MCMC Diagnostics - IFLS data

Sarah Teichman

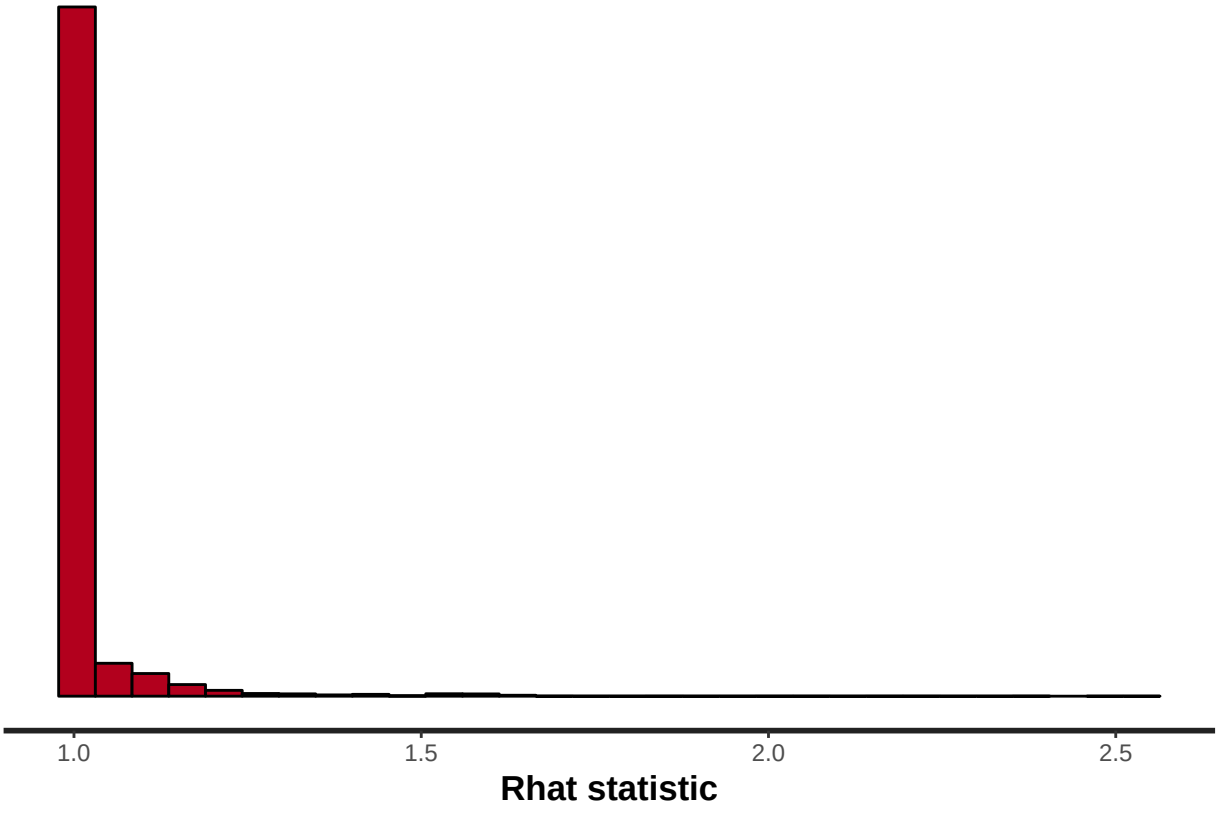
07/20/2020

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

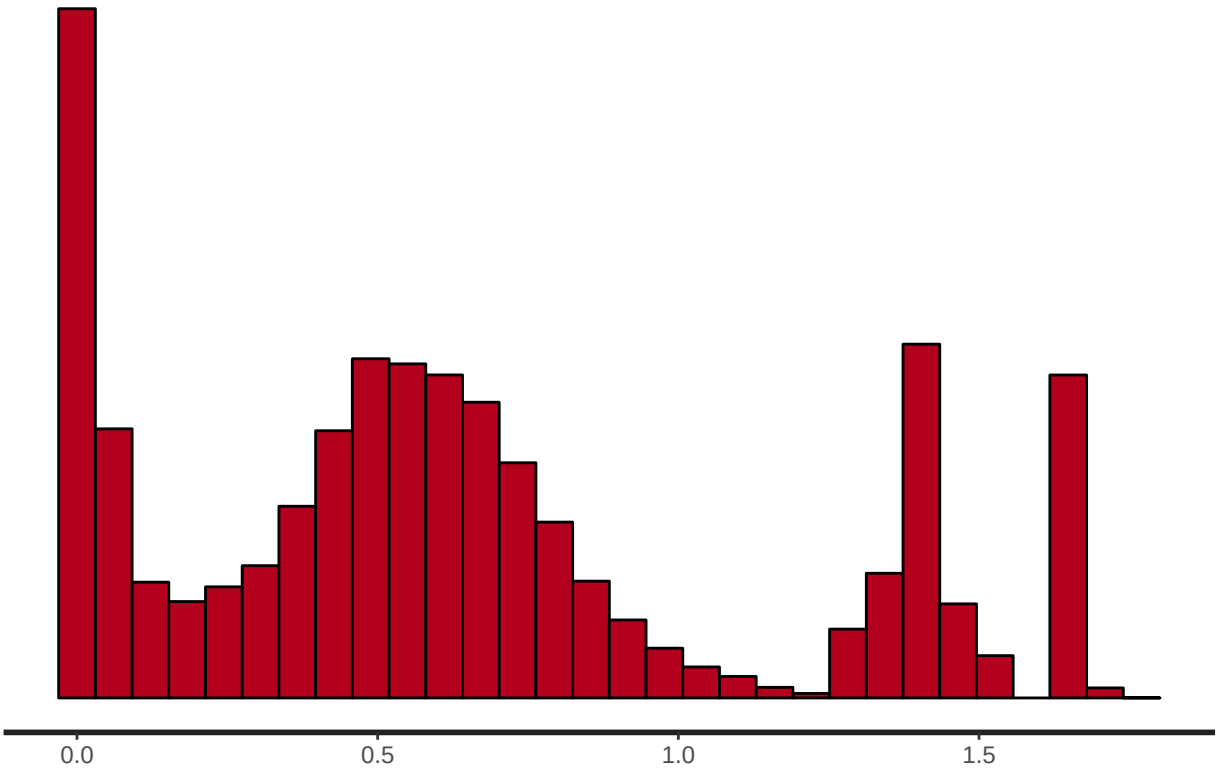
General MCMC diagnostic plots

Overall model diagnostics from rstan package.

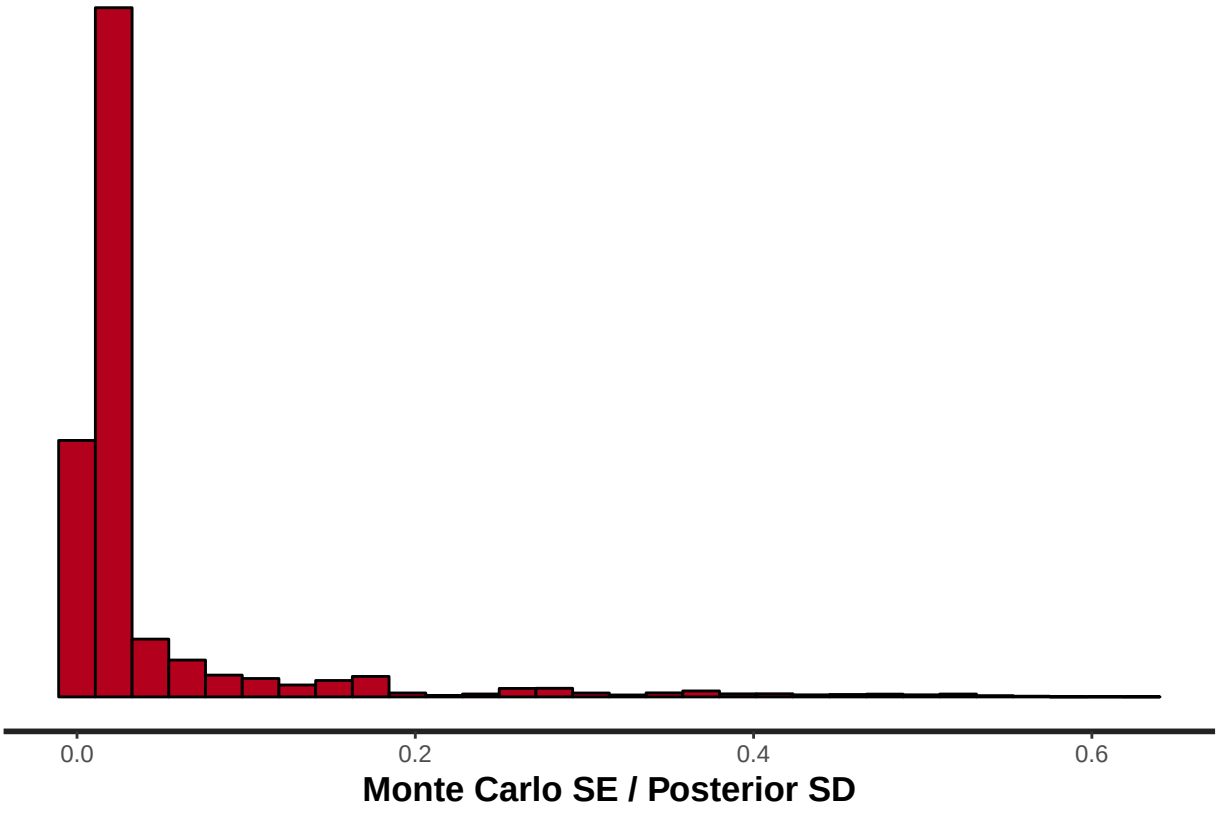
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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




