MCMC Diagnostics - IFLS data

Sarah Teichman

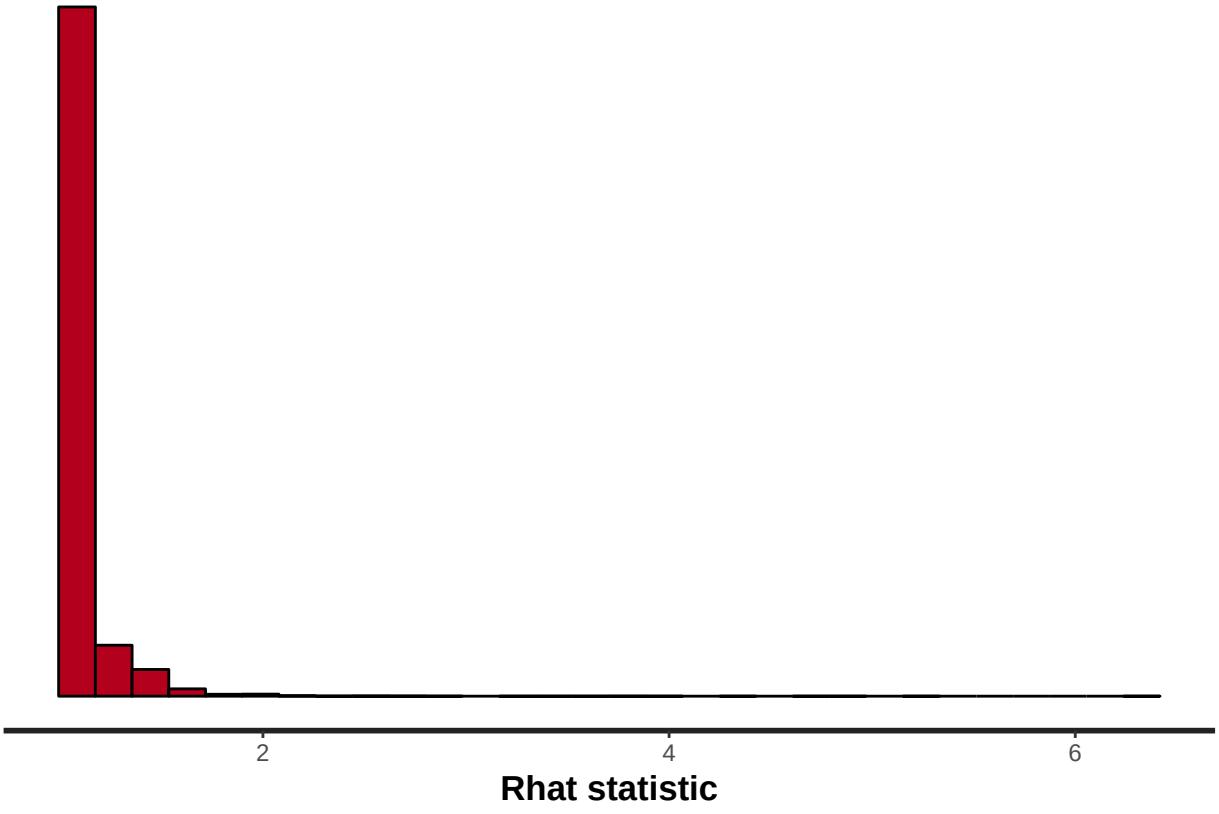
07/03/2020

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

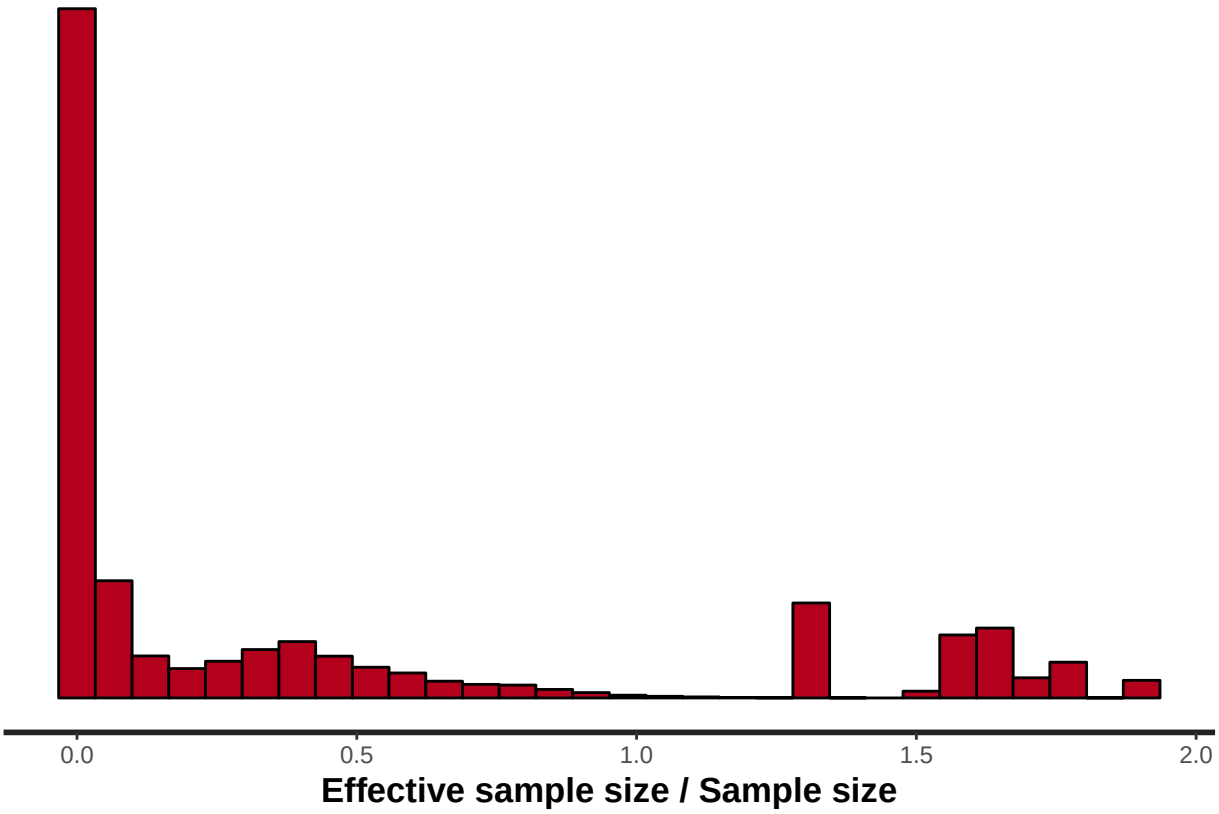
General MCMC diagnostic plots

Overall model diagnostics from rstan package.

