MCMC Diagnostics - IFLS data

Sarah Teichman

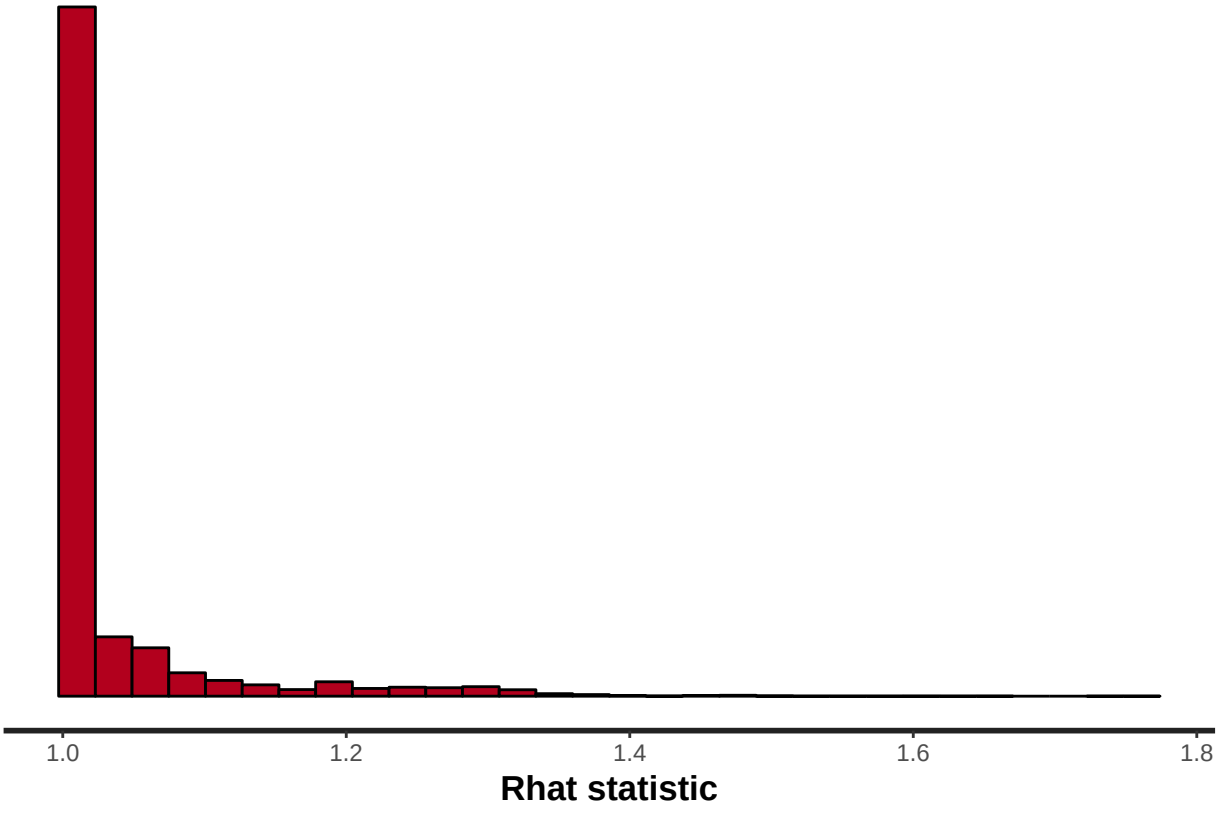
07/21/2020

```
K <- 7
Ti <- 3
N <- 1973
```

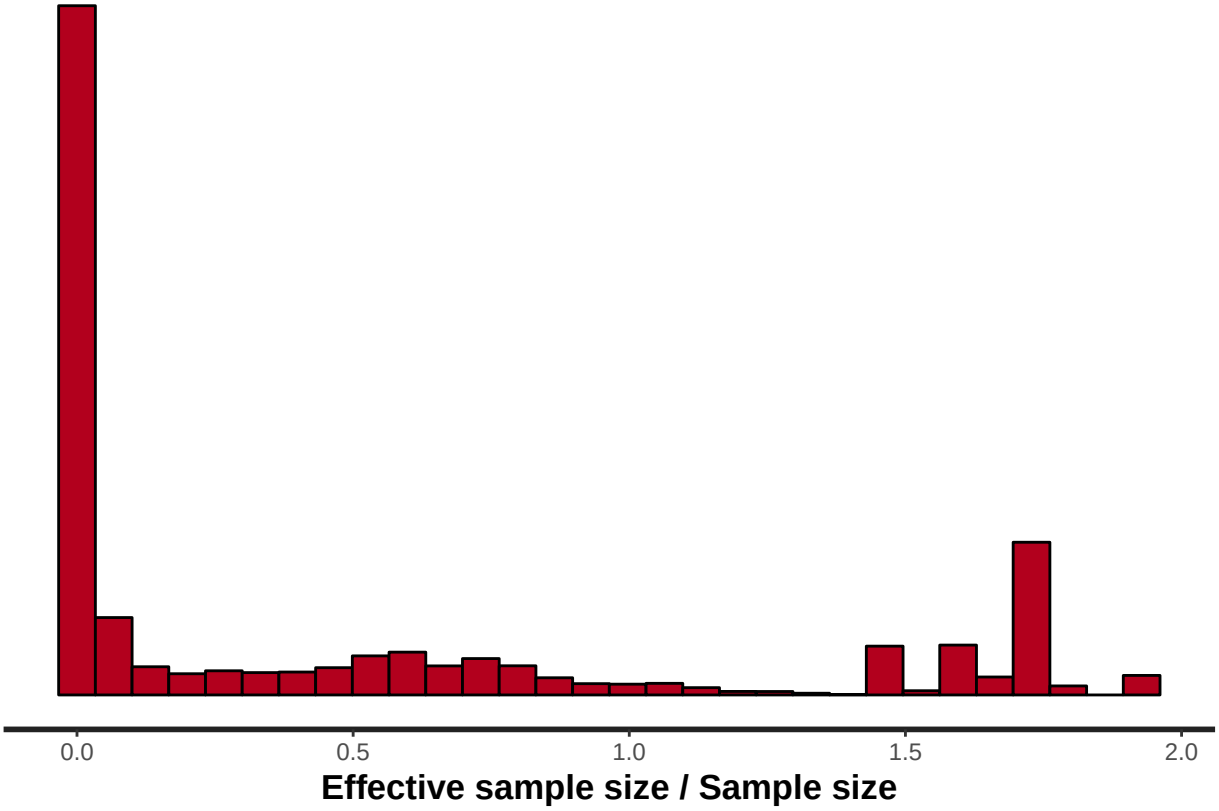
General MCMC diagnostic plots

Overall model diagnostics from rstan package.

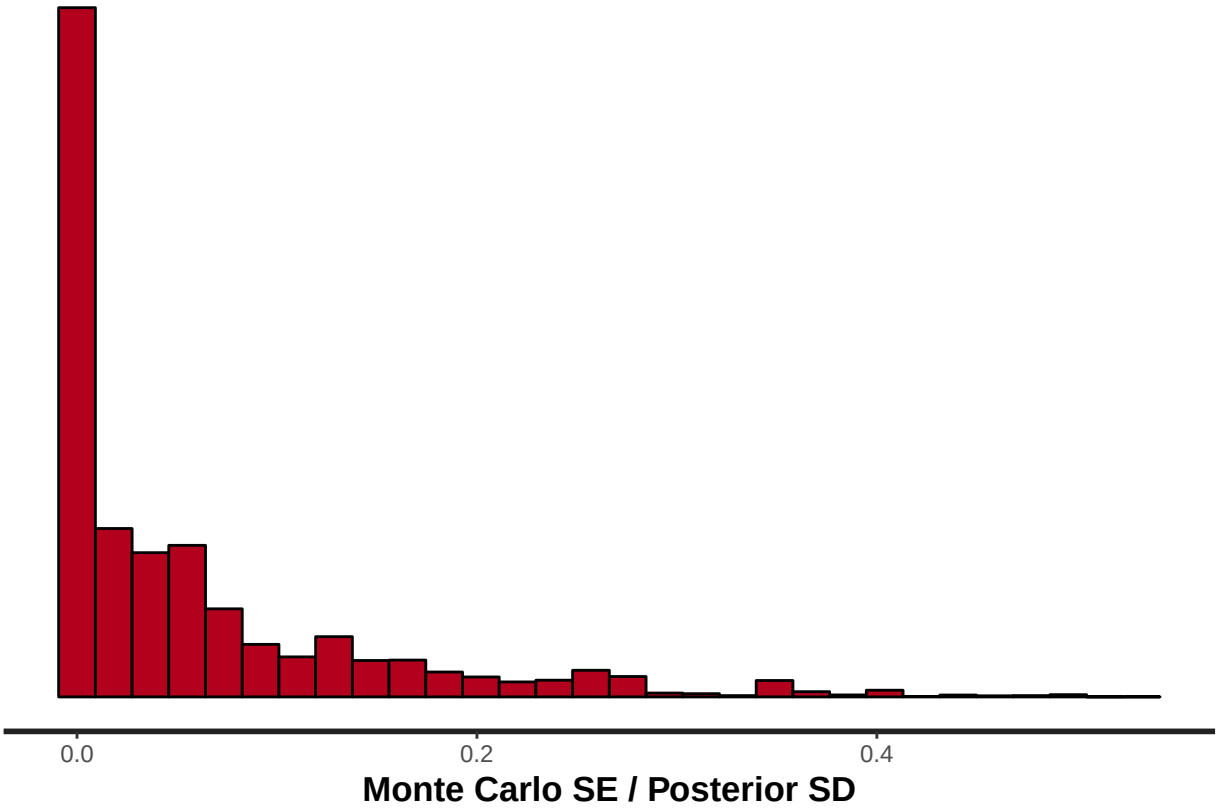
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



