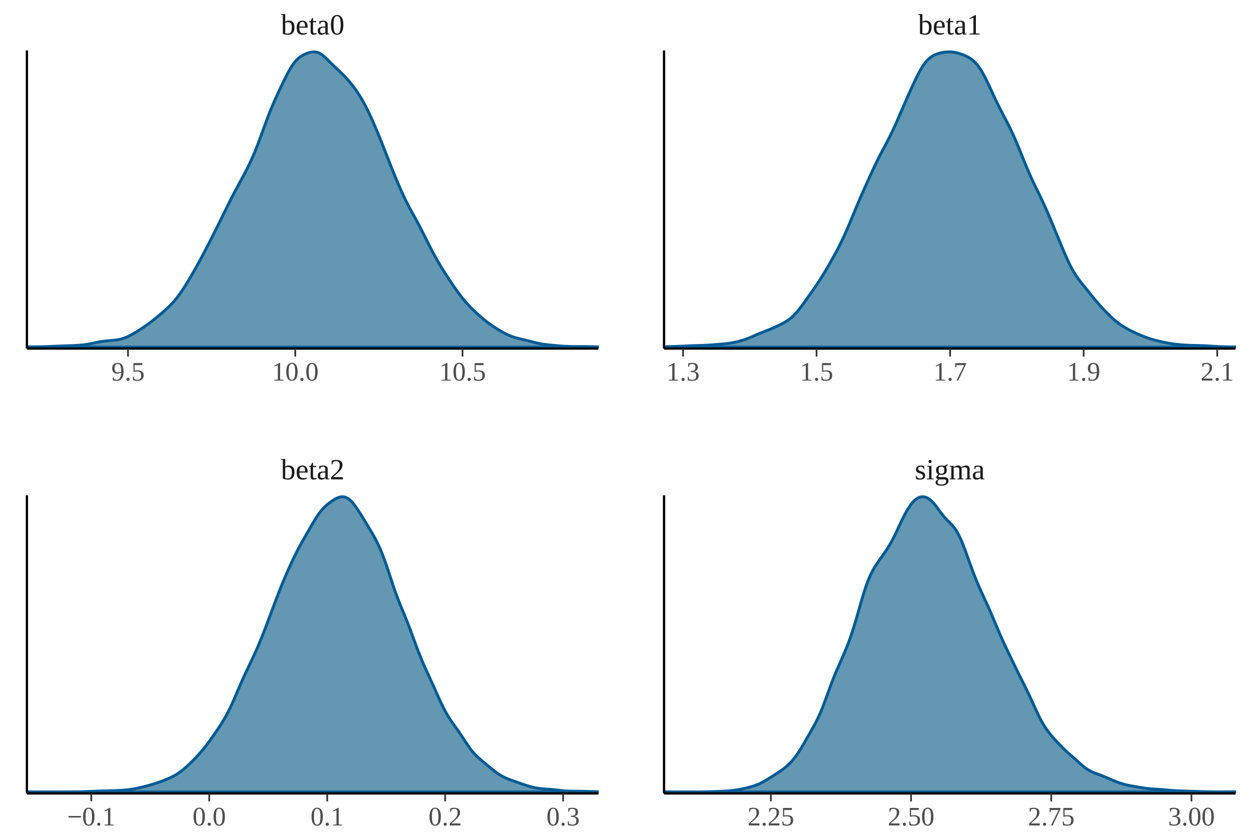
```
# trace plot
mcmc_trace(post_mcmc_test2)
```



```
# ACF plot
mcmc_acf(post_mcmc_test2)
```



```r
# posterior density plot
mcmc_dens(post_mcmc_test2)
```



```
# convergence diagnostics

# Geweke diagnostics
geweke.diag(posterior_test2)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    beta0    beta1    beta2    sigma
##  0.03682 -0.30305 -0.11113 -1.81497
```