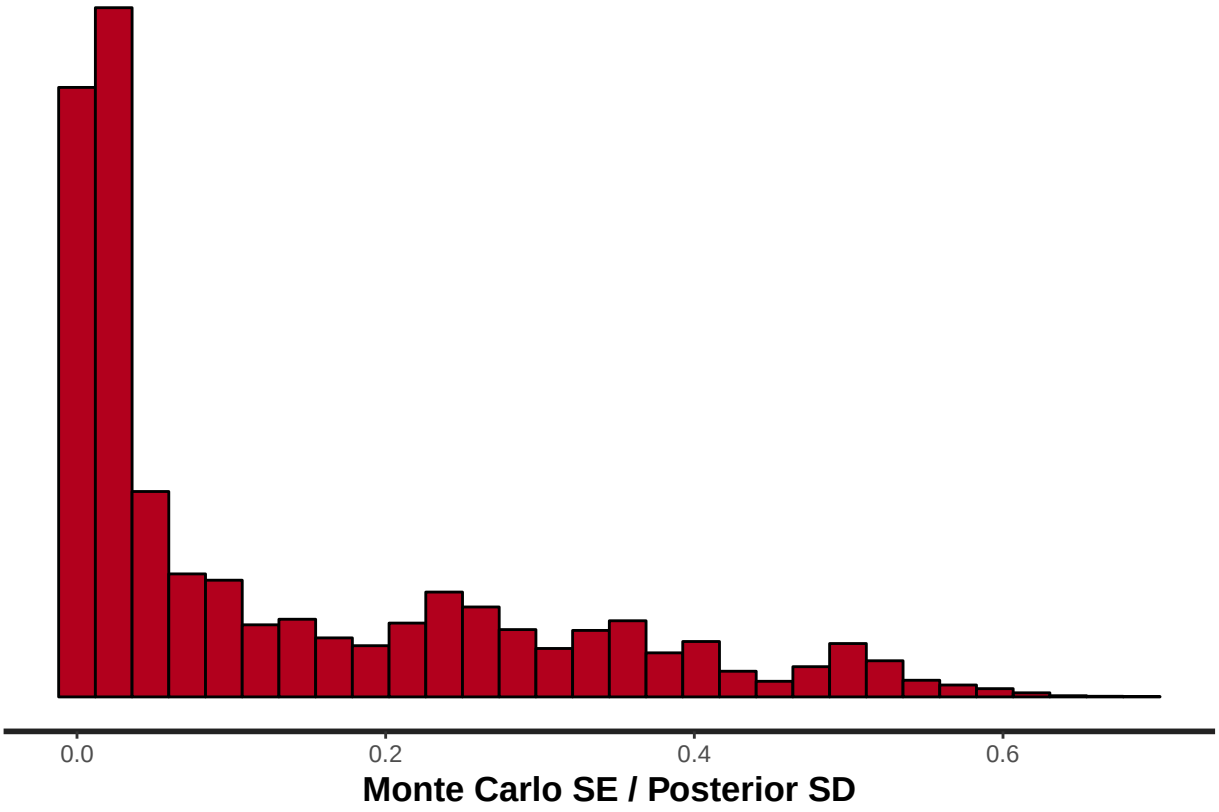
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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




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



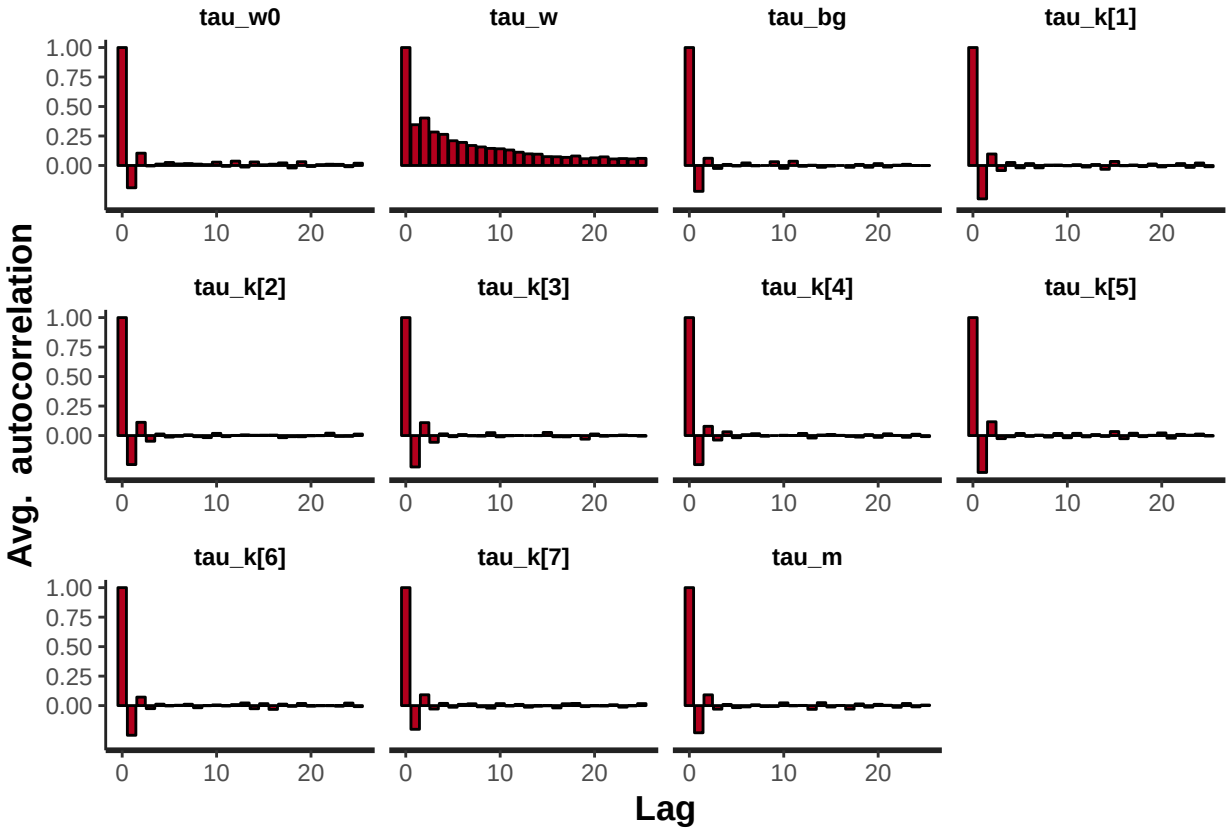
Individual Parameter Diagnostics

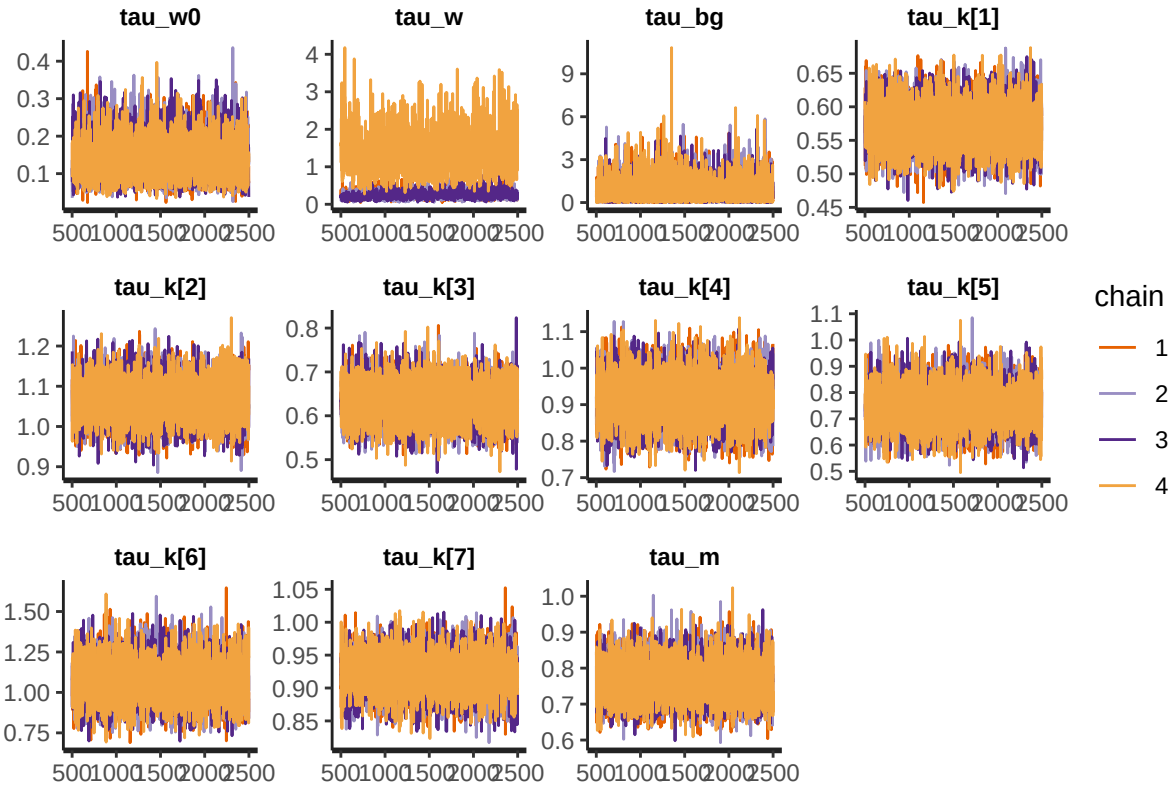
Individual parameter plots. Autocorrelation and trace plots for individual parameters, and histograms of posterior medians for group parameters.

```
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)
```