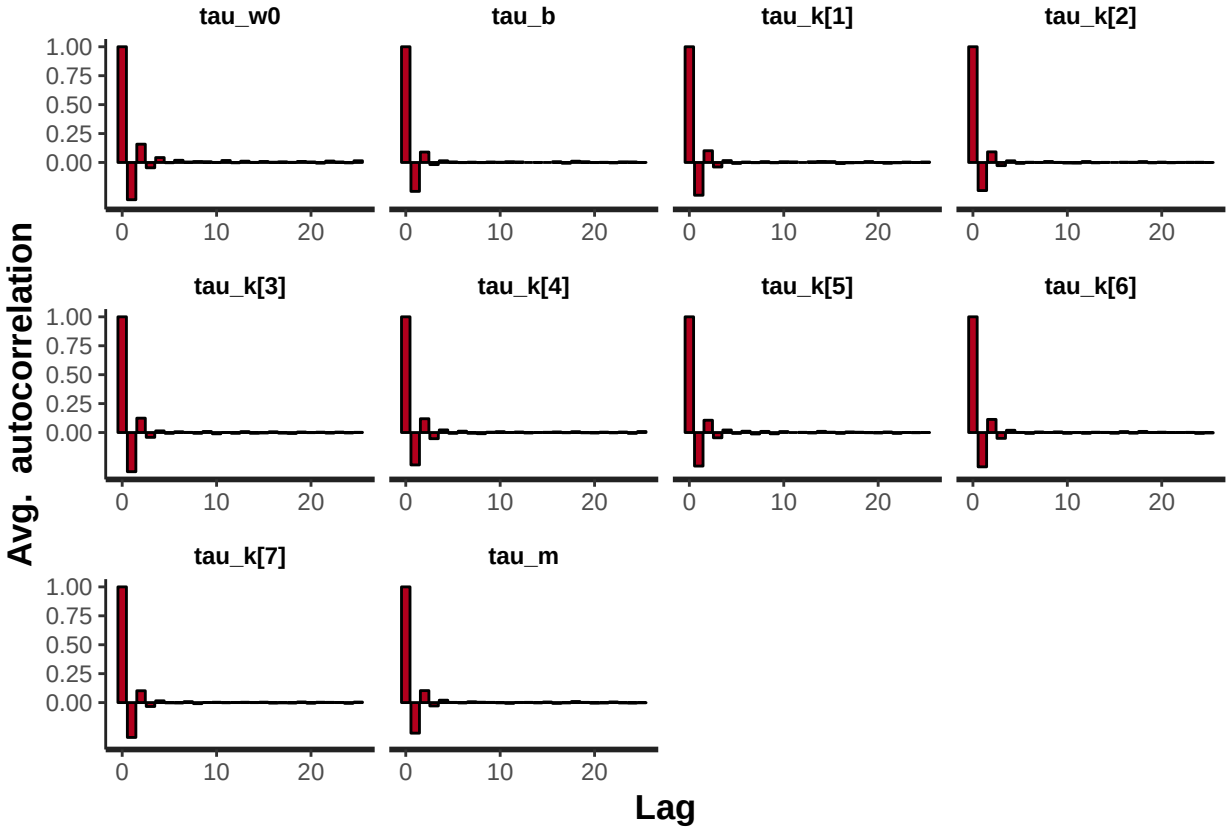
Individual Parameter Diagnostics

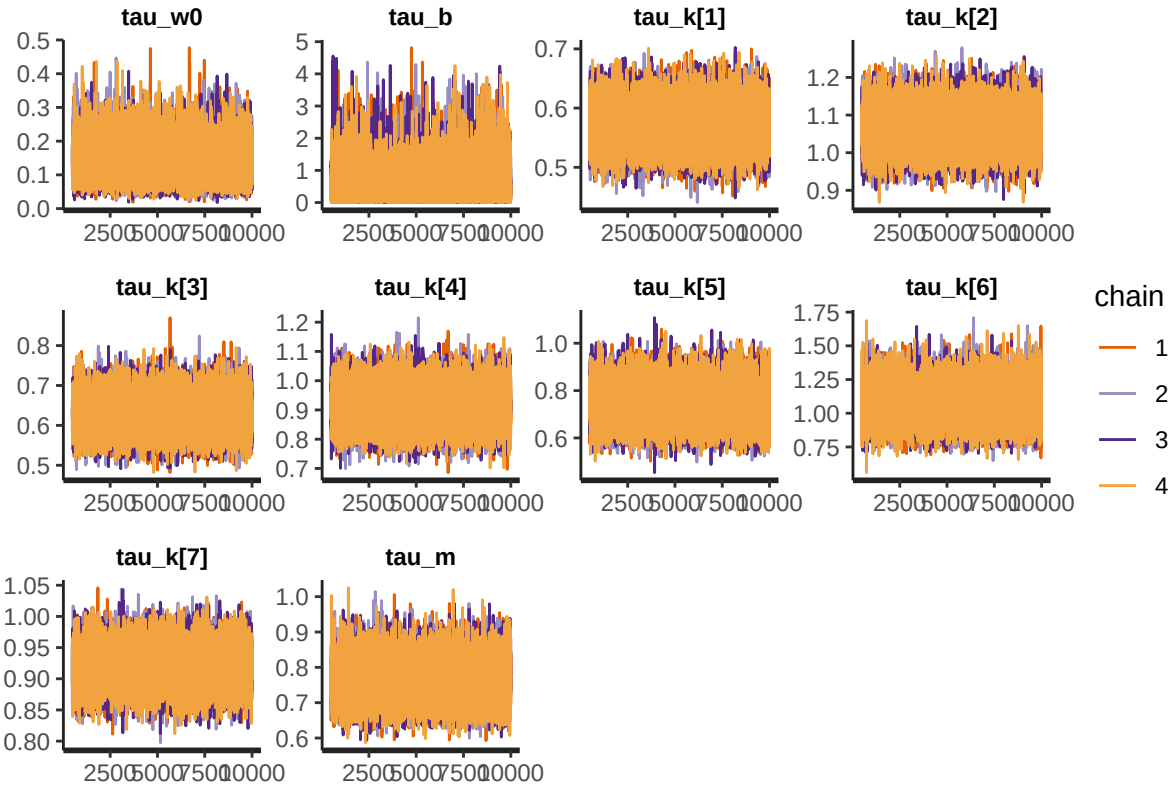
Individual parameter plots. Autocorrelation and trace plots for individual parameters, and histograms of posterior medians for group parameters.

```
get_single_plots <- function(fit, param) {
  print(fit_summ[param,c(1,2,3,5,6,7,9,10)])
  print(stan_ac(fit, pars = param))
  print(rstan::traceplot(fit, pars = param))
}
get_aggreg_plots <- function(fit, param, trim = F, trim_amount) {
  ind <- grep(paste0("^",param), rownames(as.data.frame(summary(fit)$summary)))
  medians <- data.frame(avg = as.data.frame(summary(fit)$summary)$`50%`[ind])
  title <- paste0("Posterior Medians of ",param)
  print(ggplot(medians, aes(x = avg)) + geom_histogram(bins = 30) + ggtitle(title) +
    xlab("Medians") + ylab("Count"))
  print("       ")
  if (trim == T) {
    lim <- quantile(abs(medians$avg), probs = trim_amount)
    meds_trim <- medians %>% filter(abs(medians$avg) < lim)
    print(ggplot(meds_trim, aes(x = avg)) + geom_histogram(bins = 60) +
            ggtitle(paste0(title, " Without Extreme ",100*(1-trim_amount),"%")))
  }
  means <- data.frame(avg = as.data.frame(summary(fit)$summary)$`mean`[ind])
  title <- paste0("Posterior Means of ",param)
  print(ggplot(means, aes(x = avg)) + geom_histogram(bins = 30) + ggtitle(title) +
    xlab("Means") + ylab("Count"))
  print("      ")
  sds <- data.frame(avg = as.data.frame(summary(fit)$summary)$`sd`[ind])
  title <- paste0("Posterior Standard Deviations of ",param)
  print(ggplot(sds, aes(x = avg)) + geom_histogram(bins = 30) + ggtitle(title) +
    xlab("Standard Deviations") + ylab("Count"))
}
plot_fit <- function(fit) {
  get_single_plots(fit, tau_params)
  get_single_plots(fit, beta)
  get_aggreg_plots(fit, "w")
  get_aggreg_plots(fit, "z")
  get_aggreg_plots(fit, "p")
}
plot_fit(fit)
```