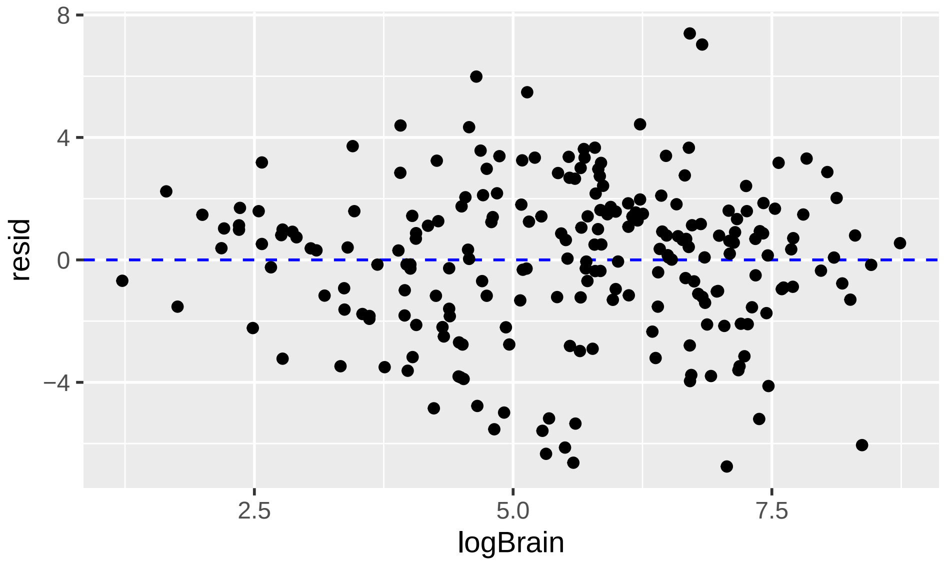
```
# Gelman-Rubin diagnostics
gelman.diag(posterior_test2)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1          1
## beta1          1          1
## beta2          1          1
## sigma          1          1
##
## Multivariate psrf
##
## 1
```

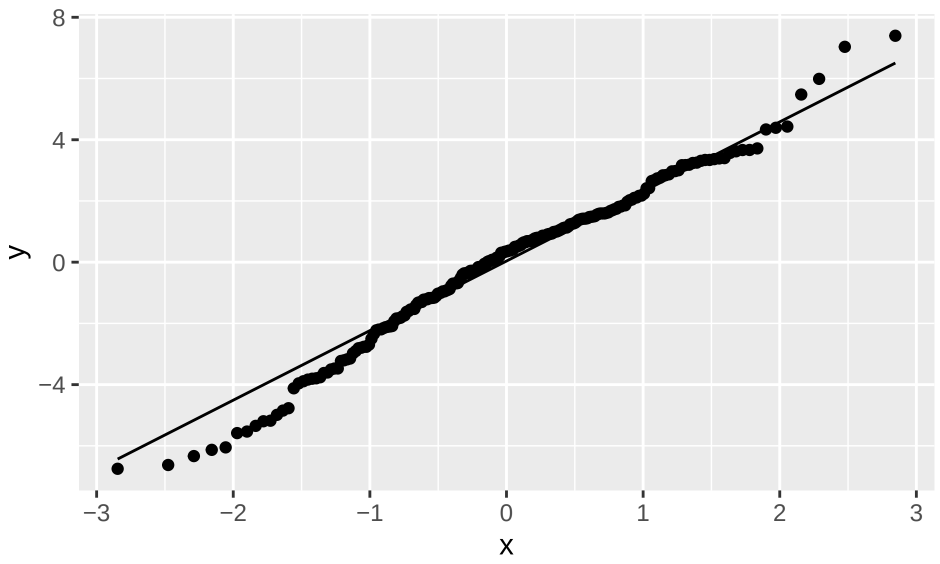
```
# effective sample size
effectiveSize(posterior_test2)
```

```
##     beta0     beta1     beta2     sigma
##  9685.538 18124.946  9245.743 23524.629
```