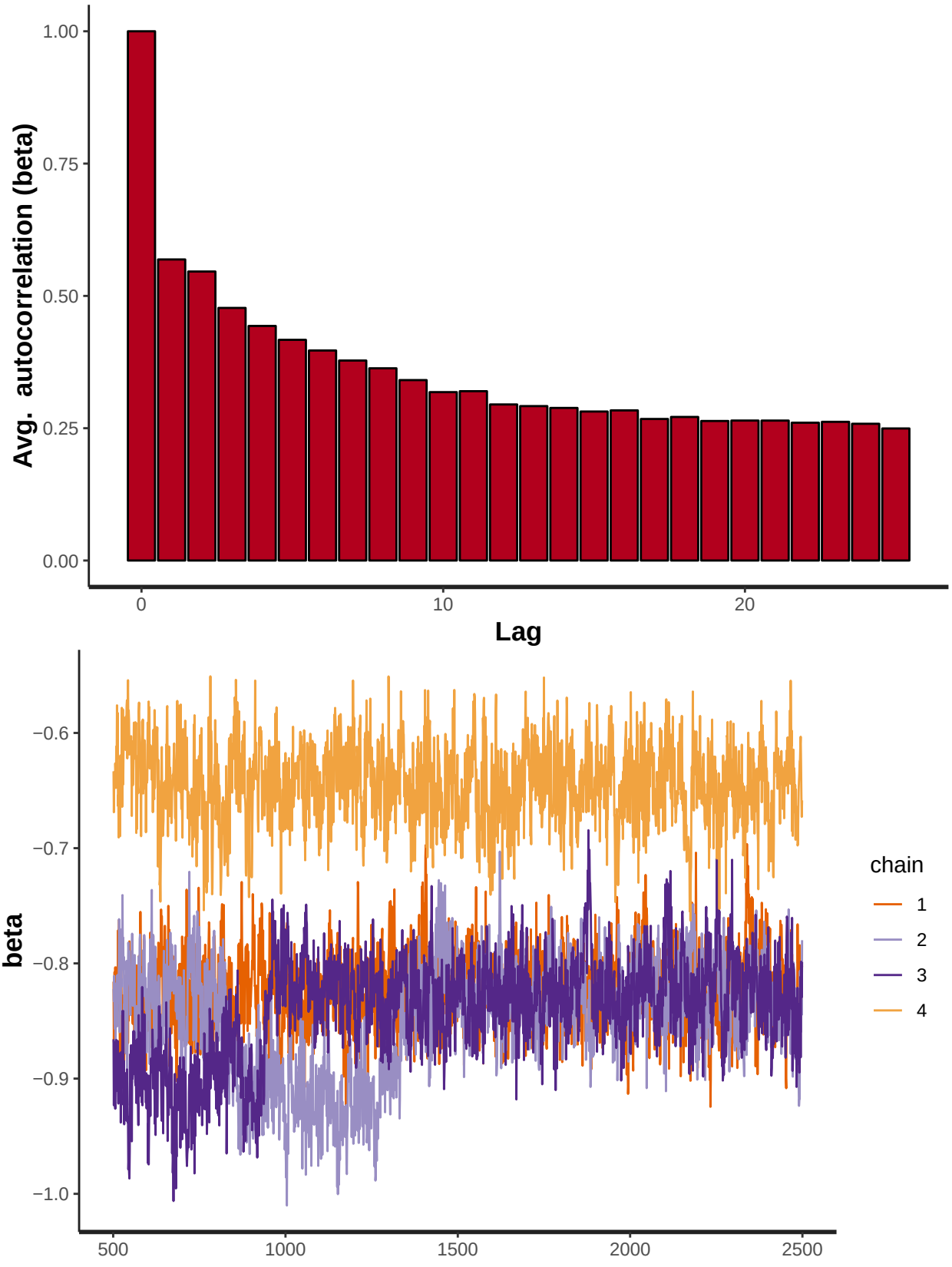
```
##               mean      se_mean         sd       25%       50%       75%
## tau_w0   0.1431145 0.0008341015 0.05297717 0.1052708 0.1364742 0.1736763
## tau_w    0.5222429 0.3450176795 0.55981819 0.1899544 0.2715353 0.5834897
## tau_bg   0.7522141 0.0072043372 0.75316486 0.2216128 0.5316443 1.0417759
## tau_k[1] 0.5711358 0.0002736120 0.03300329 0.5480292 0.5705812 0.5932533
## tau_k[2] 1.0649953 0.0004353608 0.04900747 1.0312505 1.0645345 1.0981637
## tau_k[3] 0.6291215 0.0003683748 0.04270456 0.5997756 0.6280650 0.6574208
## tau_k[4] 0.9132148 0.0005523809 0.06225600 0.8713508 0.9116937 0.9540119
## tau_k[5] 0.7433875 0.0006374770 0.07549018 0.6911024 0.7411050 0.7935560
## tau_k[6] 1.0634557 0.0011270530 0.12770829 0.9746875 1.0567068 1.1463794
## tau_k[7] 0.9206064 0.0002868427 0.02963514 0.9004979 0.9201333 0.9402973
## tau_m    0.7628096 0.0004874233 0.05365819 0.7251235 0.7610445 0.7971188
```