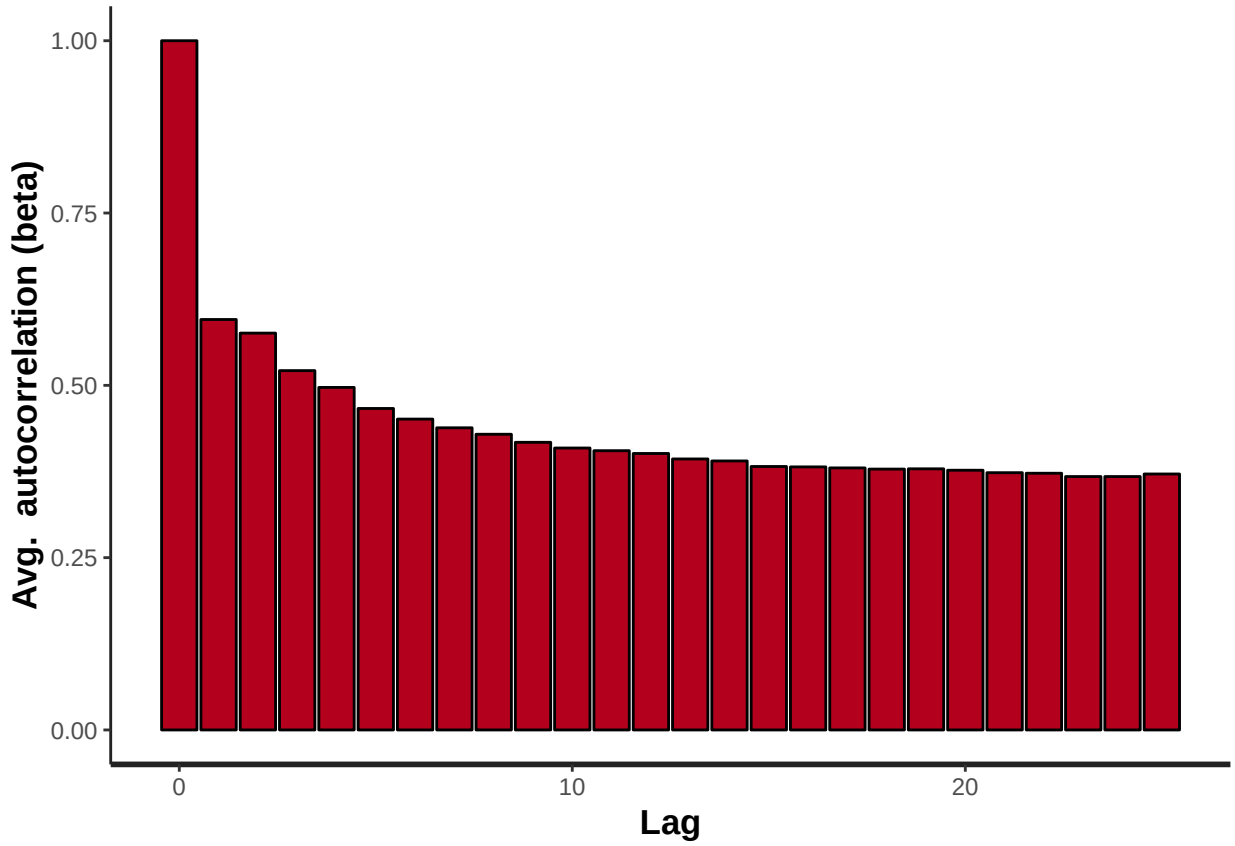
```
##               mean      se_mean         sd       25%       50%       75%
## tau_w0   0.1393623 0.0004433818 0.05233179 0.1014644 0.1322487 0.1702399
## tau_b    0.5003613 0.0023003079 0.50118804 0.1454357 0.3494985 0.6880240
## tau_k[1] 0.5711584 0.0001315253 0.03348011 0.5483916 0.5703450 0.5933404
## tau_k[2] 1.0653977 0.0002090100 0.04977134 1.0309337 1.0647099 1.0987120
## tau_k[3] 0.6293228 0.0001575772 0.04265028 0.5998784 0.6280303 0.6574728
## tau_k[4] 0.9124367 0.0002497892 0.06203567 0.8696811 0.9110412 0.9536499
## tau_k[5] 0.7431449 0.0002882225 0.07443530 0.6918243 0.7406955 0.7915888
## tau_k[6] 1.0656795 0.0004929857 0.12668075 0.9781905 1.0602587 1.1469348
## tau_k[7] 0.9205138 0.0001134266 0.02926739 0.9005347 0.9201386 0.9402220
## tau_m    0.7625909 0.0002250807 0.05309478 0.7257083 0.7601524 0.7965980
##             n_eff      Rhat
```

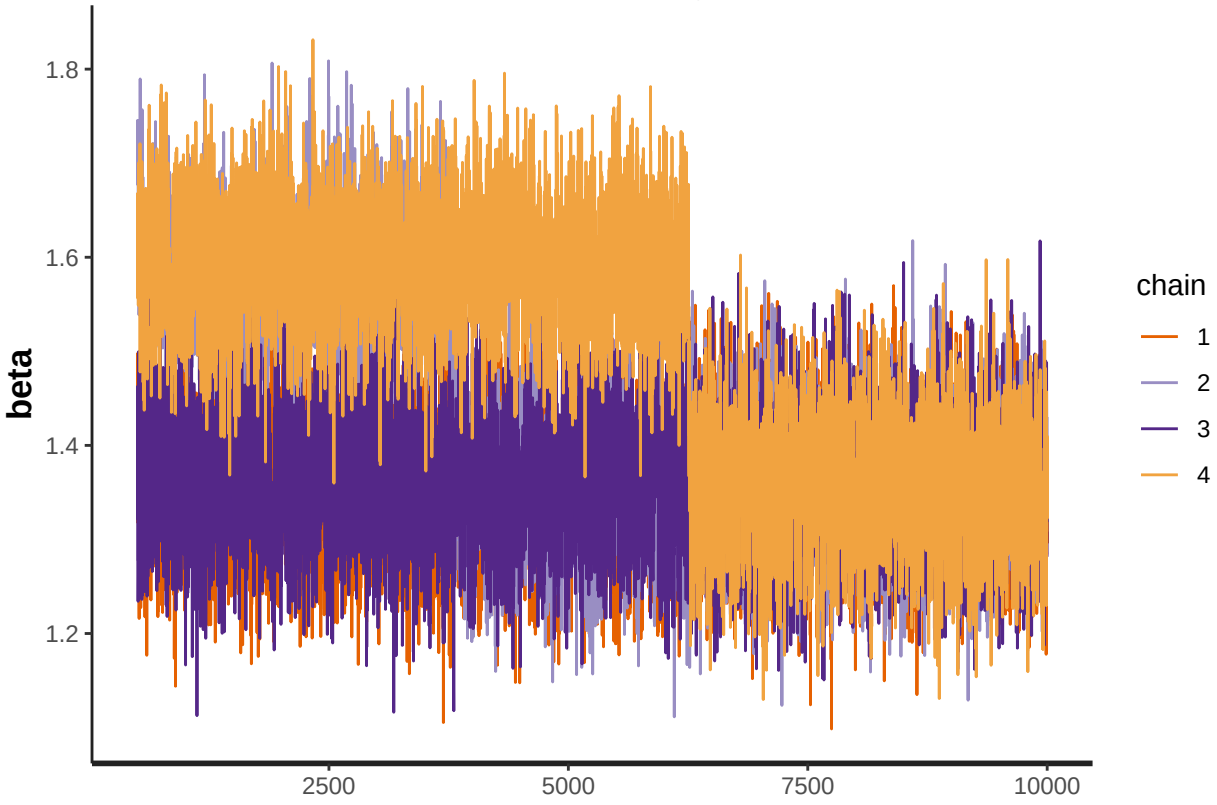
```
## tau_w0   13930.78 1.0030183
## tau_b    47471.12 1.0017243
## tau_k[1] 64797.08 0.9998993
## tau_k[2] 56705.43 1.0000874
## tau_k[3] 73258.29 0.9999264
## tau_k[4] 61678.78 0.9999273
## tau_k[5] 66696.35 0.9999715
## tau_k[6] 66031.73 0.9999950
## tau_k[7] 66579.13 0.9999999
## tau_m    55645.14 0.9999861
```