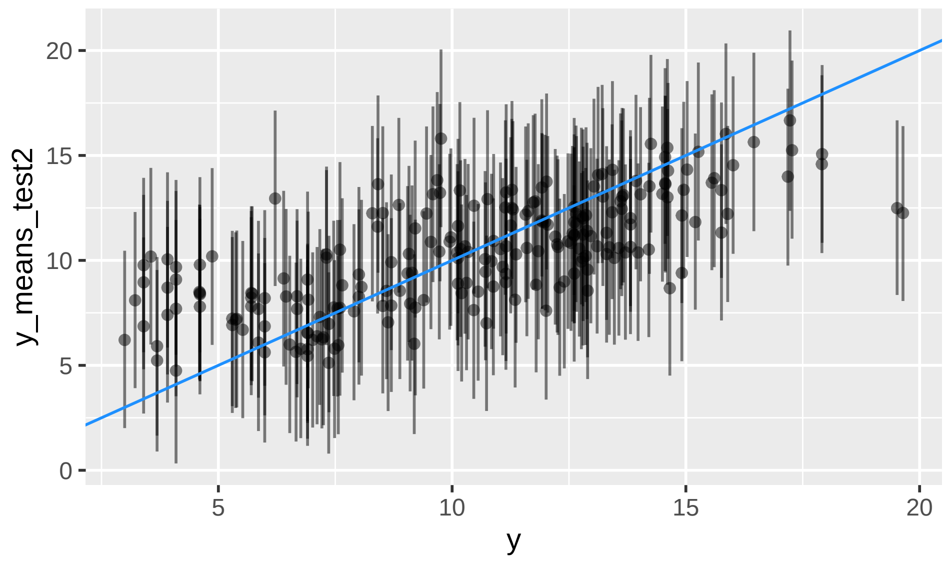
```
# assumptions diagnostics

post_draws_test2 <- tidybayes::tidy_draws(posterior_test2$mcmc)

# calculate residuals
resids_test2 <- fish_clean %>%
  mutate(
  beta0 = median(post_draws_test2$beta0),
  beta1 = median(post_draws_test2$beta1),
  beta2 = median(post_draws_test2$beta2),
  yhat = beta0 + beta1 * (logBrain-mean(logBrain)) + beta2 * (logBrain-mean(logBrain))^2,
  resid = logFecundity - yhat
)

# residuals plot for logBrain
ggplot(data = resids_test2, aes(x = logBrain, y = resid)) +
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point()
```



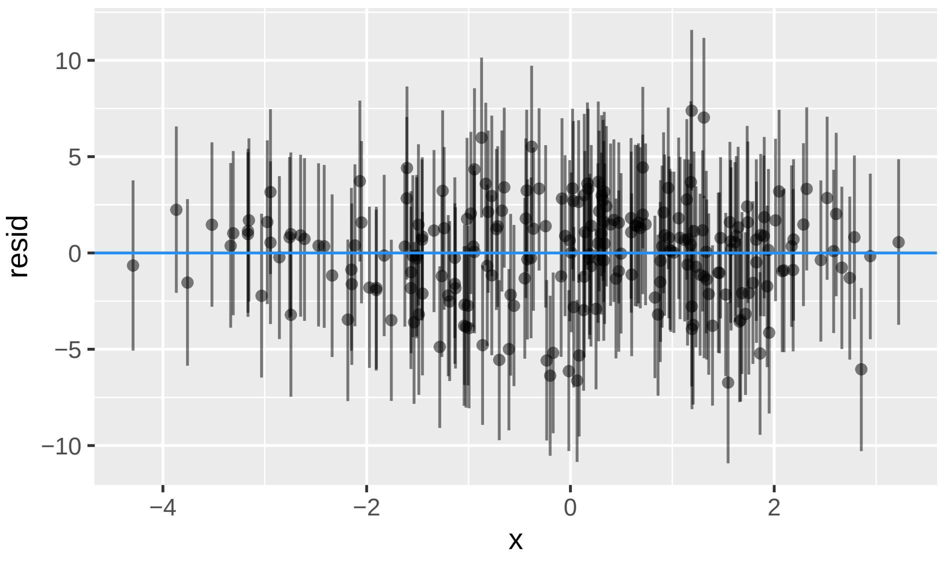
```
# normal QQ plot
ggplot(resids_test2, aes(sample=resid)) +
  stat_qq() +
  stat_qq_line()
```



```r
# posterior prediction plot

set.seed(10357)

# calculate mu given x
mu_link_test2 <- function(x) {
  post_draws_test2[["beta0"]] + post_draws_test2[["beta1"]] * x + post_draws_test2[["beta2"]] * x^2
}

# calculate S mu for each data point
mu_draws_test2 <- sapply((fish_clean$logBrain - mean(fish_clean$logBrain)), mu_link_test2)

# simulate one y for each mu in each column
S_test2 <- nrow(post_draws_test2)
y_draws_test2 <- apply(mu_draws_test2, 2, function(x) rnorm(S_test2, x, post_draws_test2[["sigma"]]))

# calculate the mean and PI for each column
y_means_test2 <- colMeans(y_draws_test2)
y_pis_test2   <- apply(y_draws_test2, 2, quantile, probs = c(0.05, 0.95))

post_pred_data_test2 <- data.frame(
  y = fish_clean$logFecundity,      # original y
  y_pred = y_means_test2,           # avg. predicted response
  y_lo = y_pis_test2[1,],           # lower bound of predicted response
  y_hi = y_pis_test2[2,]            # upper bound of predicted response
)

# render plot
ggplot(post_pred_data_test2, aes(x = y)) +
  geom_point(aes(y = y_means_test2), alpha = 0.5) +
  geom_linerange(aes(ymin = y_lo, ymax = y_hi), alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "dodgerblue")
```