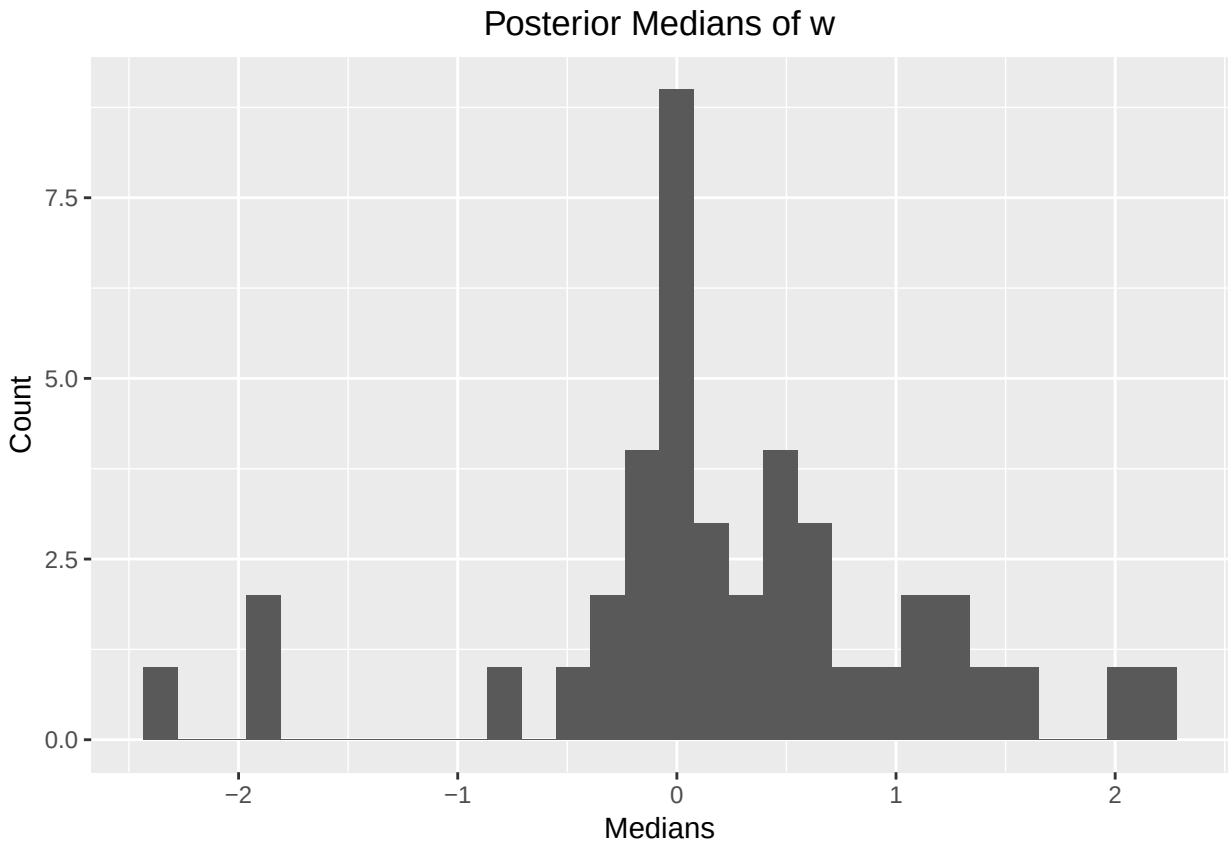
```
##                n_eff      Rhat
## tau_w0    4034.03502 1.0038033
## tau_w        2.63276 1.9960403
## tau_bg   10929.29216 1.0011862
## tau_k[1] 14549.37169 0.9999421
## tau_k[2] 12671.43800 0.9996696
## tau_k[3] 13439.05512 0.9998668
## tau_k[4] 12702.37844 0.9996259
## tau_k[5] 14023.35083 0.9997434
## tau_k[6] 12839.54141 1.0000381
## tau_k[7] 10673.97995 1.0001565
## tau_m    12118.79211 0.9998072
```



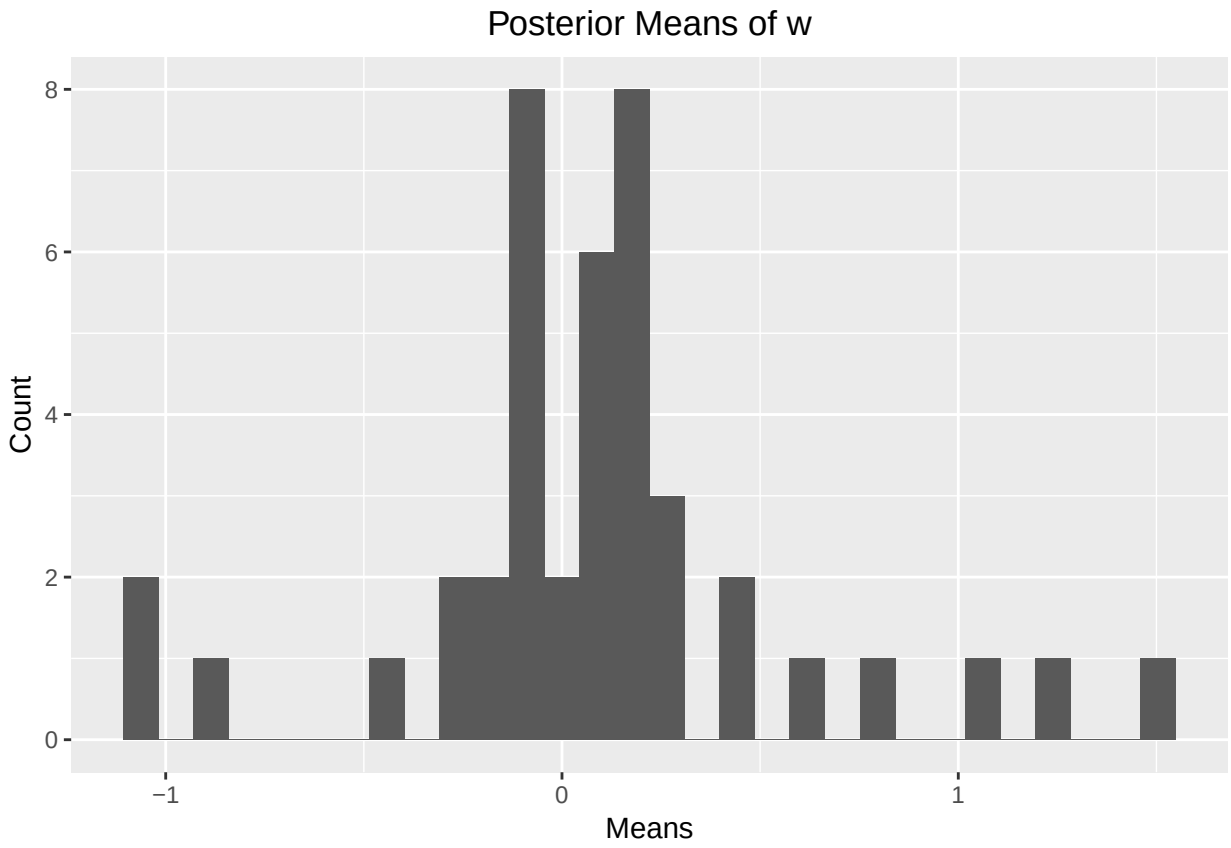



```
##             mean    se_mean        sd        25%        50%        75%
## beta -0.7893808 0.06030111 0.0935455 -0.8495605 -0.8170002 -0.7296511
##          n_eff    Rhat
## beta 2.406551 2.53102
```