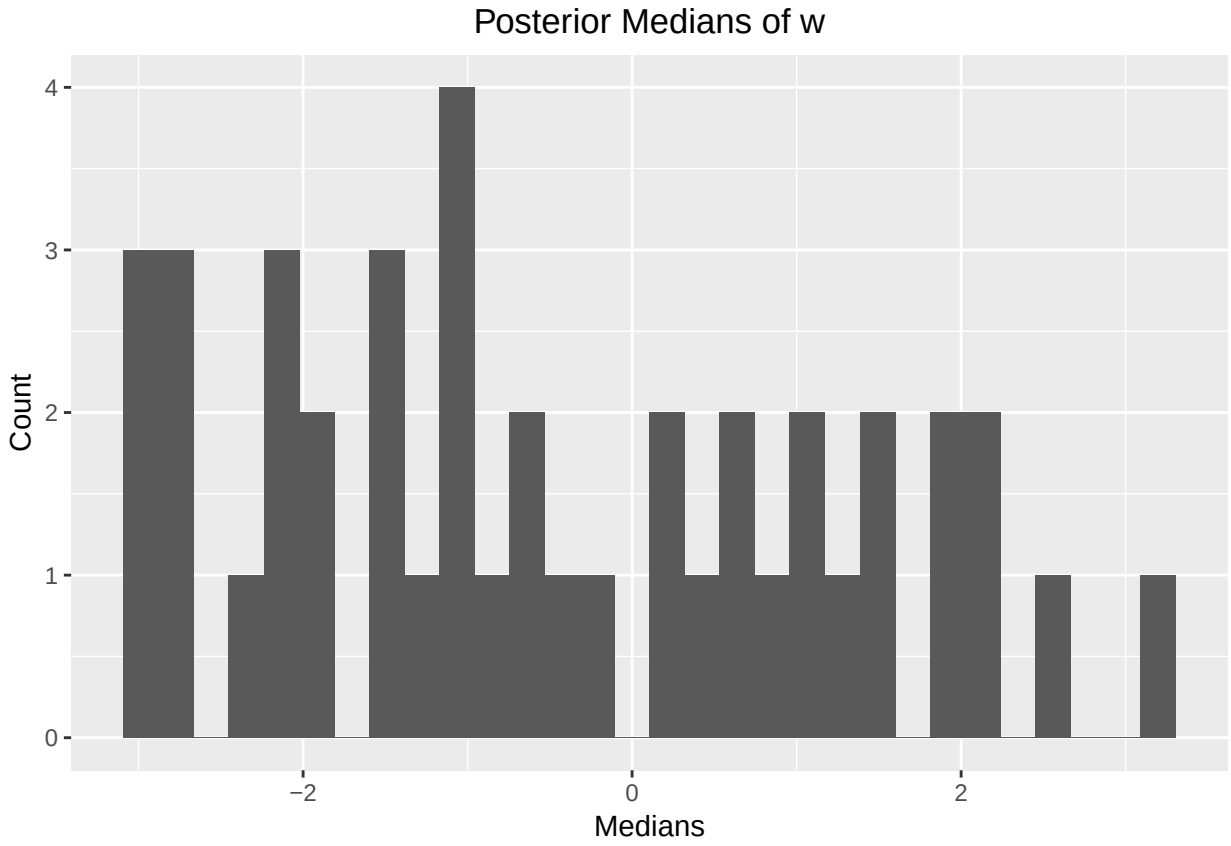




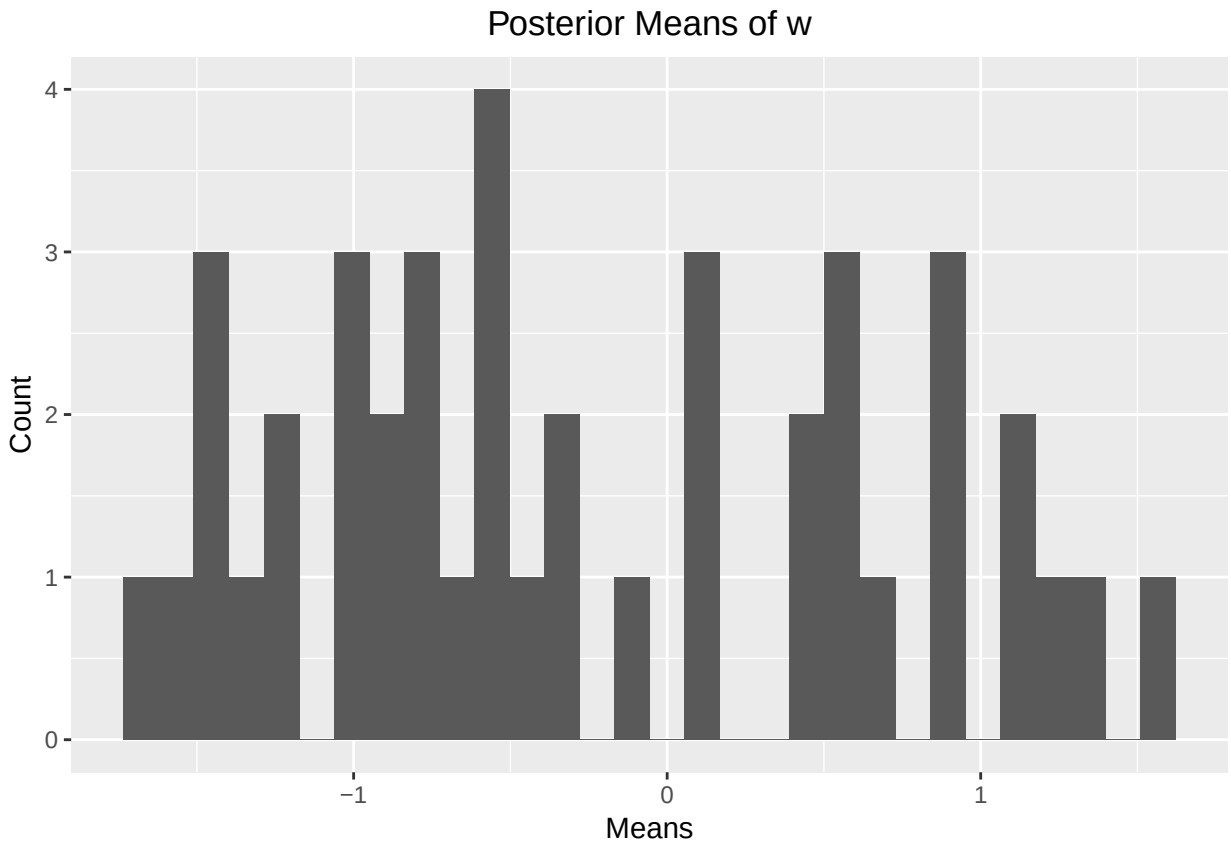

```
##          mean     se_mean         sd      25%      50%      75%    n_eff
## beta 1.41419 0.03730978 0.1193964 1.328311 1.384897 1.484532 10.24087
##          Rhat
## beta 1.482525
```