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



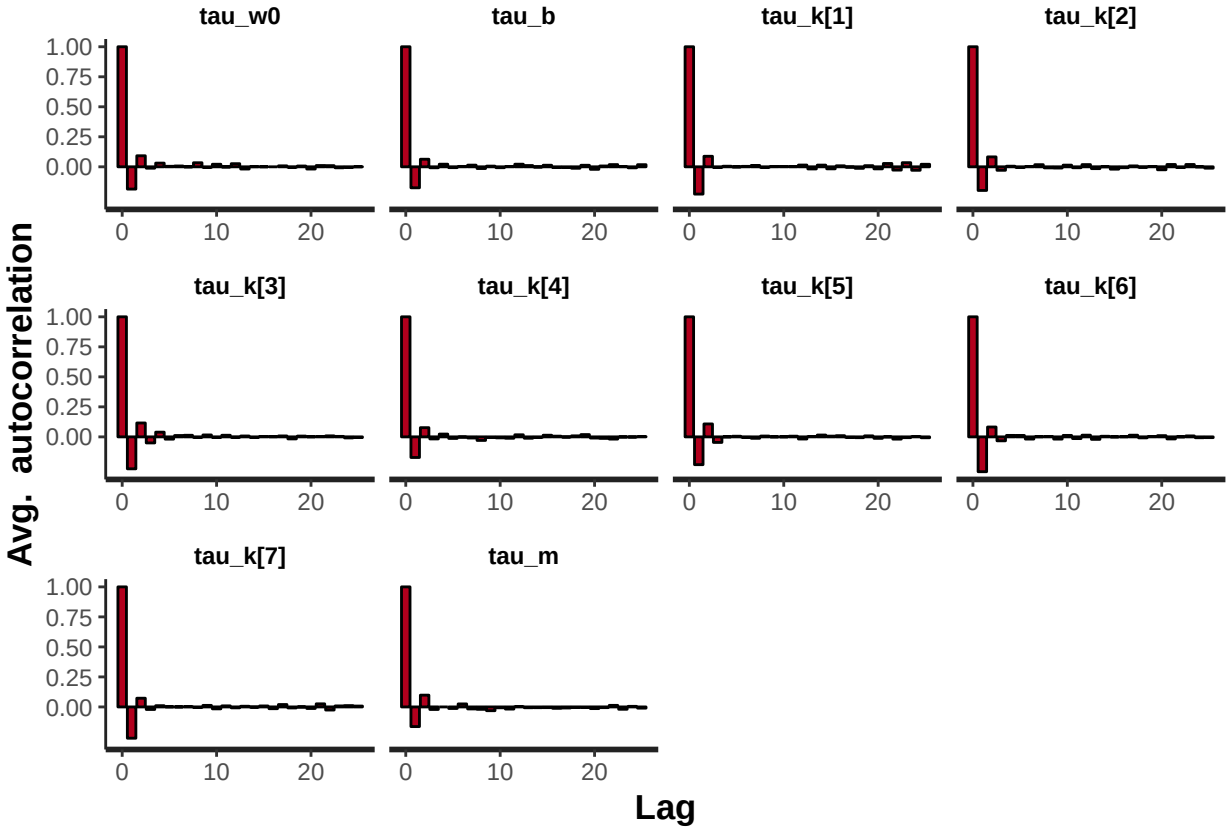
Individual Parameter Diagnostics

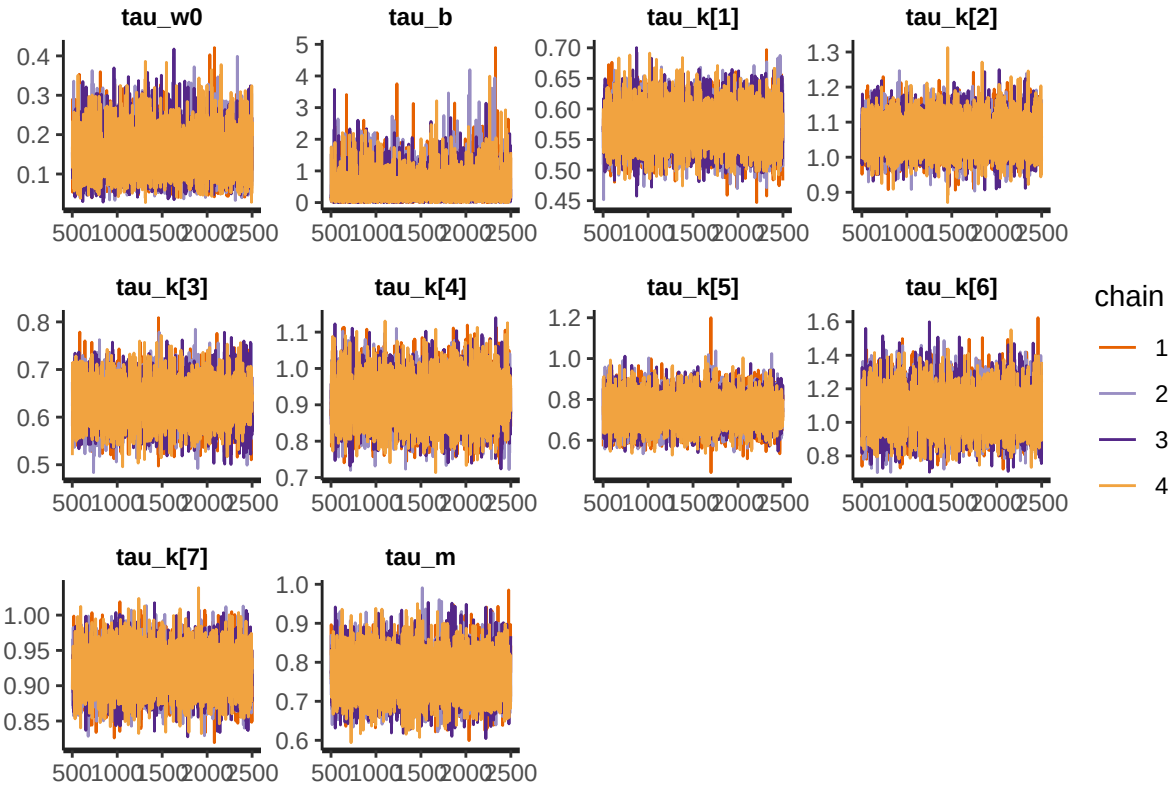
Individual parameter plots. Autocorrelation and trace plots for individual parameters, and histograms of posterior medians for group parameters.

```
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)
```