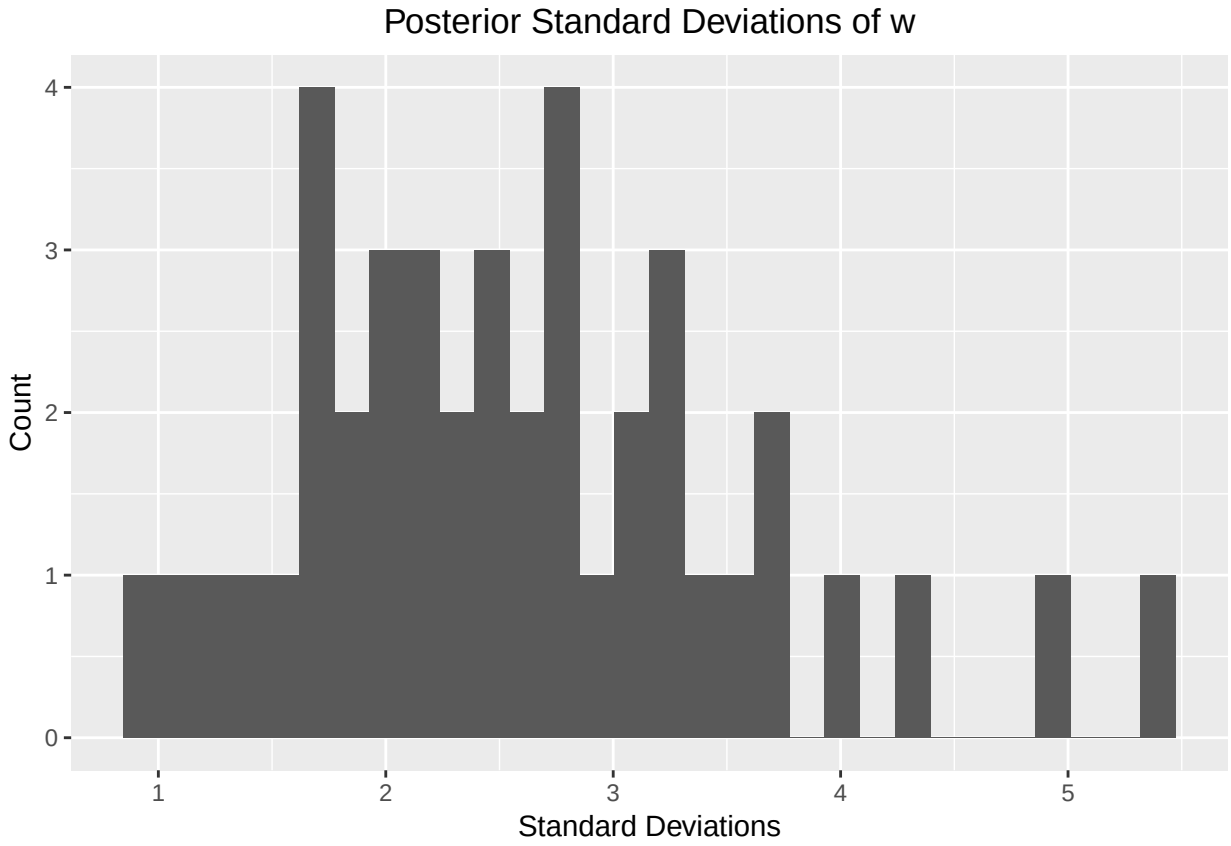

Avg. autocorrelation (beta)
1.00
0.75
0.50
0.25
0.00
0
10
20
Lag
beta
−0.6
−0.7
−0.8
−0.9
−1.0
500
1000
1500
2000
2500
chain
1
2
3
4