Avg. autocorrelation (beta)
1.00
0.75
0.50
0.25
0.00
0
10
20
Lag

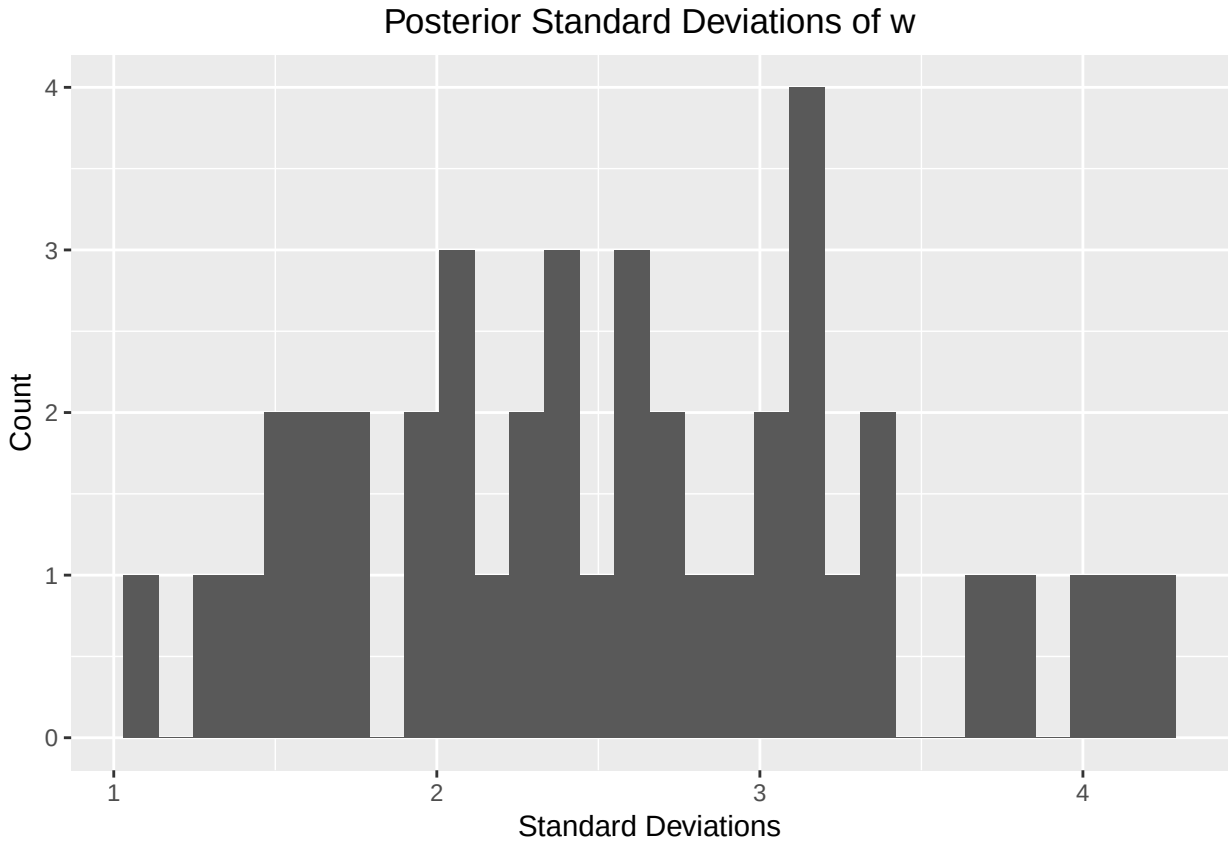

beta
1.8
1.6
1.4
1.2
2500
5000
7500
10000
chain
1
2
3
4

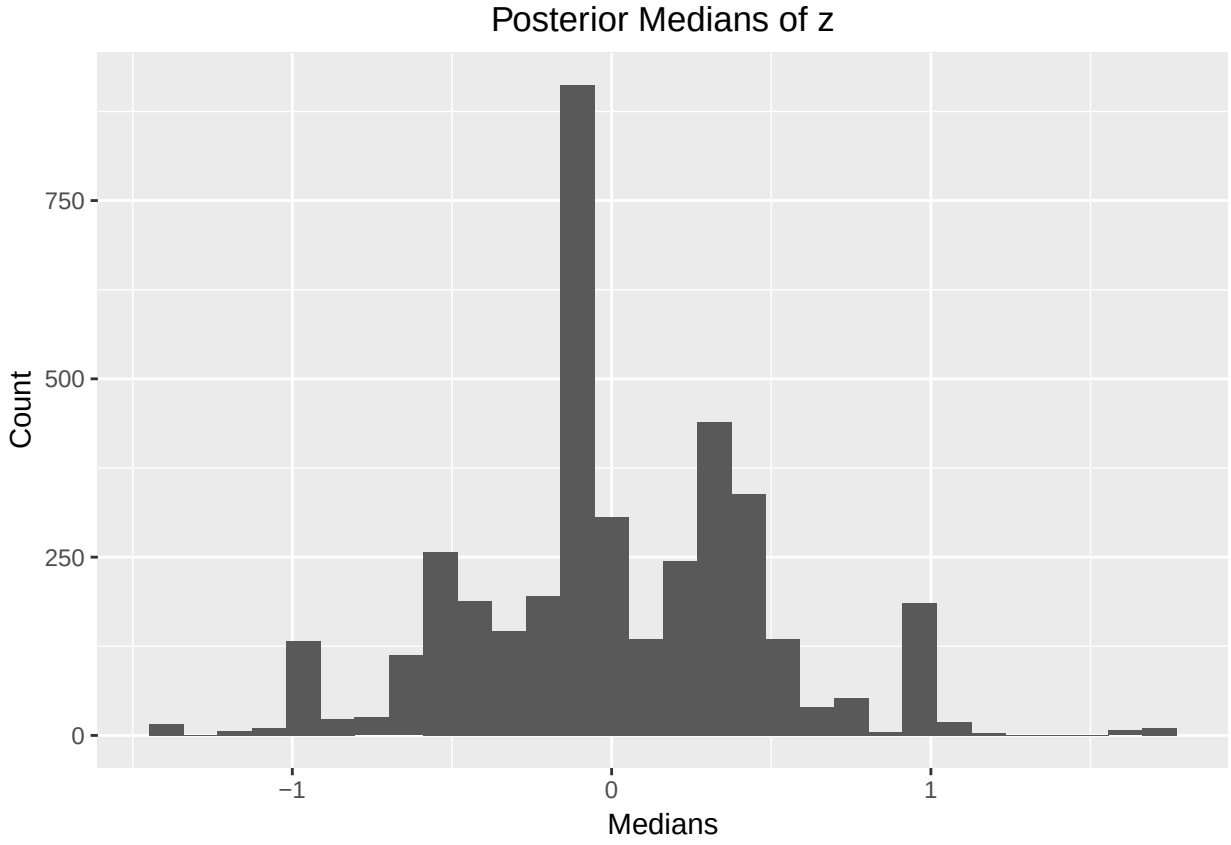



```
## [1] "       "
```

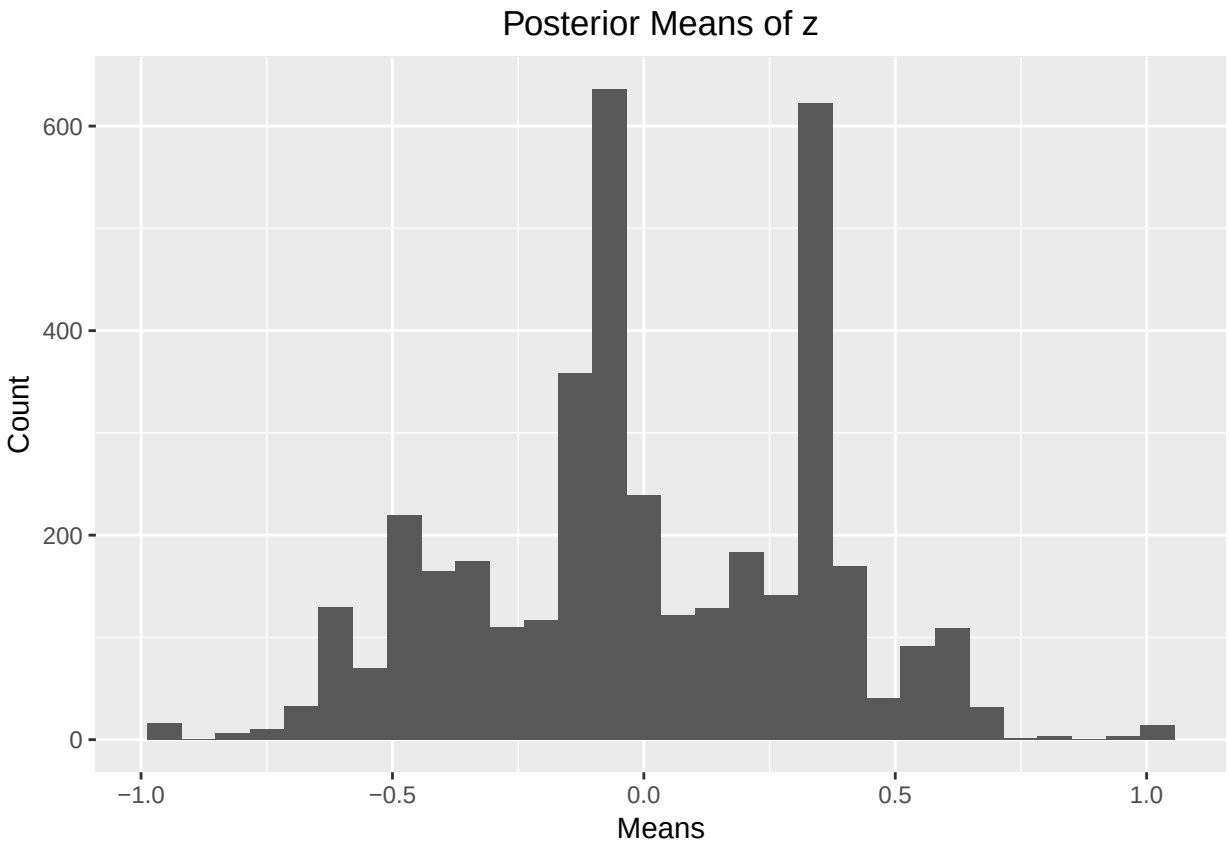