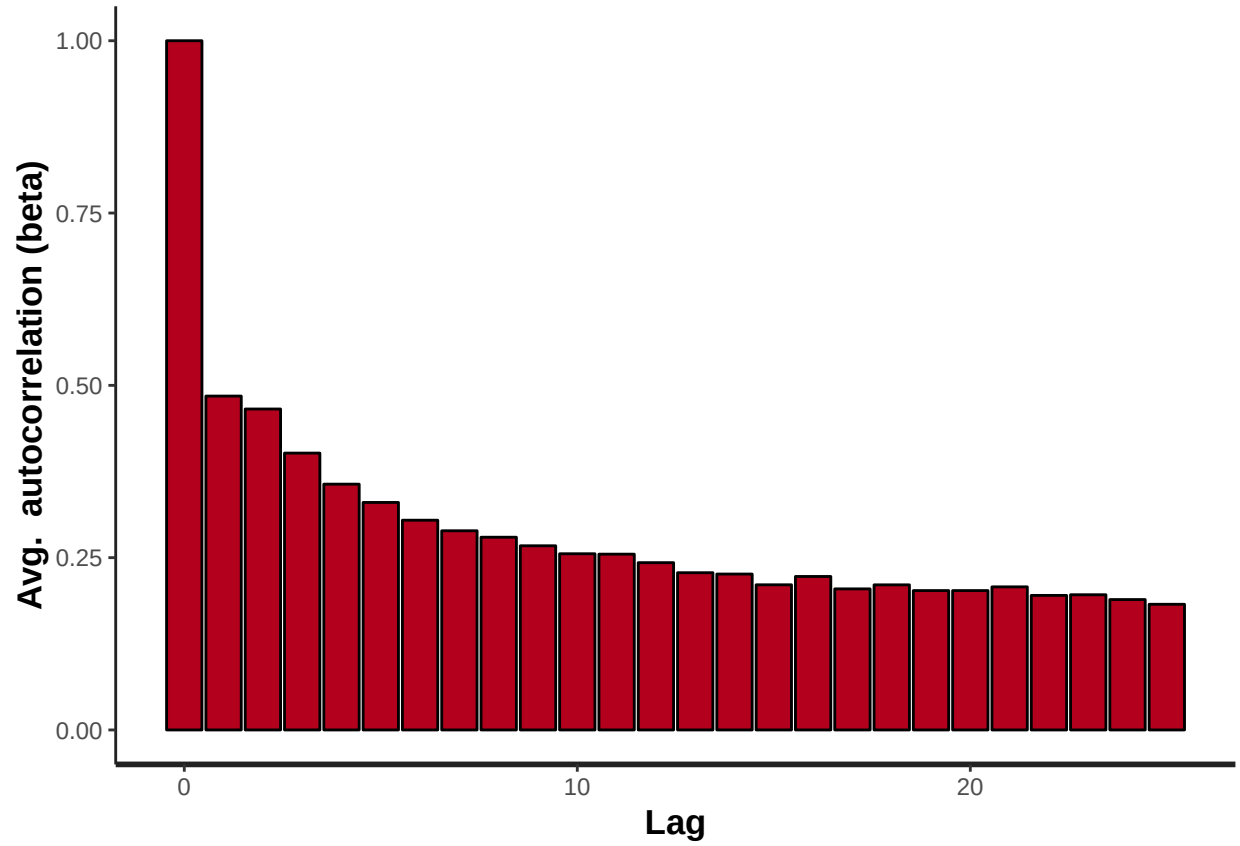
```
##               mean      se_mean         sd       25%       50%       75%
## tau_w0   0.1467123 0.0005838951 0.05516851 0.1070829 0.1408186 0.1775320
## tau_b    0.4477844 0.0047170821 0.45691039 0.1298817 0.3070689 0.6118770
## tau_k[1] 0.5709837 0.0003237884 0.03421349 0.5469175 0.5702316 0.5946055
## tau_k[2] 1.0653074 0.0004692265 0.04996238 1.0314909 1.0644623 1.0987653
## tau_k[3] 0.6290430 0.0004010428 0.04257017 0.5999882 0.6278160 0.6568670
## tau_k[4] 0.9120477 0.0006250761 0.06274102 0.8688532 0.9097249 0.9537379
## tau_k[5] 0.7423042 0.0006948458 0.07547221 0.6906908 0.7391188 0.7915704
## tau_k[6] 1.0653028 0.0010510752 0.12510010 0.9791048 1.0595866 1.1477319
## tau_k[7] 0.9206736 0.0002537771 0.02926233 0.9012115 0.9200680 0.9399175
## tau_m    0.7631593 0.0005460626 0.05412461 0.7249667 0.7603573 0.7984375
##              n_eff      Rhat
```

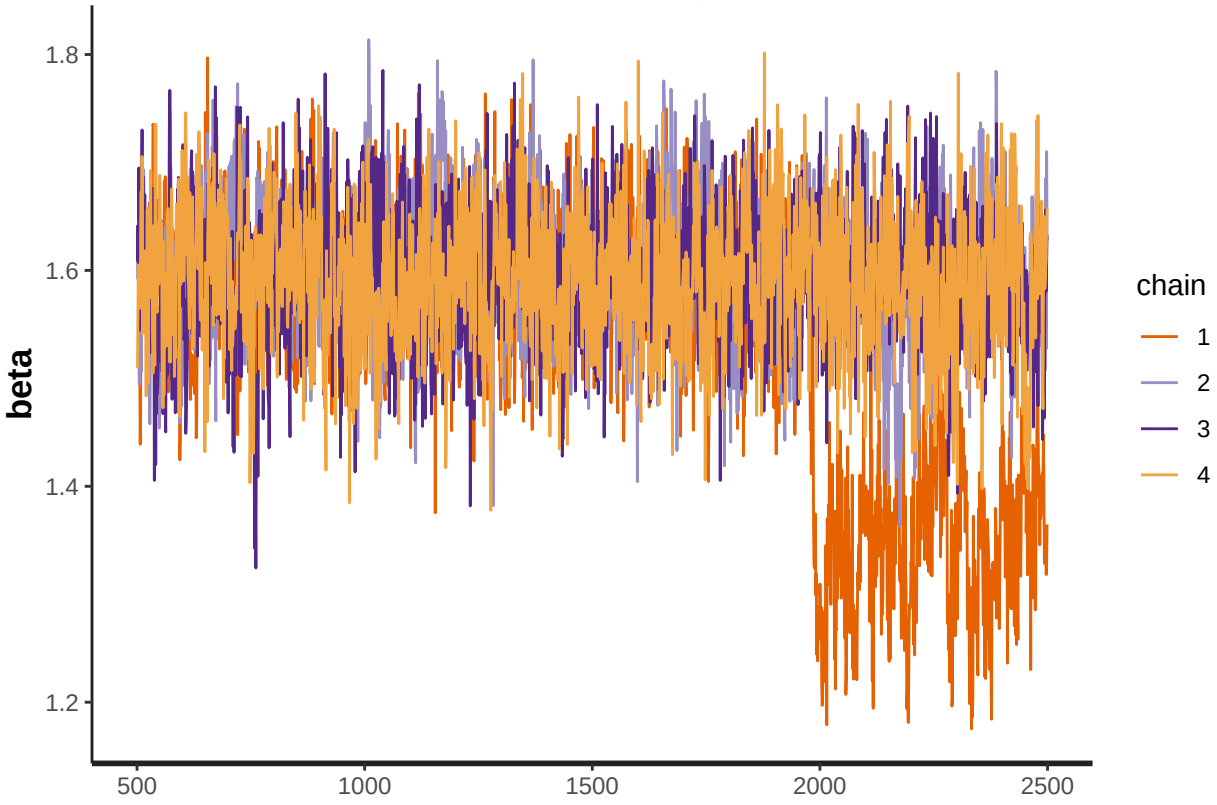
```
## tau_w0    8927.151 1.0000991
## tau_b     9382.427 1.0009735
## tau_k[1] 11165.346 0.9999195
## tau_k[2] 11337.604 0.9996419
## tau_k[3] 11267.544 0.9996970
## tau_k[4] 10074.823 1.0002552
## tau_k[5] 11797.699 0.9997012
## tau_k[6] 14166.017 0.9998289
## tau_k[7] 13295.755 0.9998074
## tau_m     9824.370 0.9999769
```