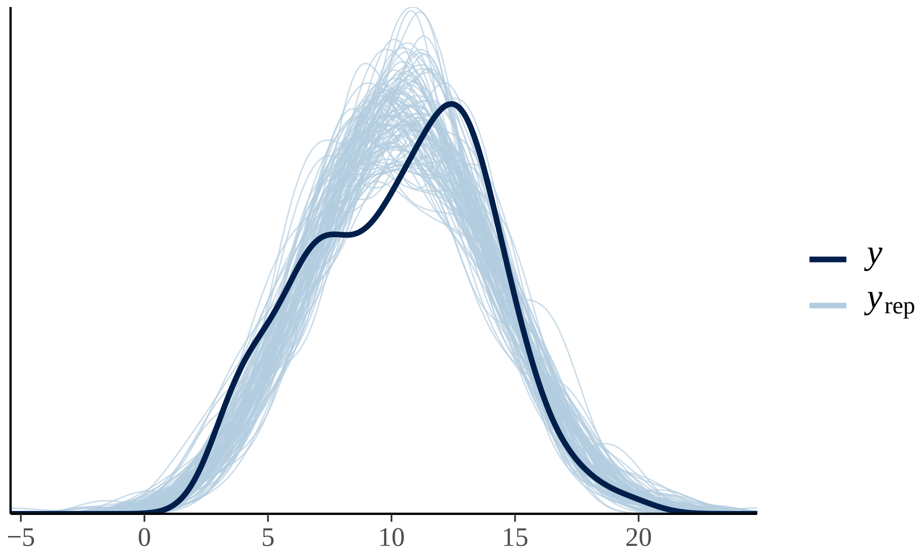


```
# predictive residuals plot

pred_resids_test2 <- post_pred_data_test2 %>%
  mutate(
    resid = y - y_pred,
    resid_lo = y - y_lo,
    resid_hi = y - y_hi,
    x = jitter((fish_clean$logBrain - mean(fish_clean$logBrain)), factor = 2.5), # avoiding some overla
)

# predictive residuals plot for logBrain
ggplot(pred_resids_test2, aes(x = x)) +
  geom_point(aes(y = resid), alpha = 0.5) +
  geom_linerange(aes(ymin = resid_lo, ymax = resid_hi), alpha = 0.5) +
  geom_hline(yintercept = 0, color = "dodgerblue")
```

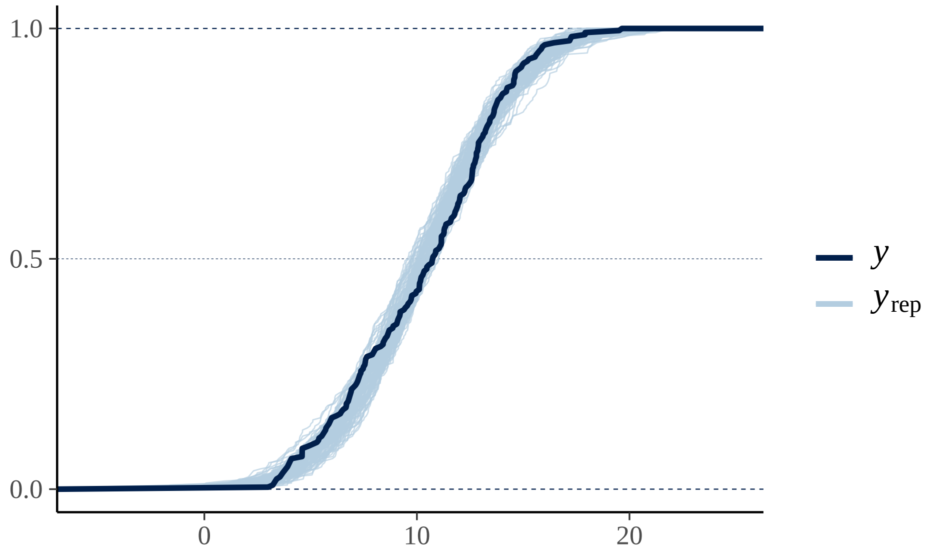


```
# posterior predictive check

# randomly draw 100 from all the simulated data
```