```
## [1] "      "
```

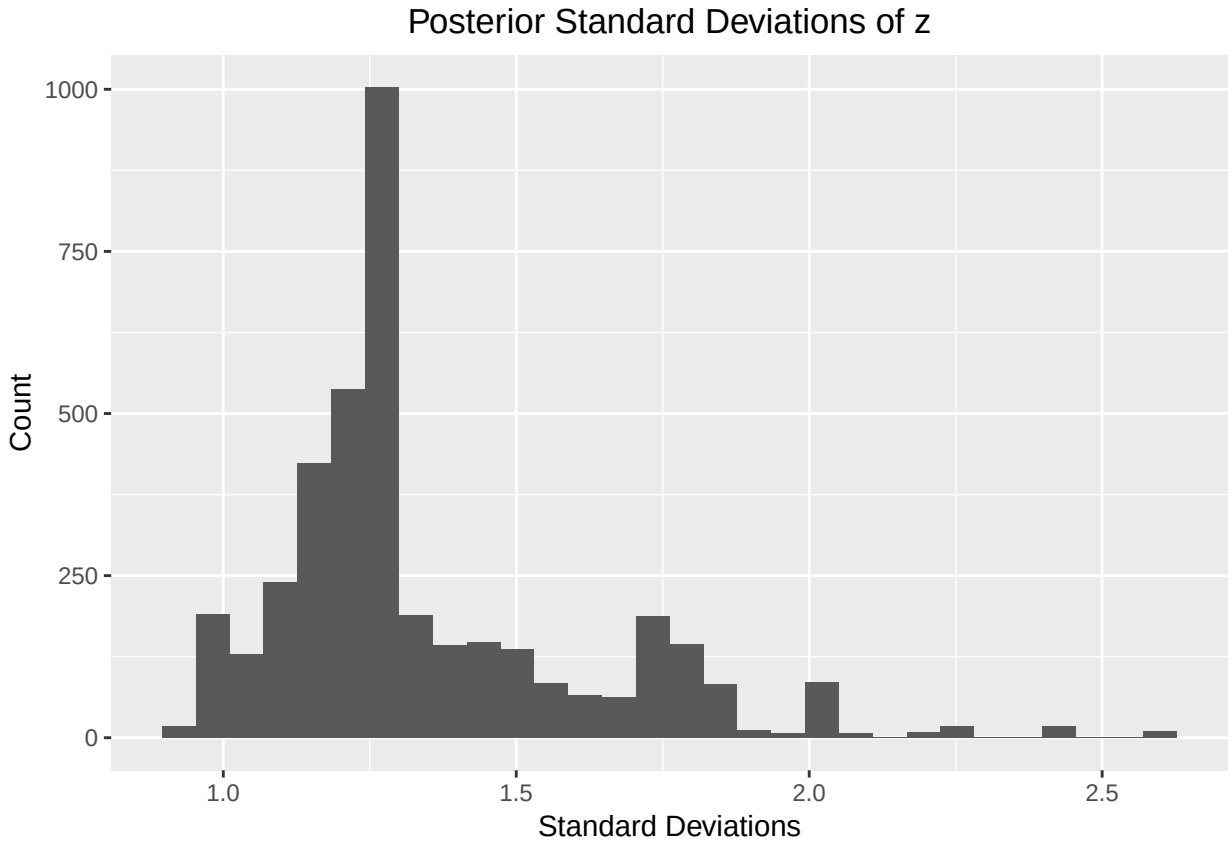

Posterior Standard Deviations of w
Count
0
1
2
3
4
1
2
3
4
Standard Deviations

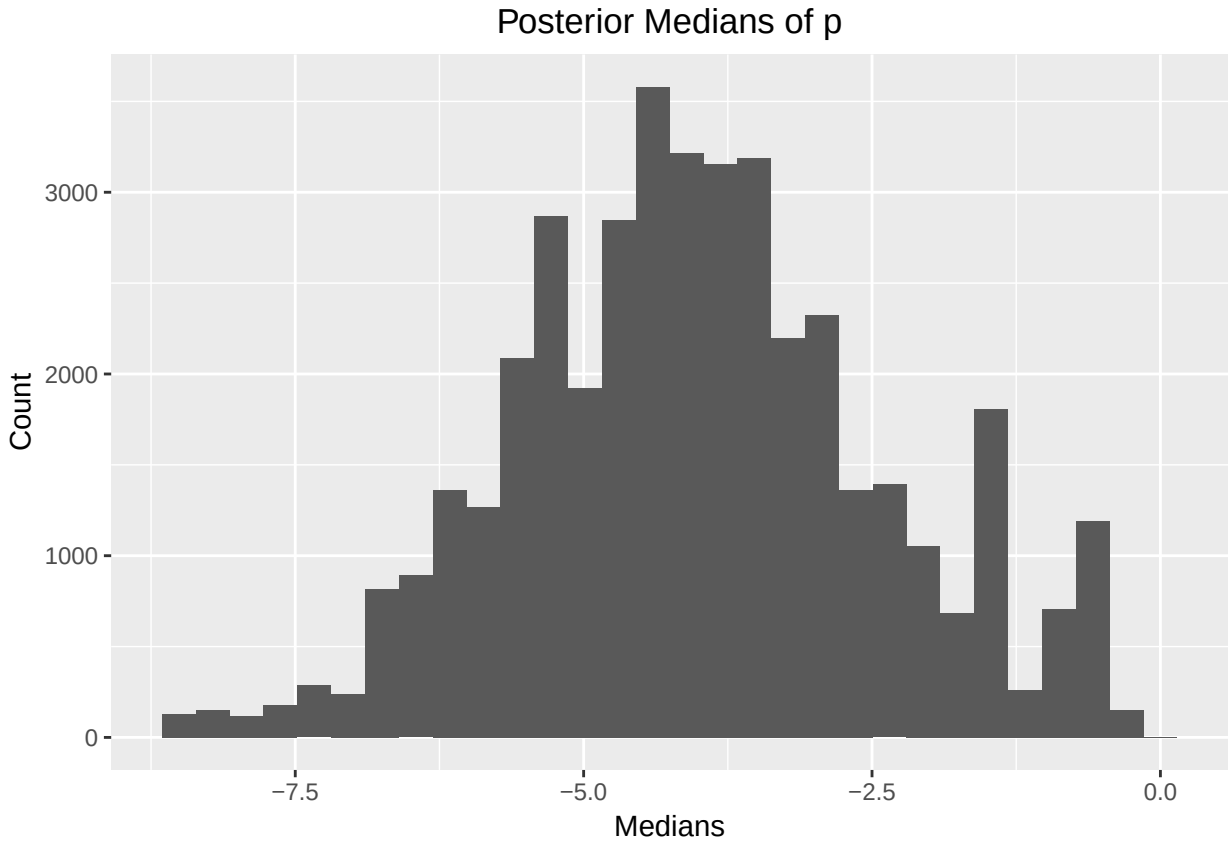

Posterior Medians of z
Count
0
250
500
750
−1
0
1
Medians

```
## [1] "       "
```