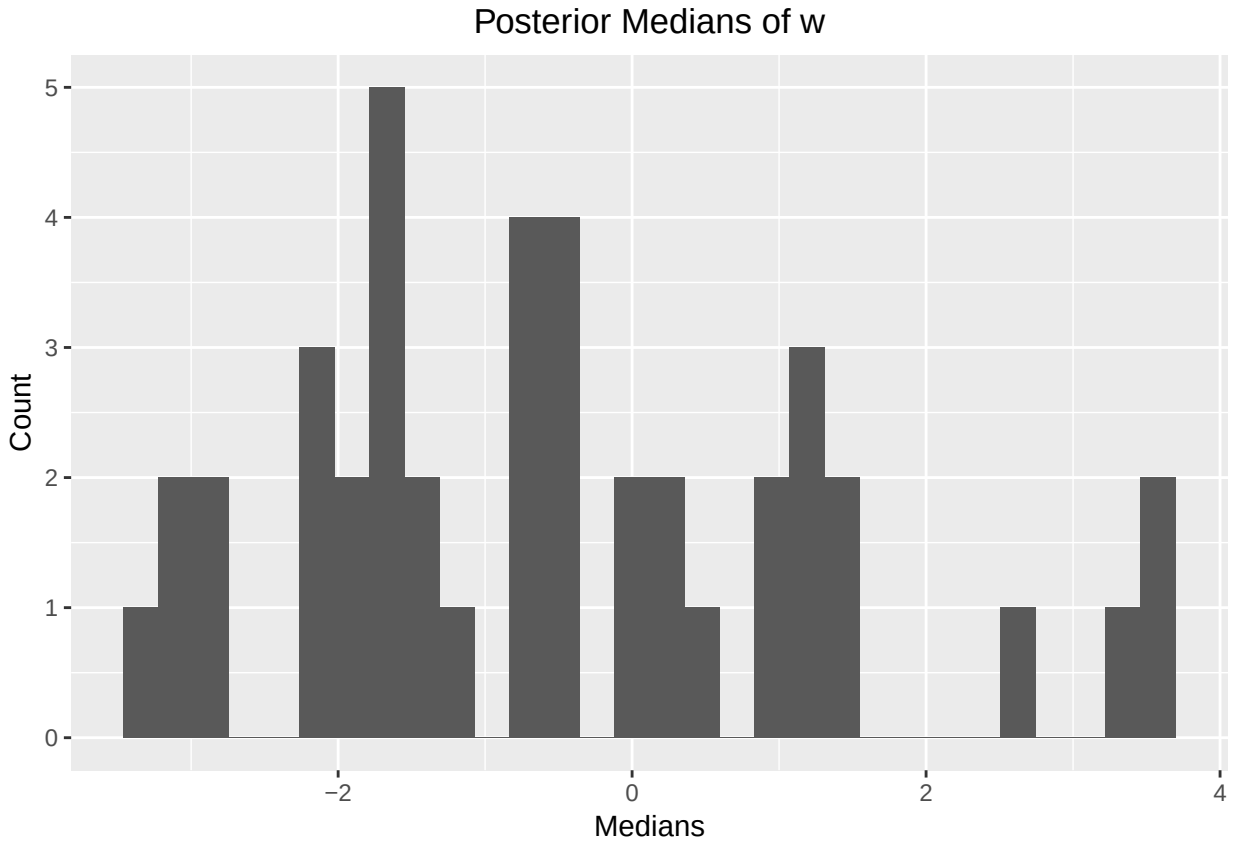




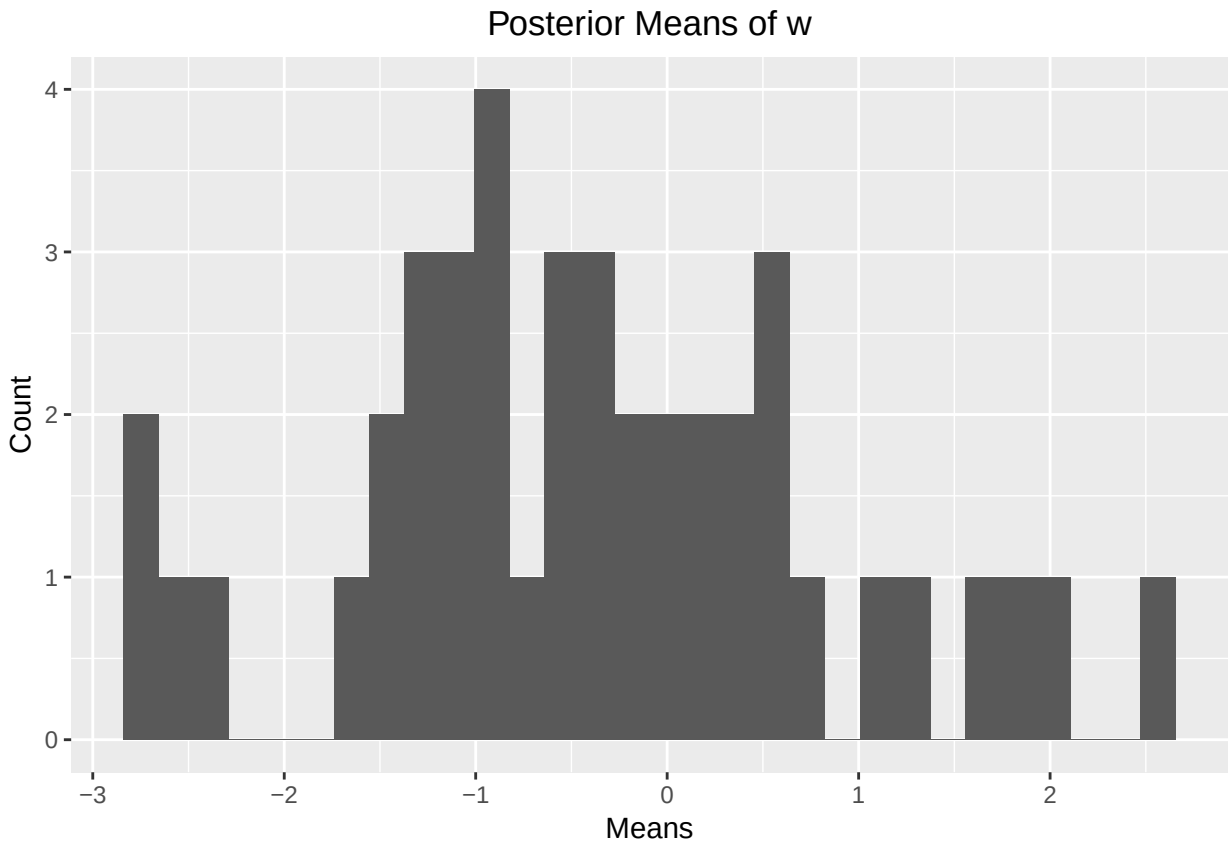

```
##           mean    se_mean         sd      25%      50%      75%  n_eff
## beta 1.578286 0.0188415 0.08820759 1.540189 1.588957 1.633234 21.917
##          Rhat
## beta 1.163328
```