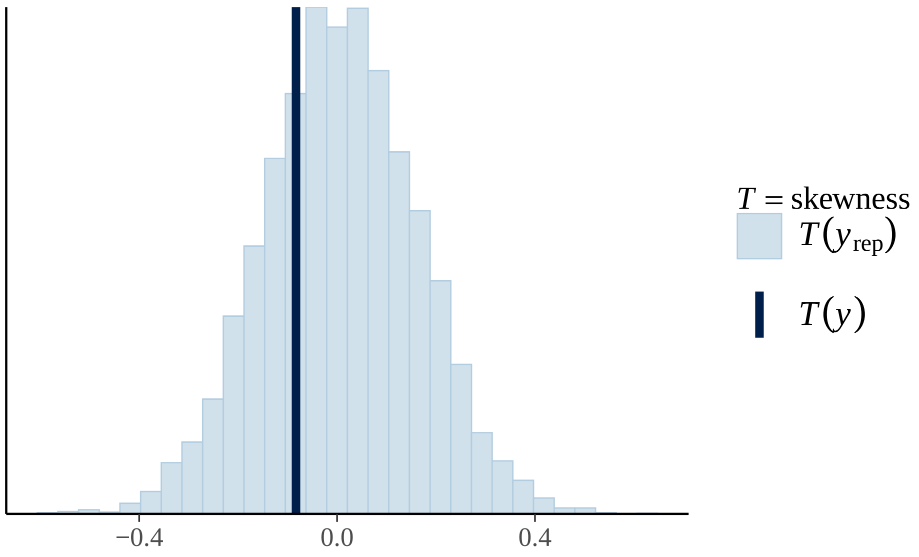
```
set.seed(170357)
c_sample <- sample(1:24000, 100)

# overlayed density curves
ppc_dens_overlay(y = fish_clean$logFecundity,
                 yrep = y_draws_test2[c_sample,])
```



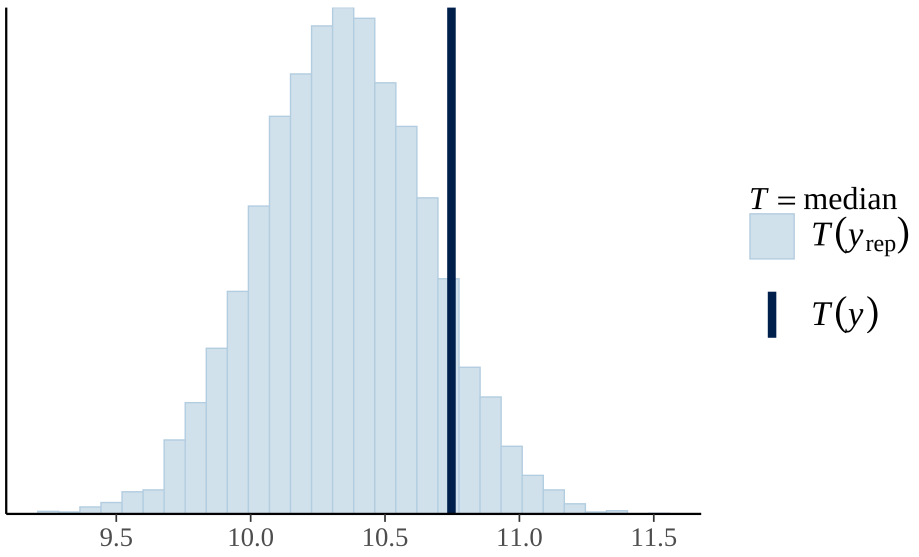
```
# overlayed empirical CDFs
ppc_ecdf_overlay(fish_clean$logFecundity,
                 y_draws_test2[c_sample,])
```



```
# investigating skew
ppc_stat(y = fish_clean$logFecundity,      # observed y
         yrep = y_draws_test2[1:8000,],    # simulated y
         stat = "skewness"                 # name of function
)
```



```
# investigating median
ppc_stat(y = fish_clean$logFecundity,       # observed y
         yrep = y_draws_test2[1:8000,],     # simulated y
         stat = "median"                    # name of function
)
```


T = median
$T(y_{\mathrm{rep}})$
$T(y)$