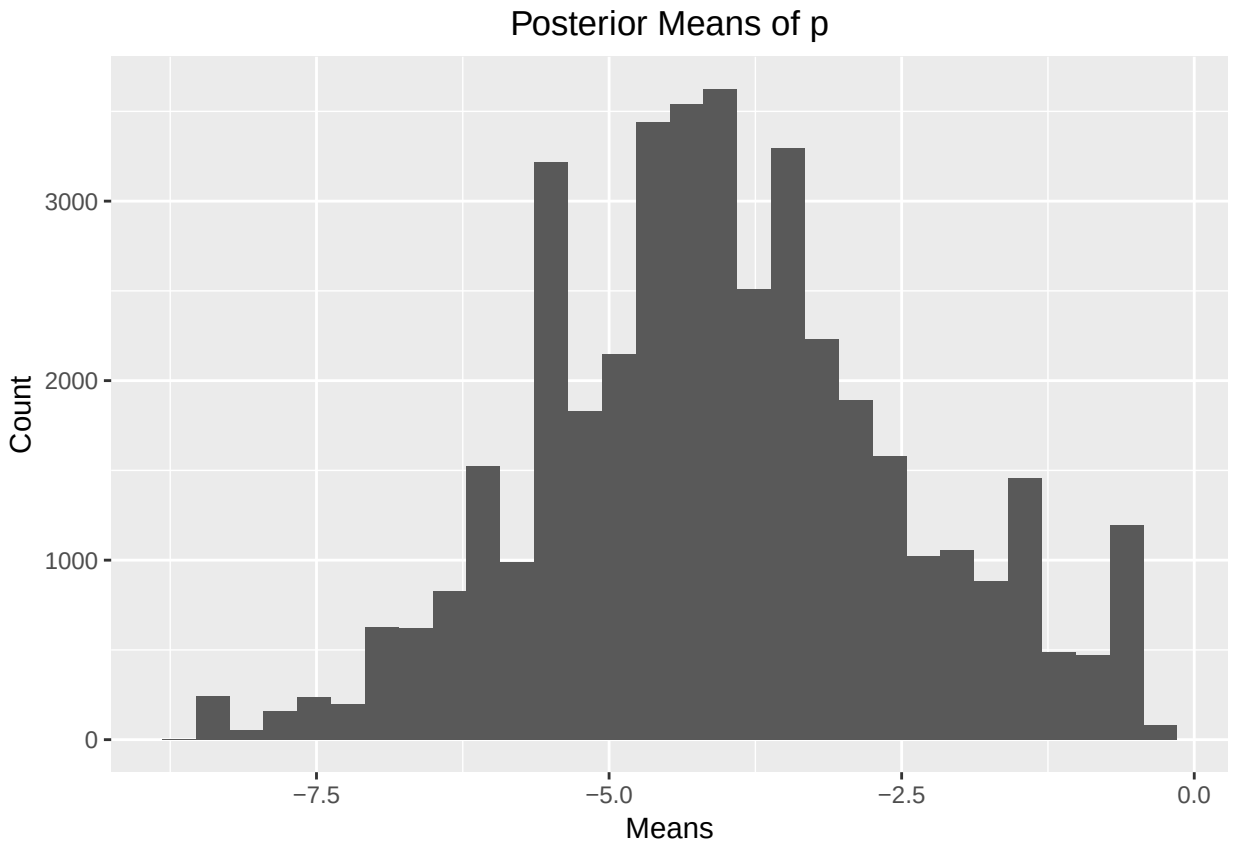



```
## [1] "       "
```

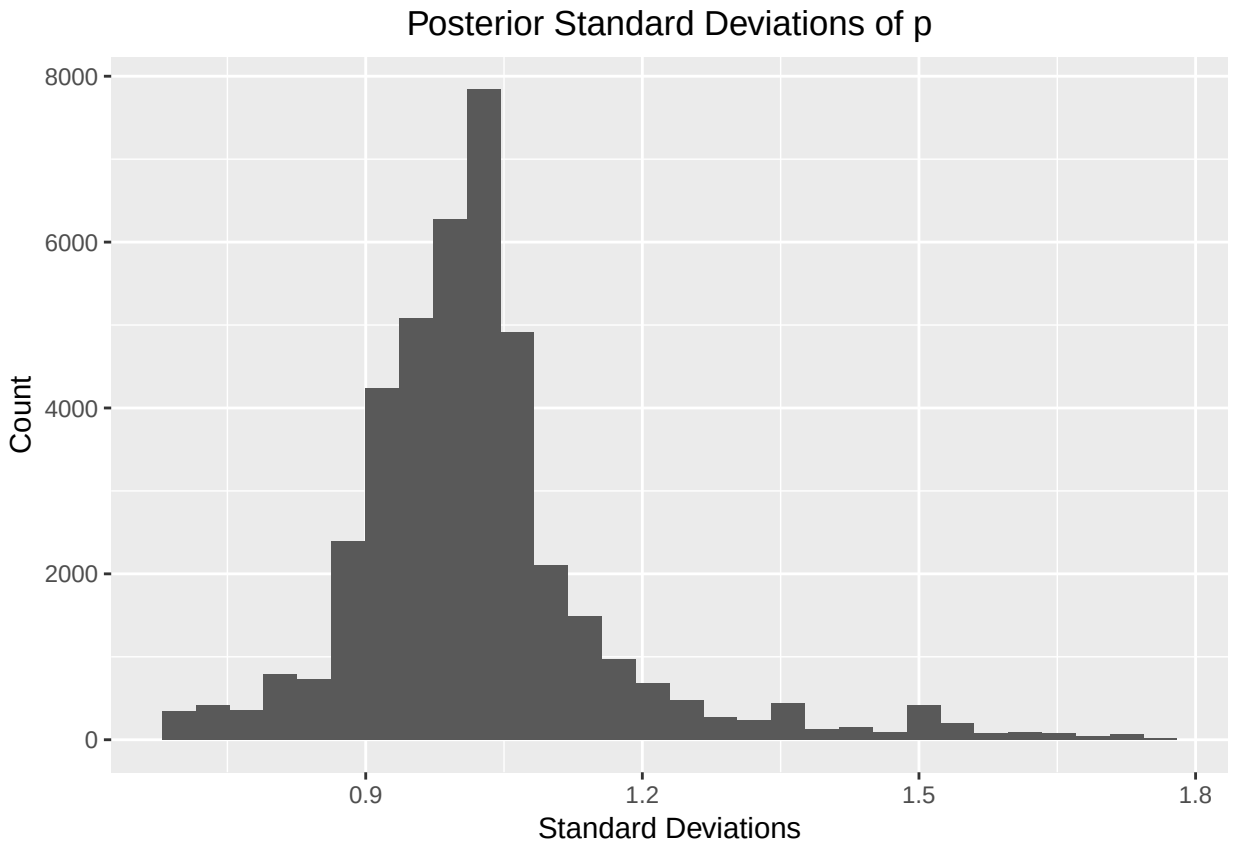

Posterior Standard Deviations of z
Count
1000
750
500
250
0
1.0
1.5
2.0
2.5
Standard Deviations


Posterior Medians of p
Count
3000
2000
1000
0
−7.5
−5.0
−2.5
0.0
Medians

```
## [1] "       "
```



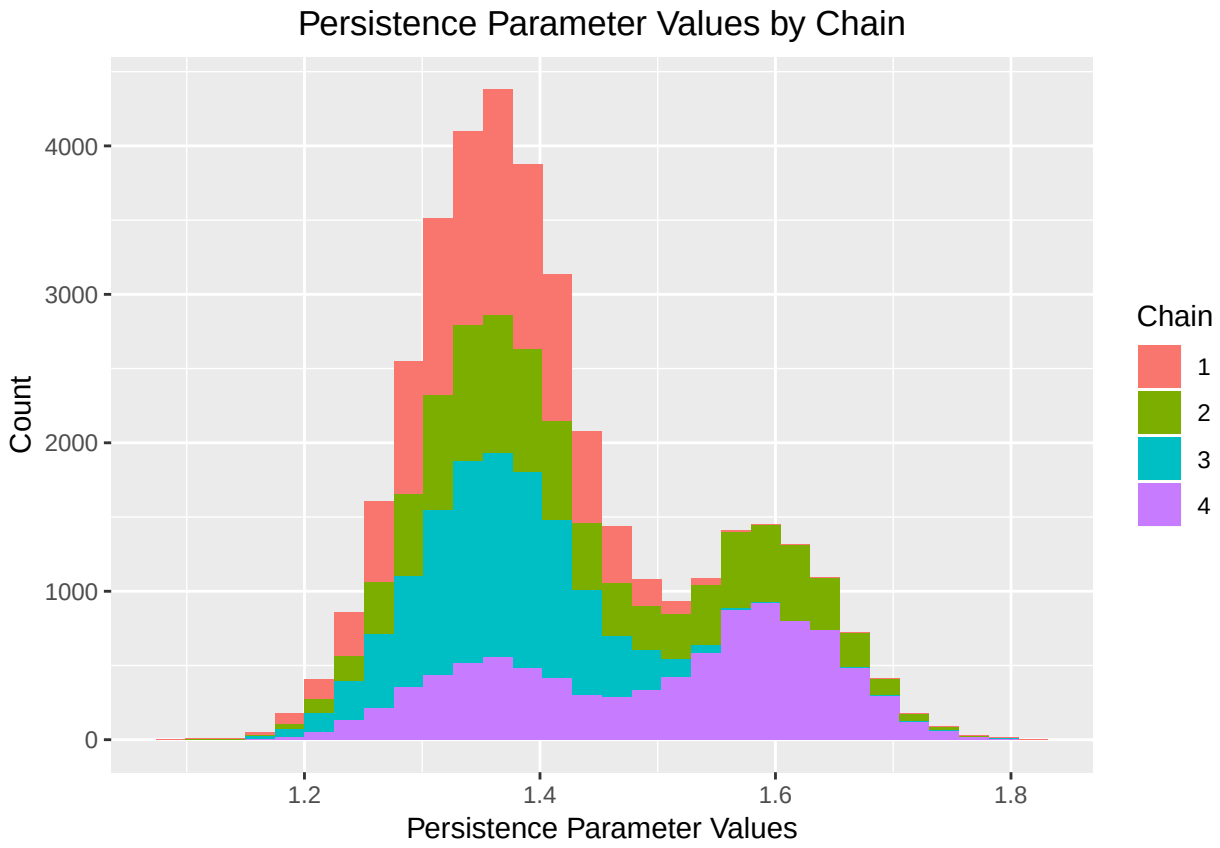

```
## [1] "       "
```



Histograms for β values and w, and z posterior means across chains.