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

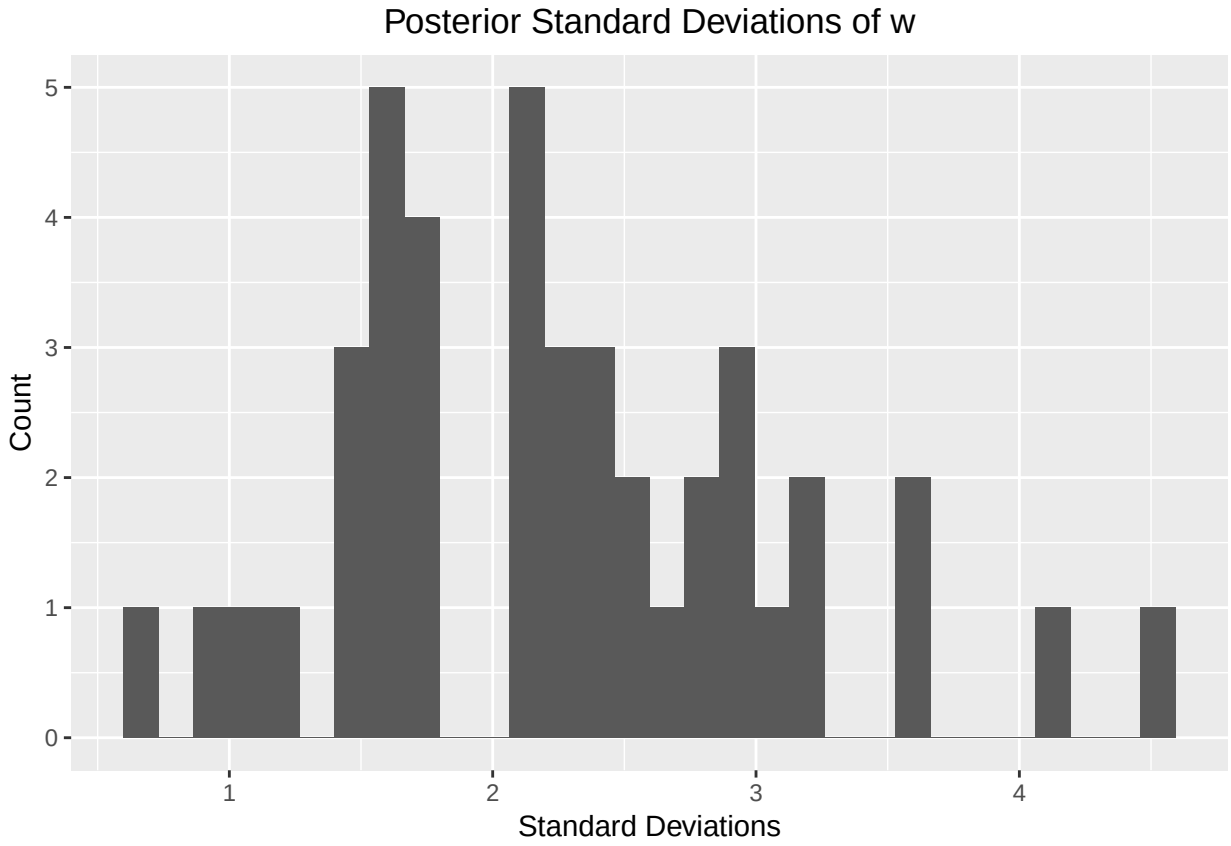

beta
1.8
1.6
1.4
1.2
500
1000
1500
2000
2500
chain
1
2
3
4

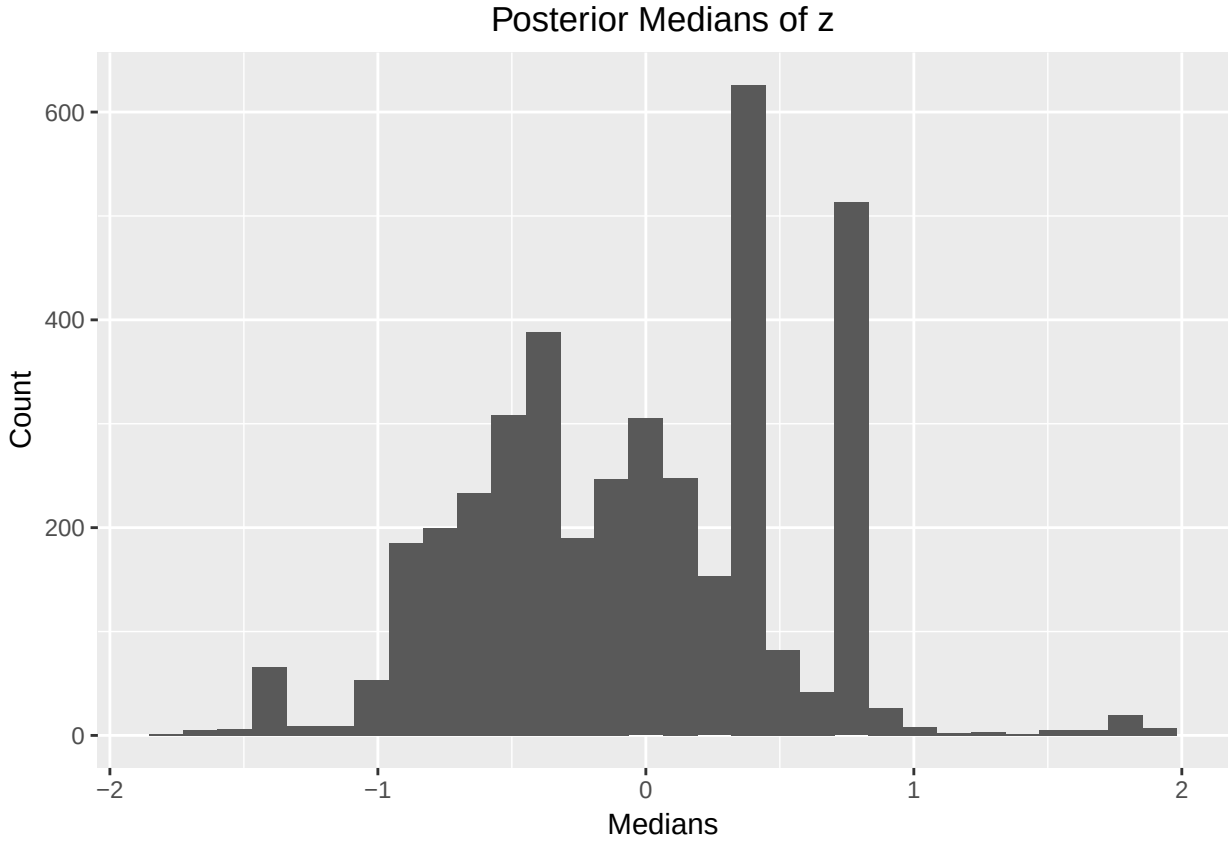



```
## [1] "       "
```

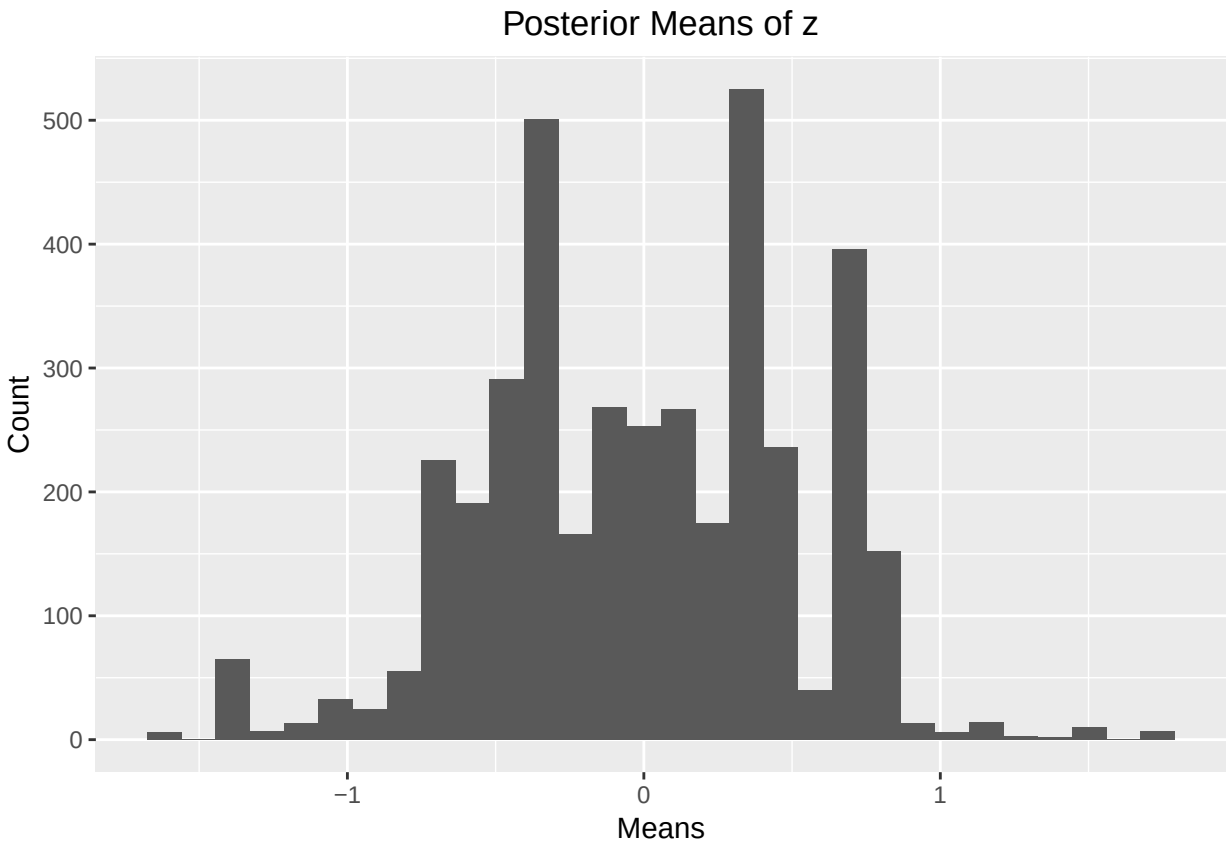