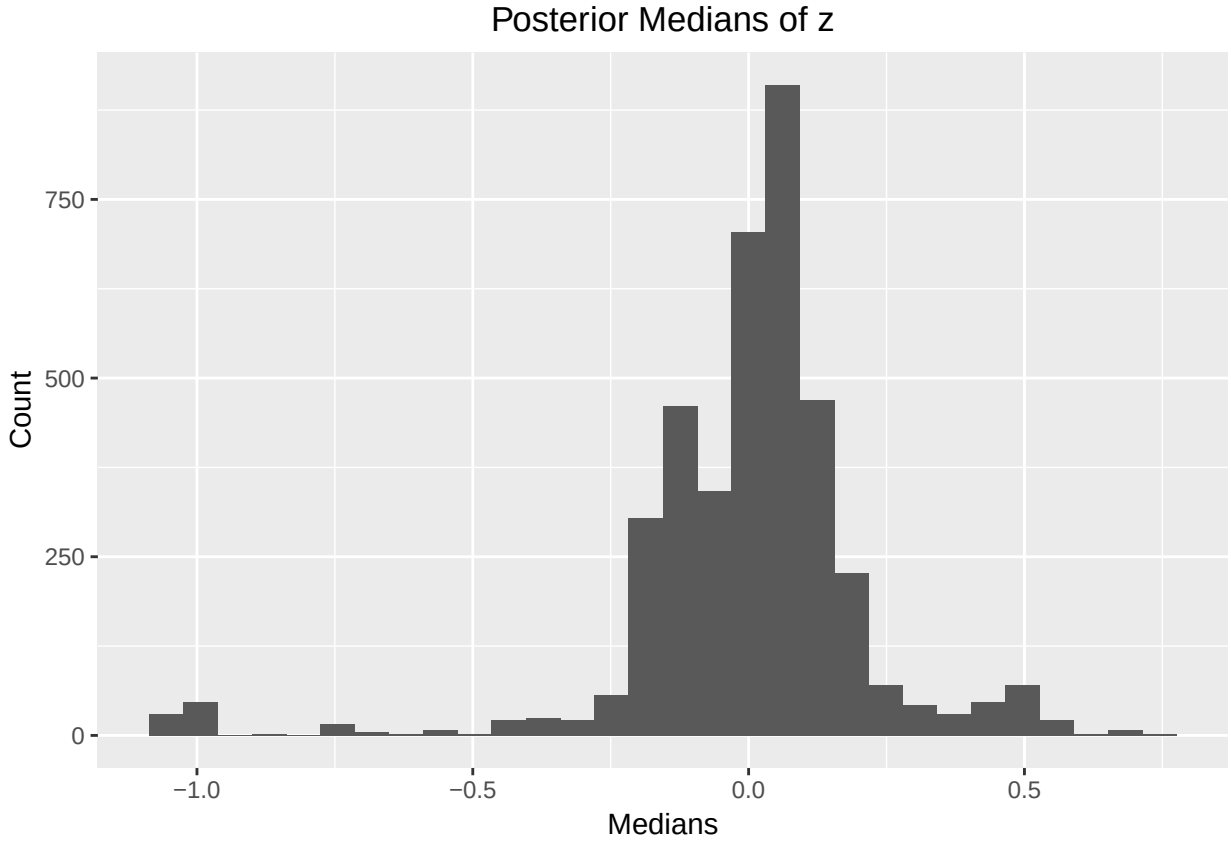



```
## [1] "       "
```

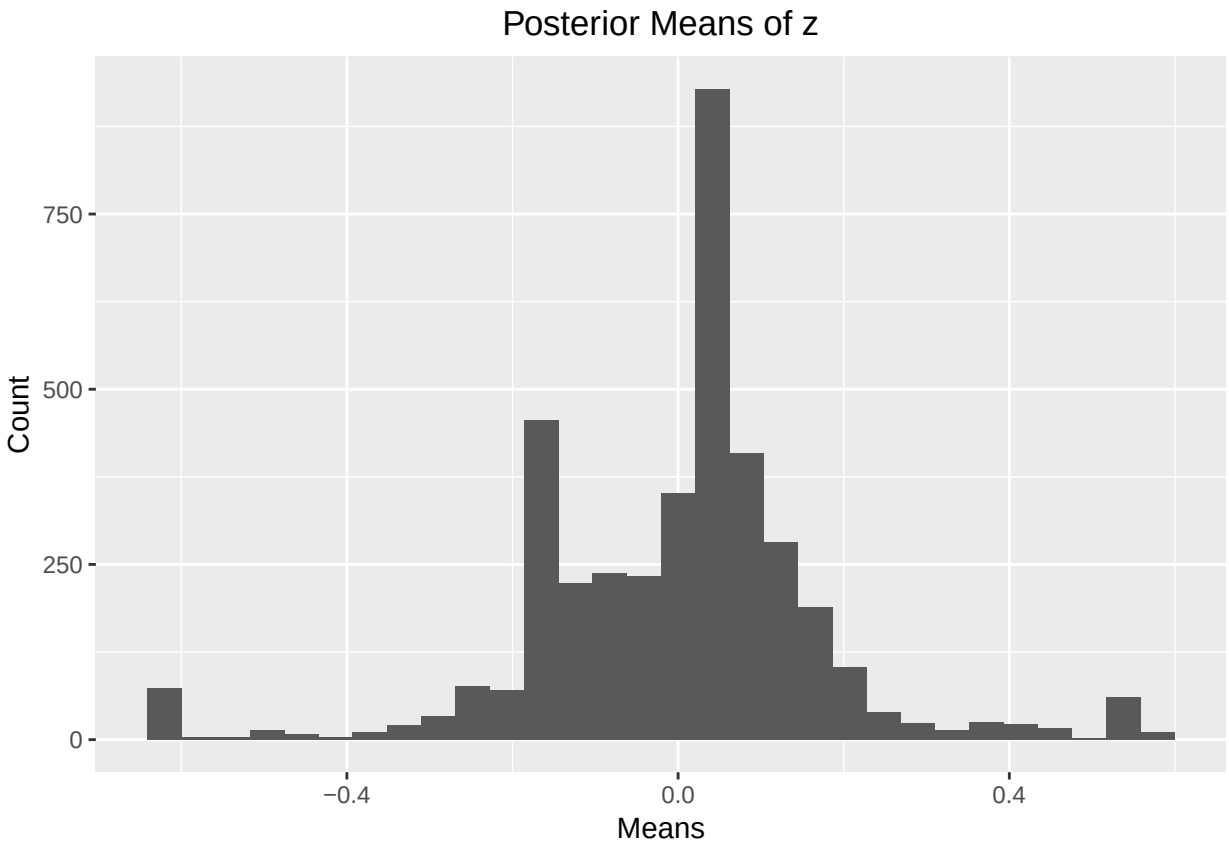



```
## [1] "      "
```

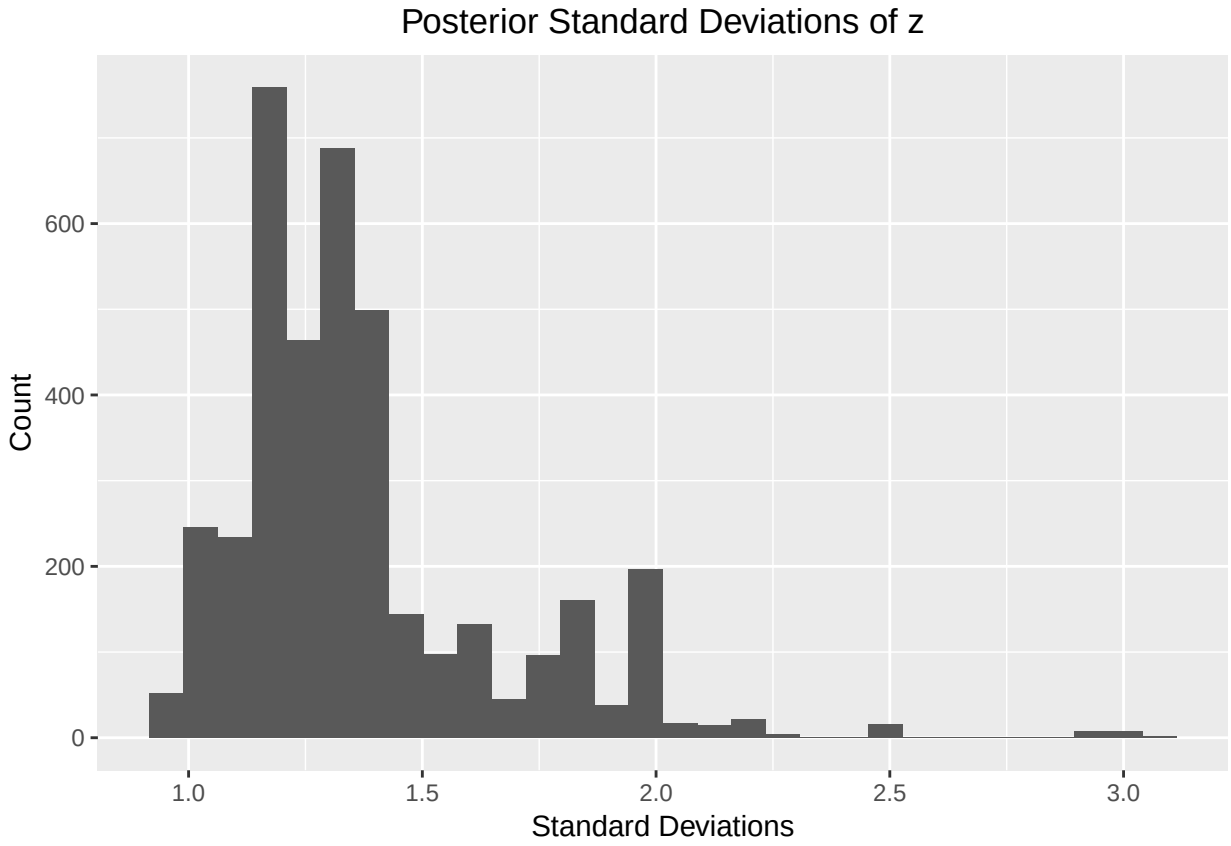

Posterior Standard Deviations of w
Count
0
1
2
3
4
1
2
3
4
5
Standard Deviations

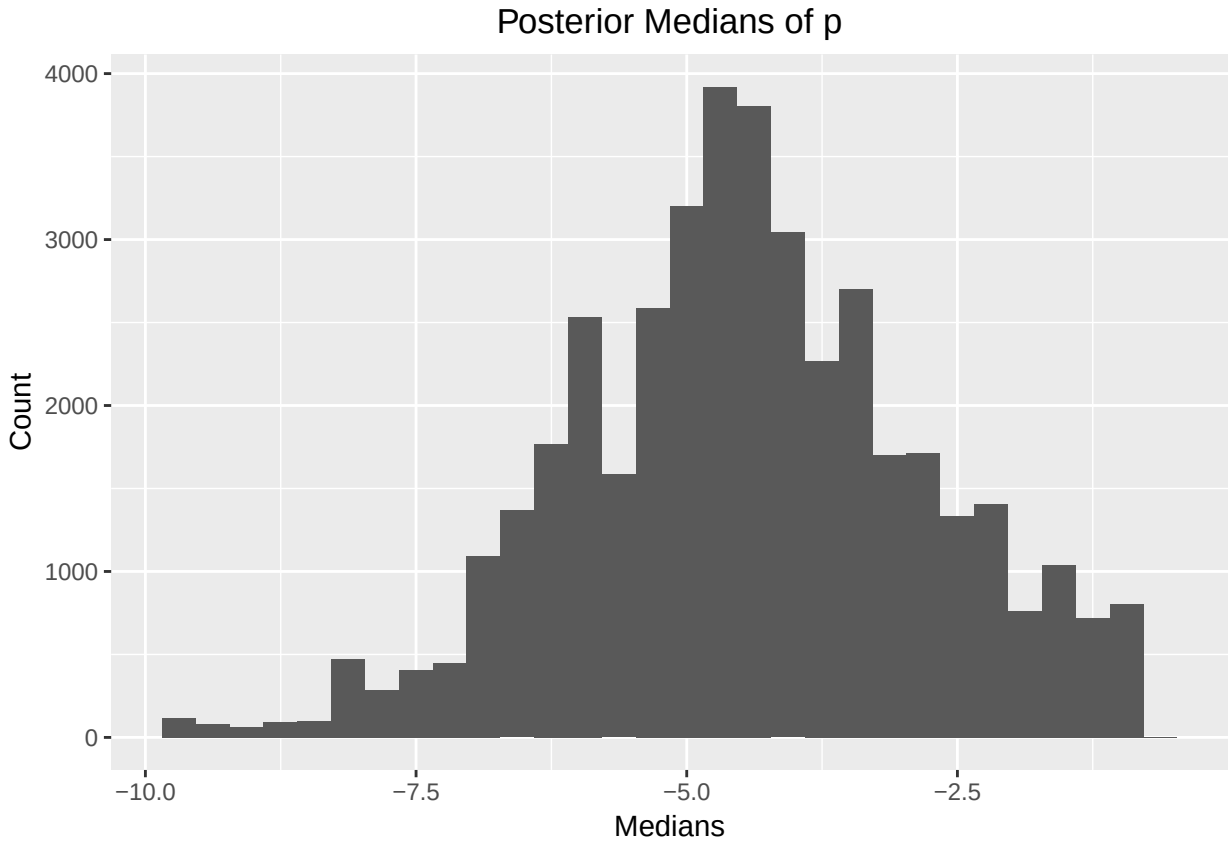

Posterior Medians of z