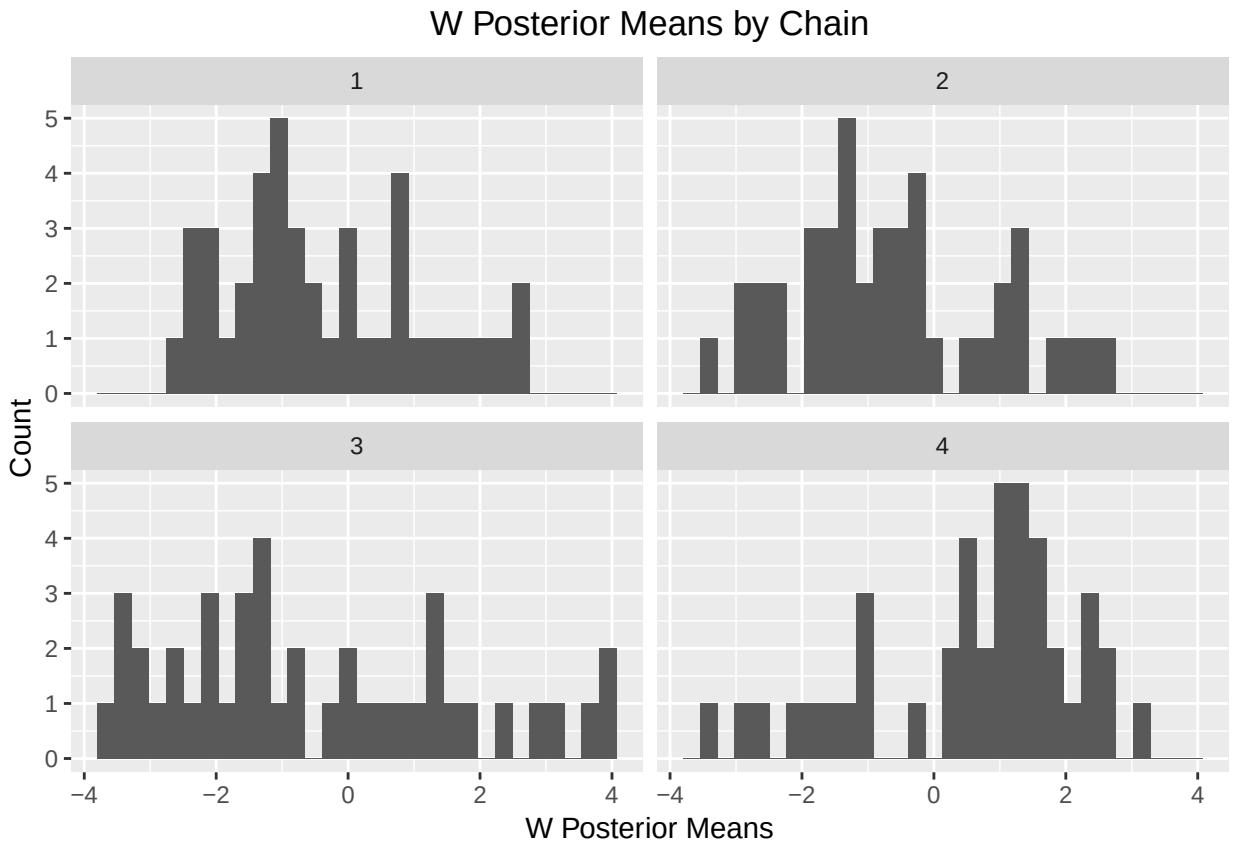
```
draws <- as.matrix(fit)
beta_col <- draws[,which(colnames(draws) == "beta")]
l <- length(beta_col)/4
plot_dat <- data.frame(val = beta_col, Chain = as.factor(c(rep(1,l),rep(2,l),rep(3,l),rep(4,l))))
ggplot(plot_dat, aes(x = val, fill = Chain)) + geom_histogram() +
  xlab("Persistence Parameter Values") + ylab("Count") + ggtitle("Persistence Parameter Values by Chain
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



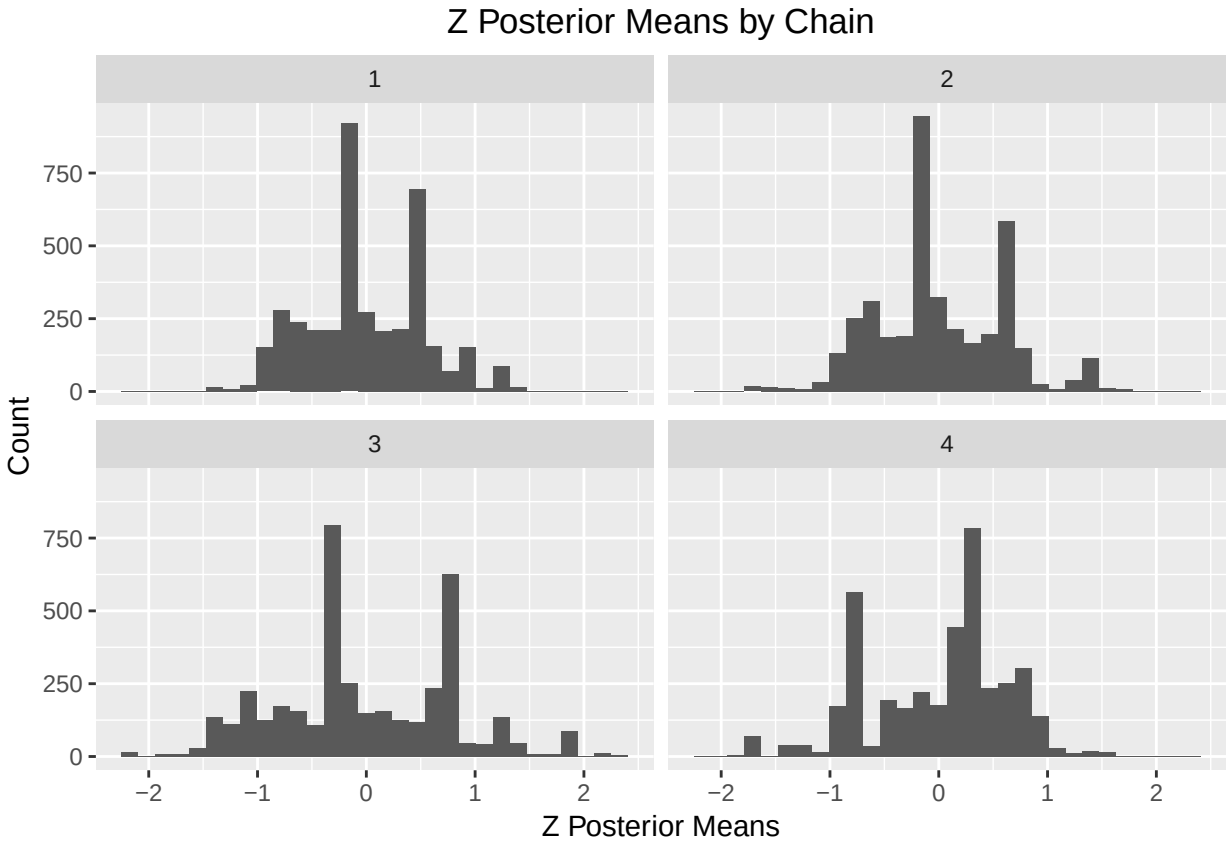

```
w_ind <- grep("^w", colnames(draws))
z_ind <- grep("^z", colnames(draws))
w_dat <- data.frame(chain1 = colMeans(draws[1:l,w_ind]),
                    chain2 = colMeans(draws[(l+1):(2*l),w_ind]),
                    chain3 = colMeans(draws[(2*l+1):(3*l),w_ind]),
                    chain4 = colMeans(draws[(3*l+1):(4*l),w_ind])) %>%
  pivot_longer(cols = everything(), names_to = "Chain", names_prefix = "chain") %>%
  mutate(Chain = as.factor(Chain))
z_dat <- data.frame(chain1 = colMeans(draws[1:l,z_ind]),
                    chain2 = colMeans(draws[(l+1):(2*l),z_ind]),
                    chain3 = colMeans(draws[(2*l+1):(3*l),z_ind]),
                    chain4 = colMeans(draws[(3*l+1):(4*l),z_ind])) %>%
  pivot_longer(cols = everything(), names_to = "Chain", names_prefix = "chain") %>%
  mutate(Chain = as.factor(Chain))
ggplot(w_dat, aes(x = value)) +
  geom_histogram() + facet_wrap(~Chain) + xlab("W Posterior Means") + ylab("Count") +
  ggtitle("W Posterior Means by Chain")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(z_dat, aes(x = value)) +
  geom_histogram() + facet_wrap(~Chain) +
  xlab("Z Posterior Means") + ylab("Count") +
  ggtitle("Z Posterior Means by Chain")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


Z Posterior Means by Chain
1
2
3
4
Count
750
500
250
0
−2
−1
0
1
2
Z Posterior Means