```
## [1] "      "
```

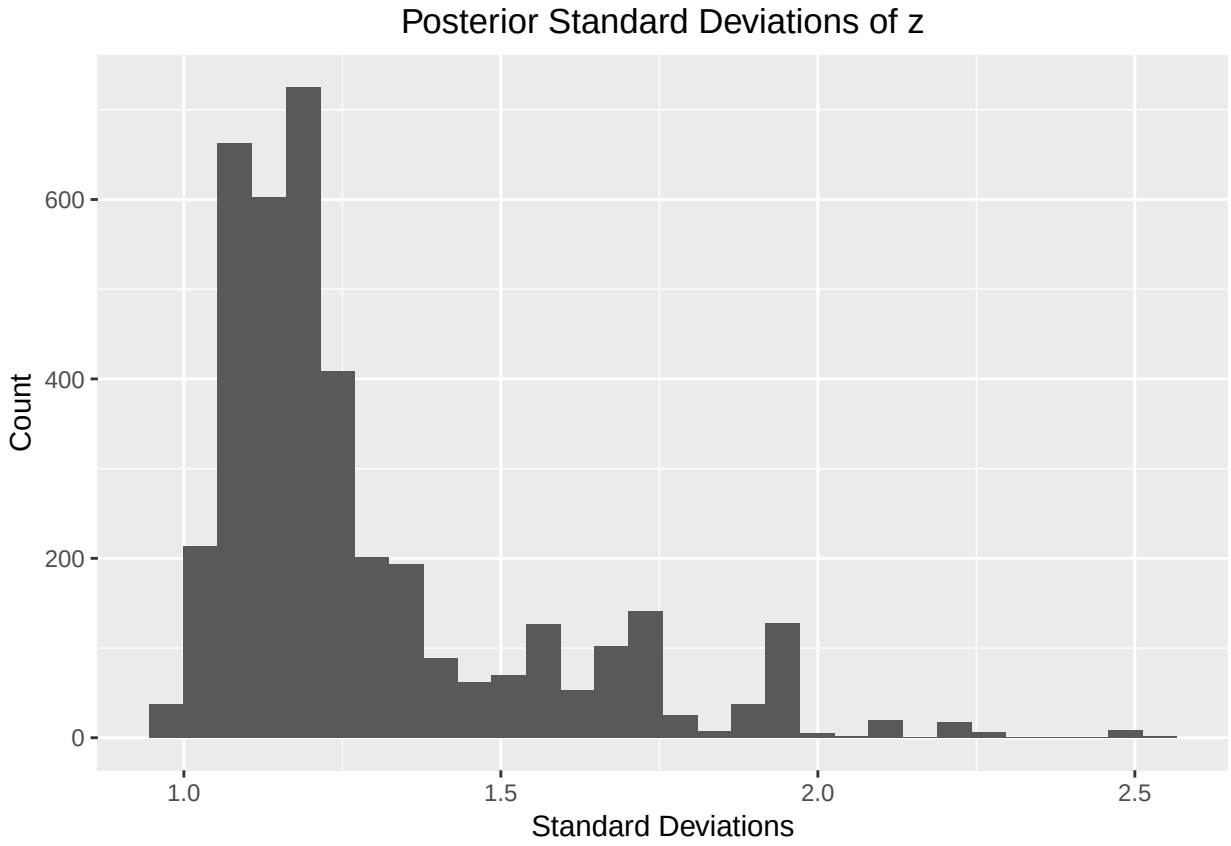

Posterior Standard Deviations of w
Count
0
1
2
3
4
5
1
2
3
4
Standard Deviations

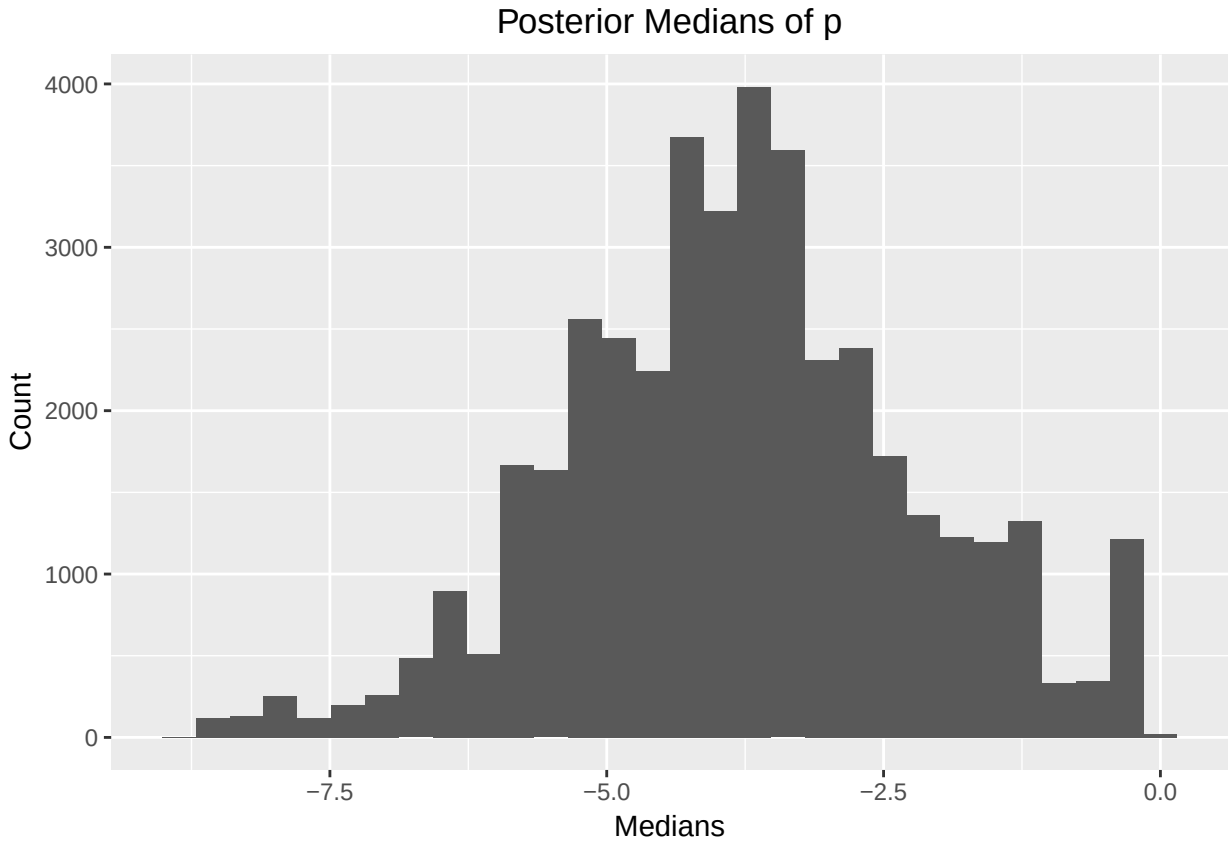

Posterior Medians of z
Count
0
200
400
600
−2
−1
0
1
2
Medians