Count
0
250
500
750
−1.0
−0.5
0.0
0.5
Medians

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

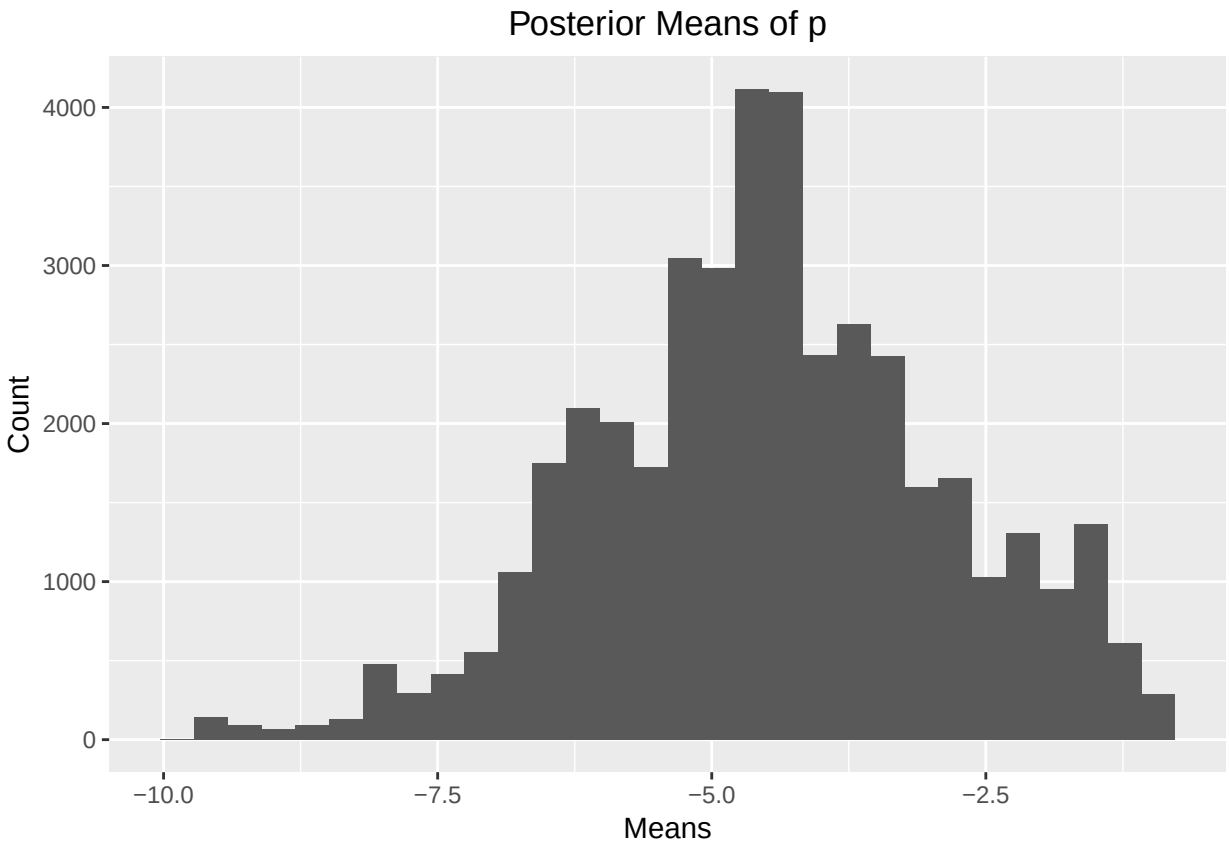



```
## [1] "       "
```

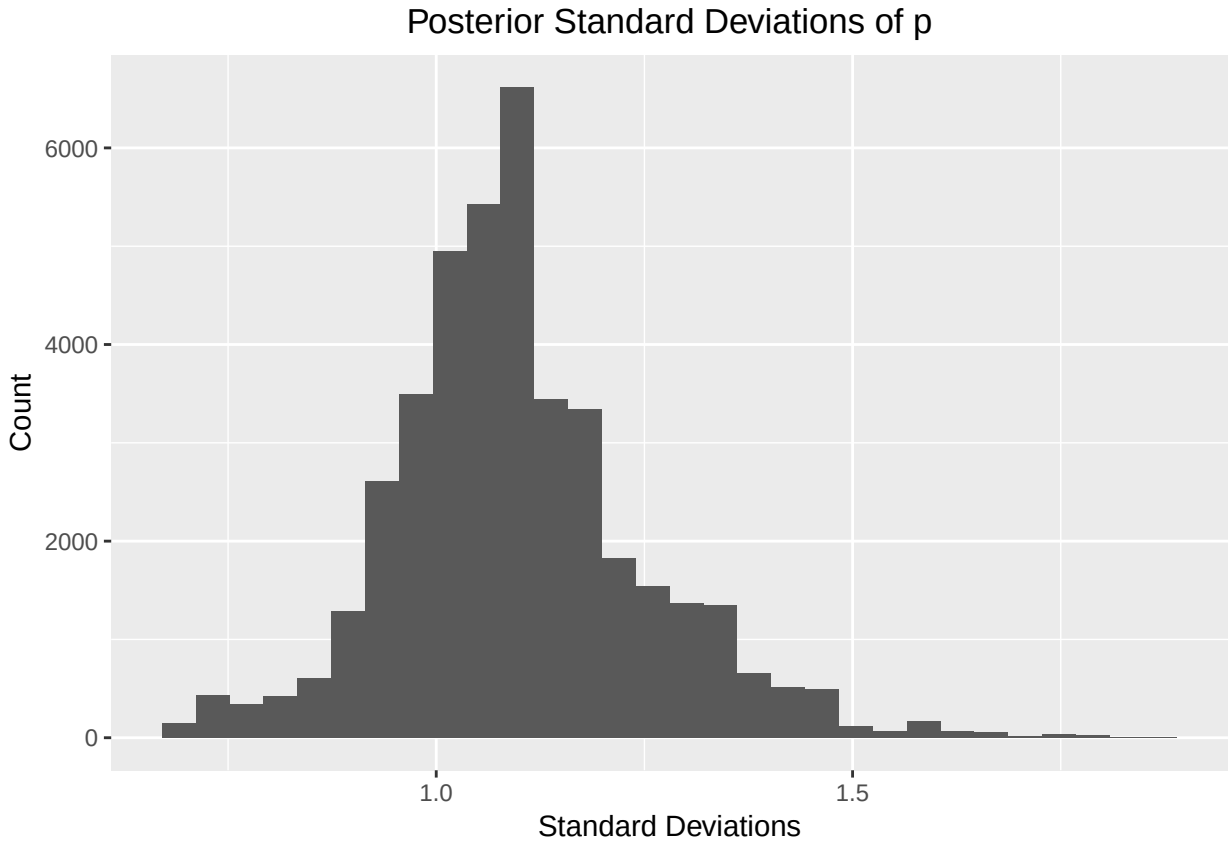

Posterior Standard Deviations of z
Count
600
400
200
0
1.0
1.5
2.0
2.5
3.0
Standard Deviations