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

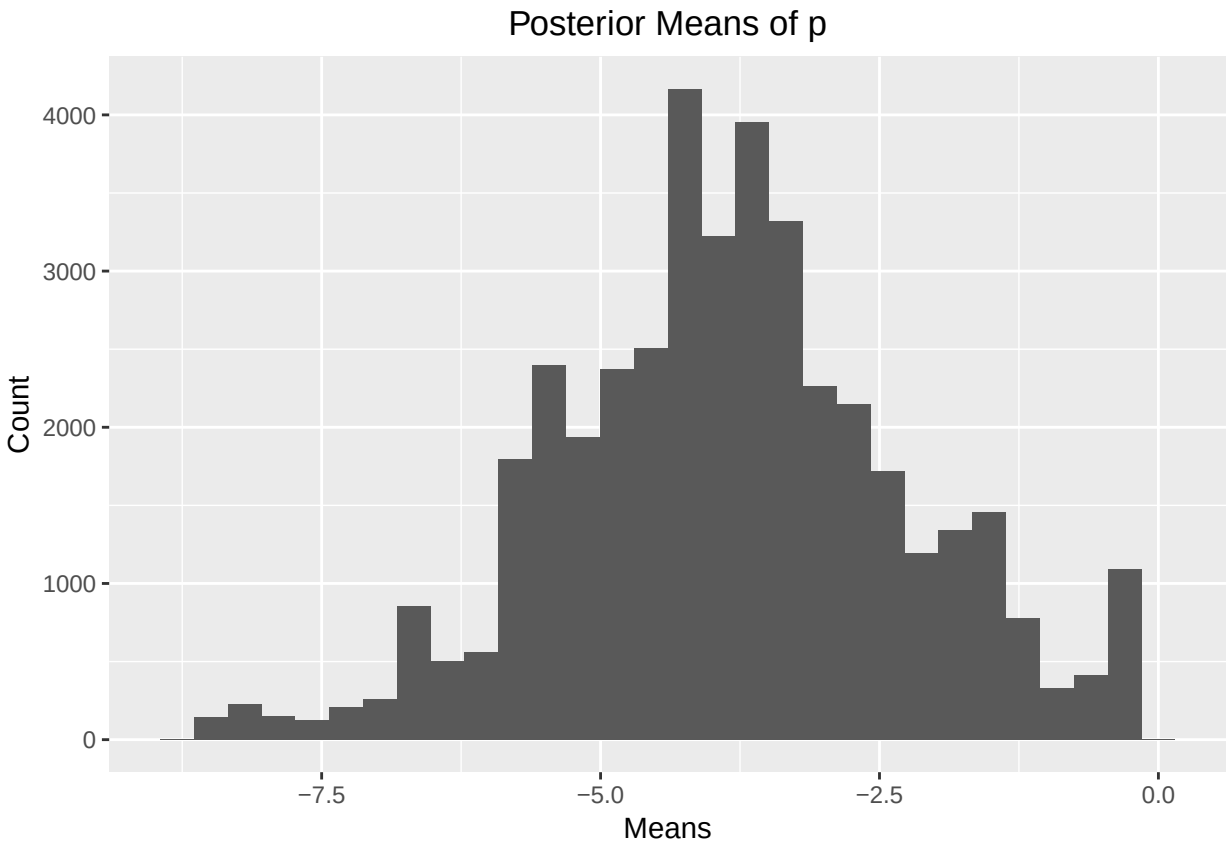



```
## [1] "       "
```

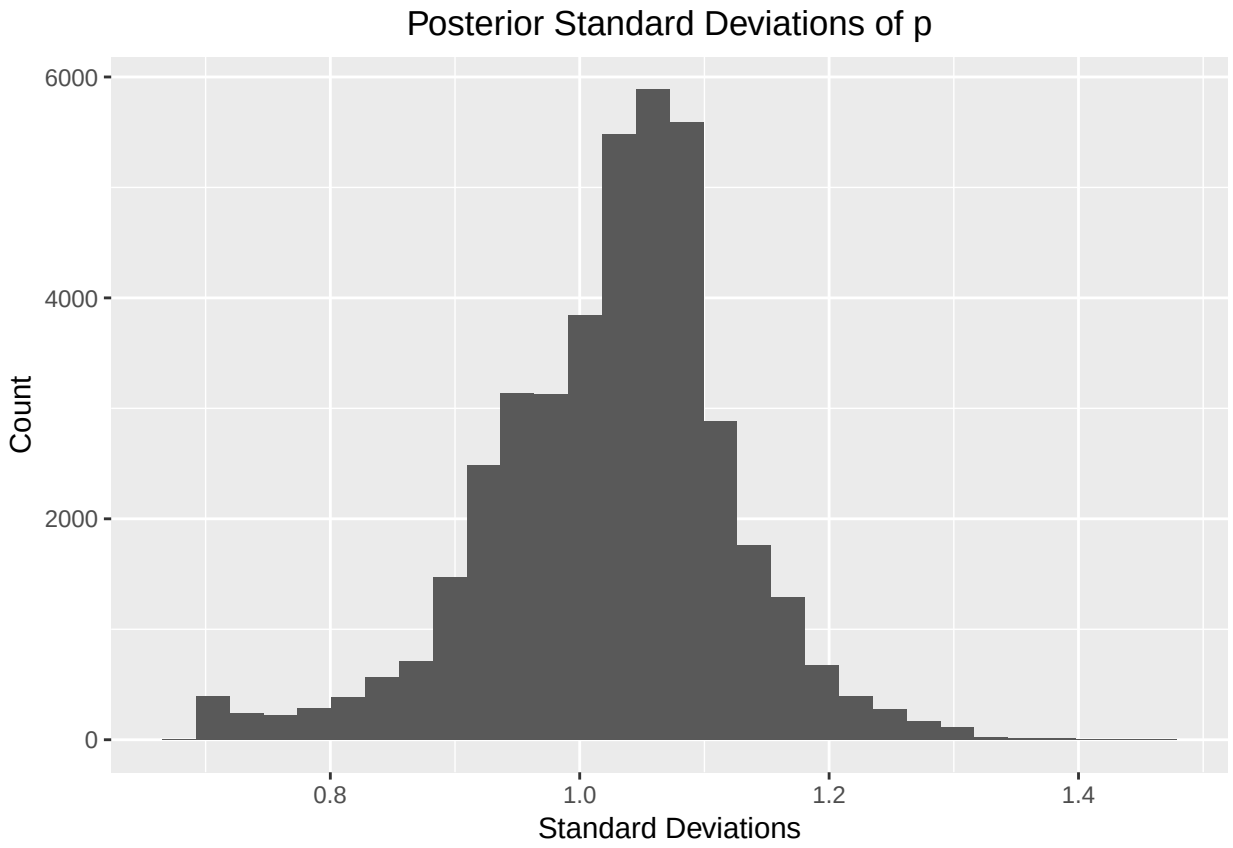

Posterior Standard Deviations of z
Count
600
400
200
0
1.0
1.5
2.0
2.5
Standard Deviations


Posterior Medians of p
Count
4000
3000
2000
1000
0
−7.5
−5.0
−2.5
0.0
Medians

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


Posterior Means of p
Count
4000
3000
2000
1000
0
−7.5
−5.0
−2.5
0.0
Means

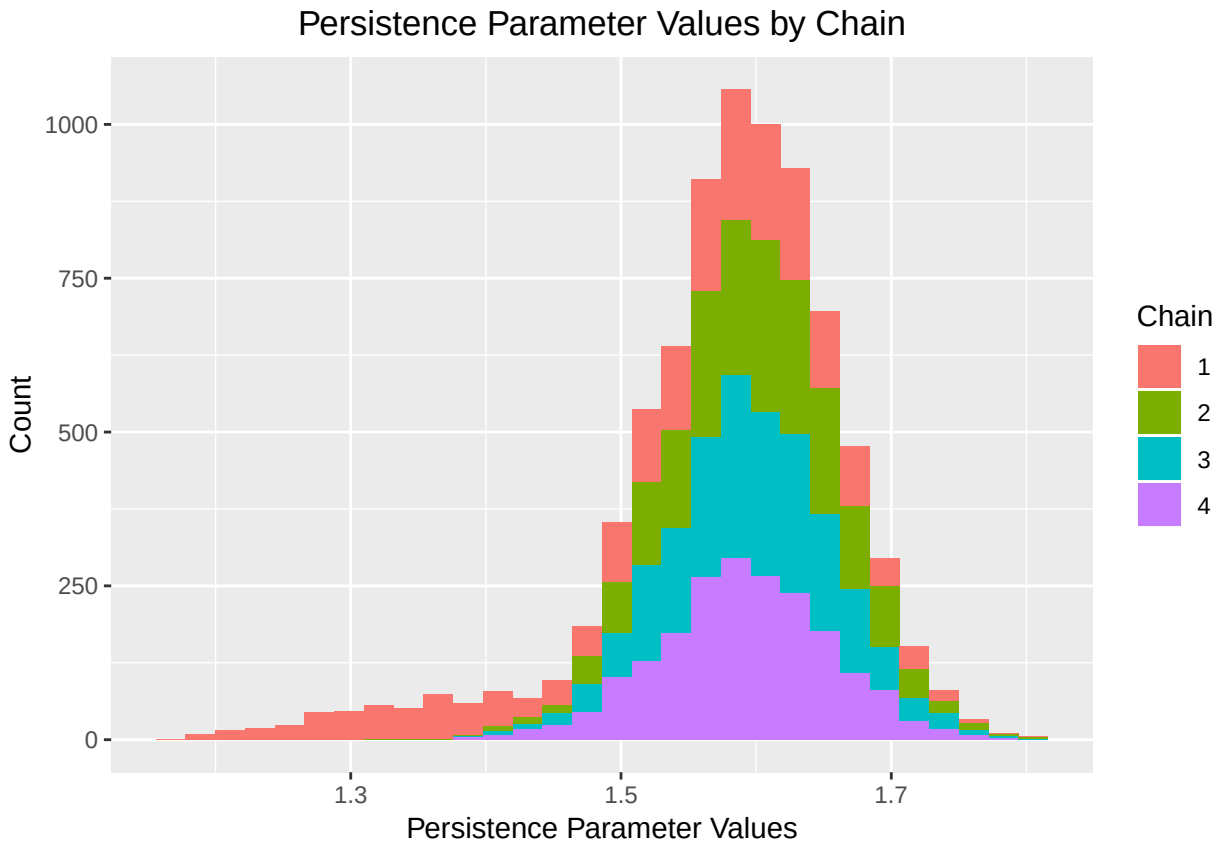
```
## [1] "       "
```



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

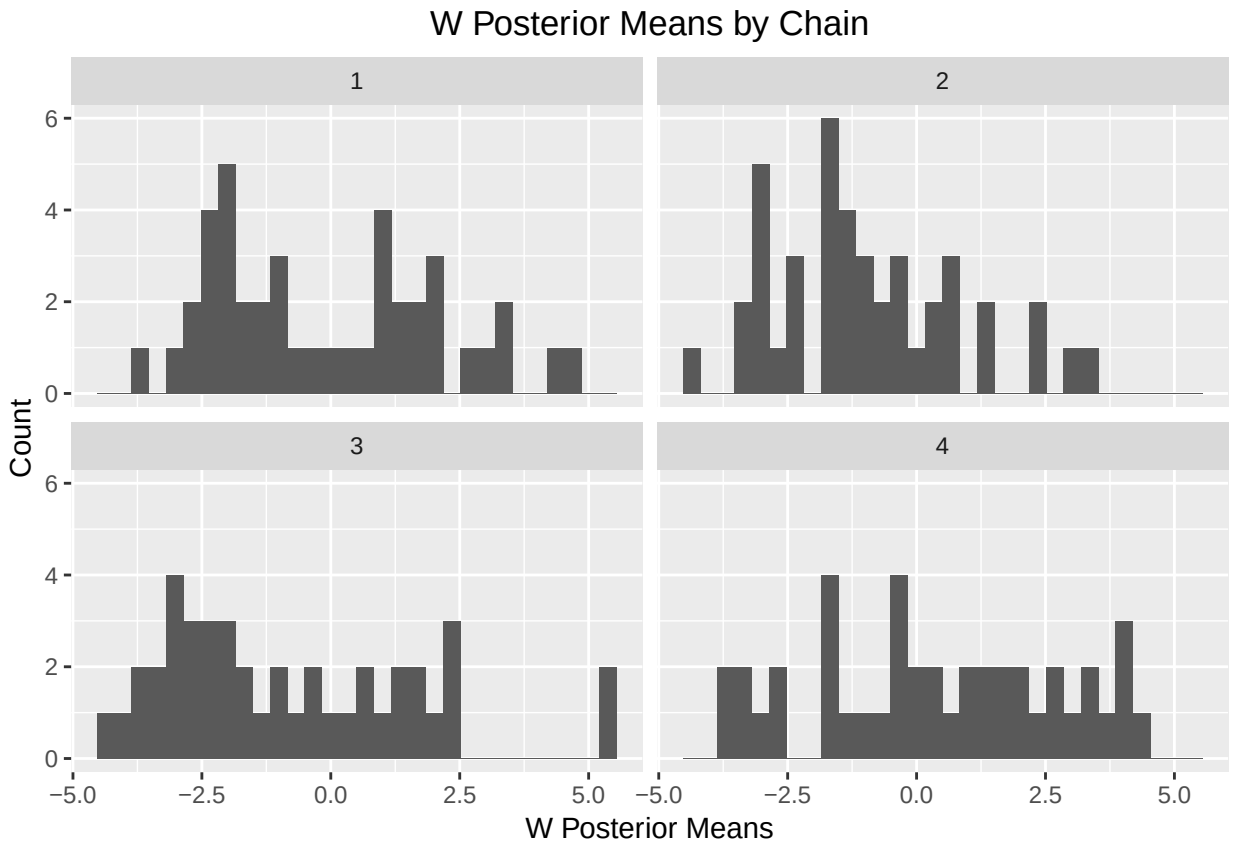
```
draws <- as.matrix(fit)
beta_col <- draws[,which(colnames(draws) == "beta")]
l <- length(beta_col)/4
plot_dat <- data.frame(val = beta_col, Chain = as.factor(c(rep(1,l),rep(2,l),rep(3,l),rep(4,l))))
ggplot(plot_dat, aes(x = val, fill = Chain)) + geom_histogram() +
  xlab("Persistence Parameter Values")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

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



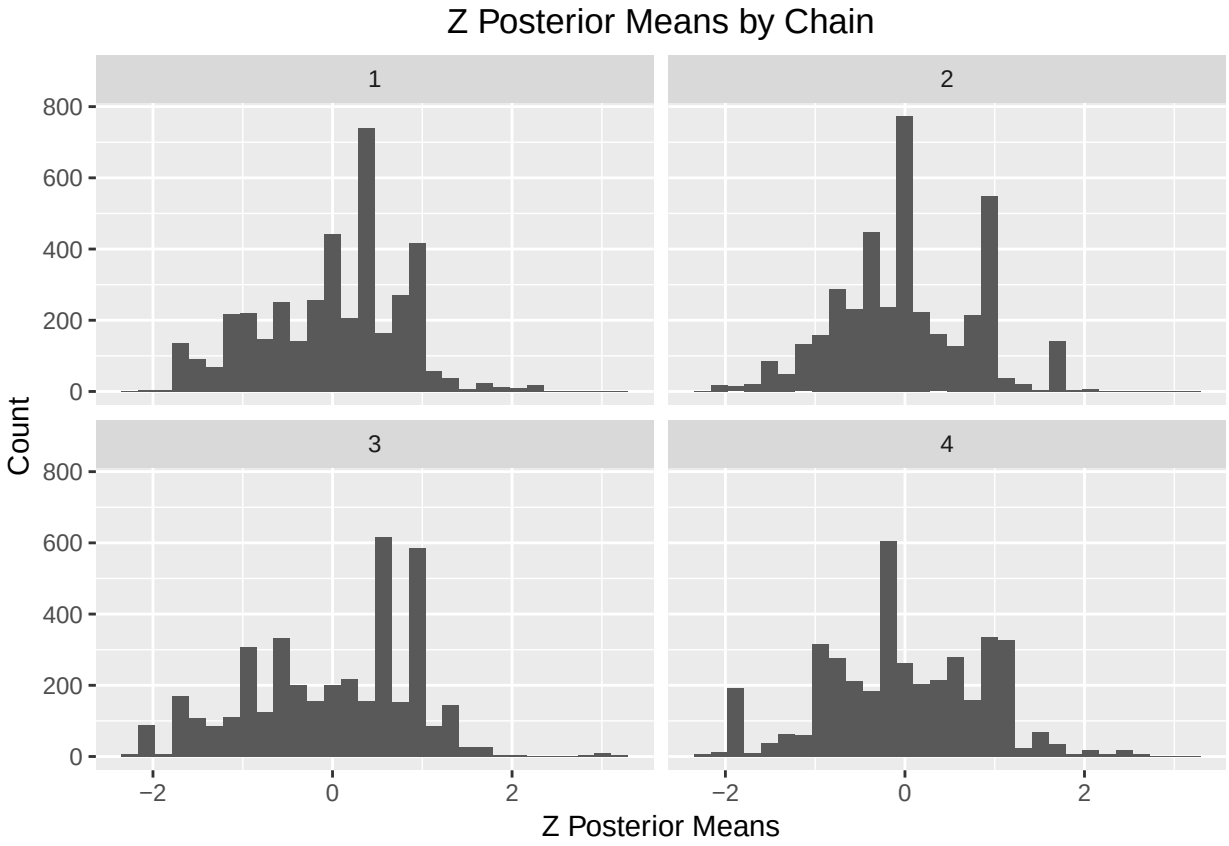

```
w_ind <- grep("^w", colnames(draws))
z_ind <- grep("^z", colnames(draws))
w_dat <- data.frame(chain1 = colMeans(draws[1:l,w_ind]),
                    chain2 = colMeans(draws[(l+1):(2*l),w_ind]),
                    chain3 = colMeans(draws[(2*l+1):(3*l),w_ind]),
                    chain4 = colMeans(draws[(3*l+1):(4*l),w_ind])) %>%
  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
800
600
400
200
0
−2
0
2
Z Posterior Means