Posterior Medians of p
Count
4000
3000
2000
1000
0
−10.0
−7.5
−5.0
−2.5
Medians

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


Posterior Means of p
Count
4000
3000
2000
1000
0
−10.0
−7.5
−5.0
−2.5
Means

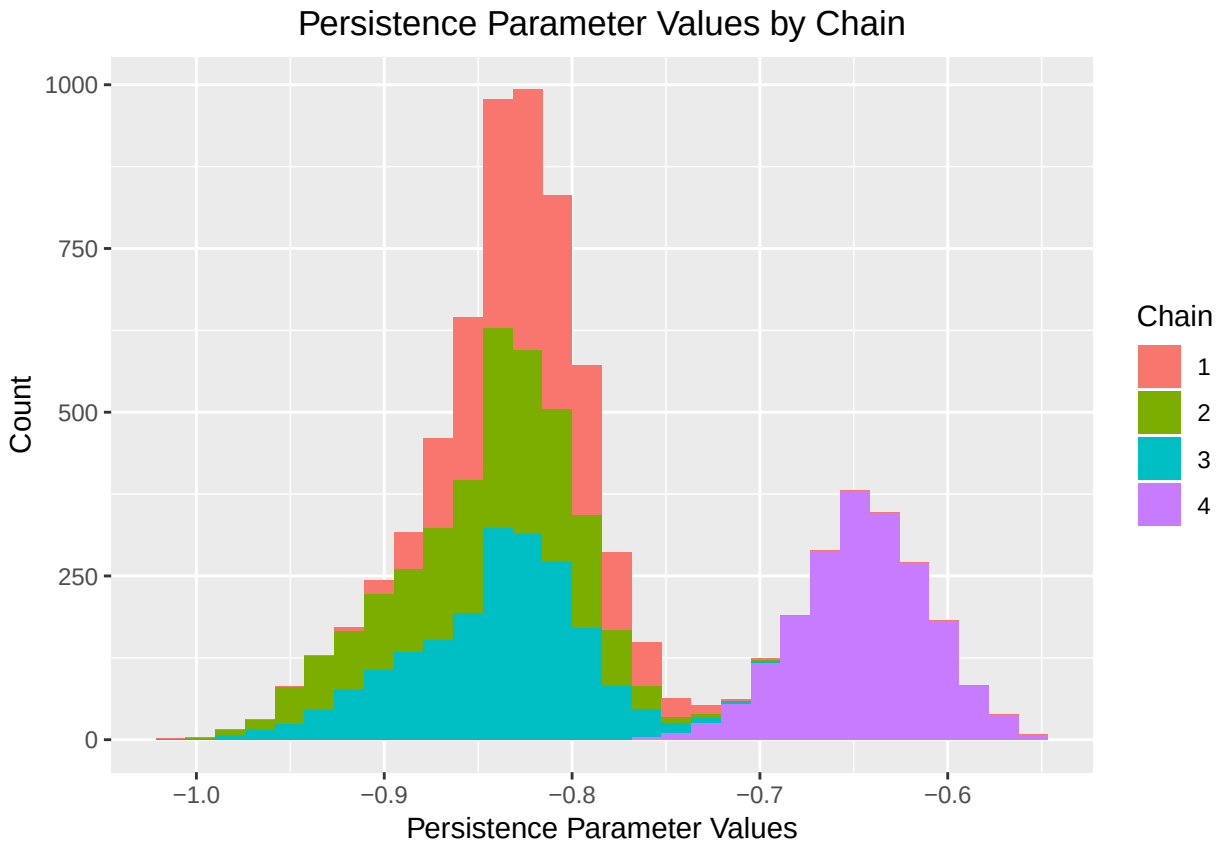
```
## [1] "       "
```



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

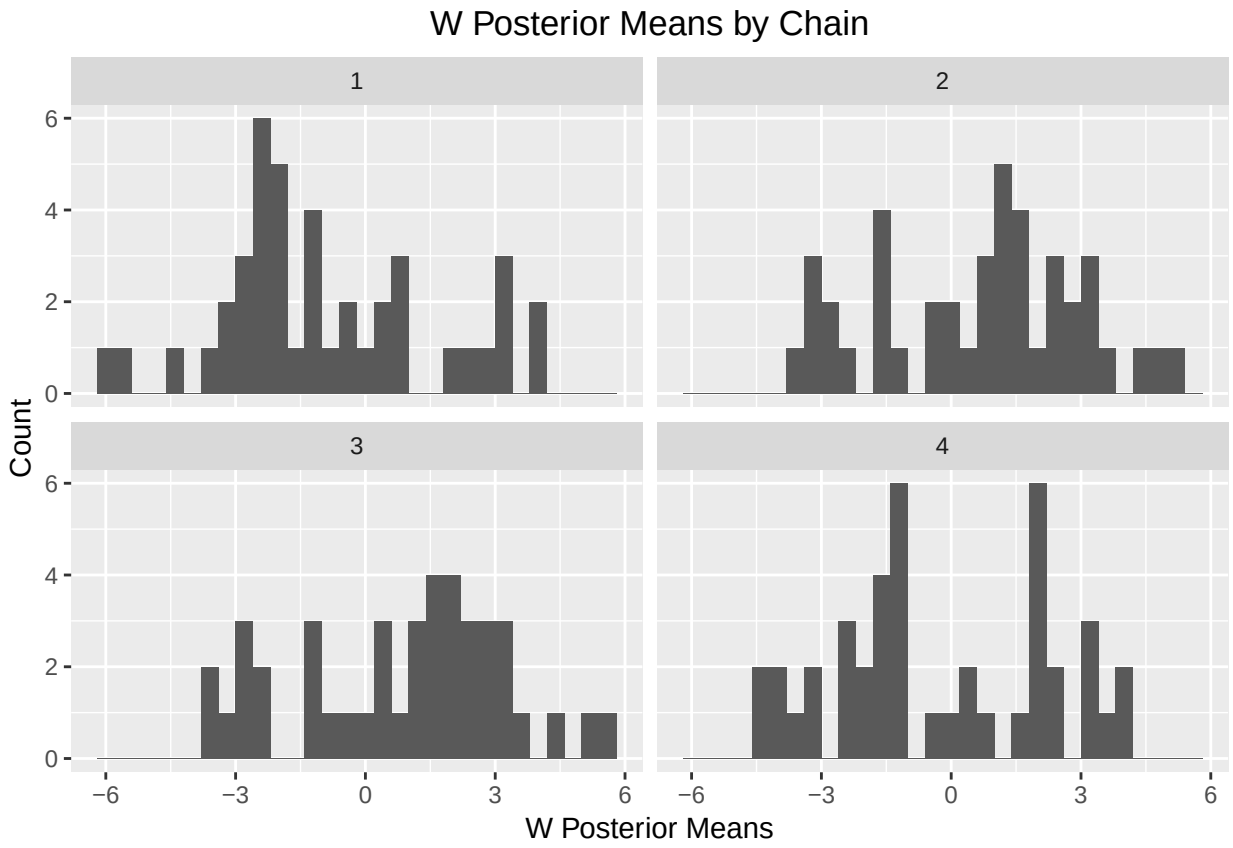
```
draws <- as.matrix(fit)
beta_col <- draws[,which(colnames(draws) == "beta")]
l <- length(beta_col)/4
plot_dat <- data.frame(val = beta_col, Chain = as.factor(c(rep(1,l),rep(2,l),rep(3,l),rep(4,l))))
ggplot(plot_dat, aes(x = val, fill = Chain)) + geom_histogram() +
  xlab("Persistence Parameter Values")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

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



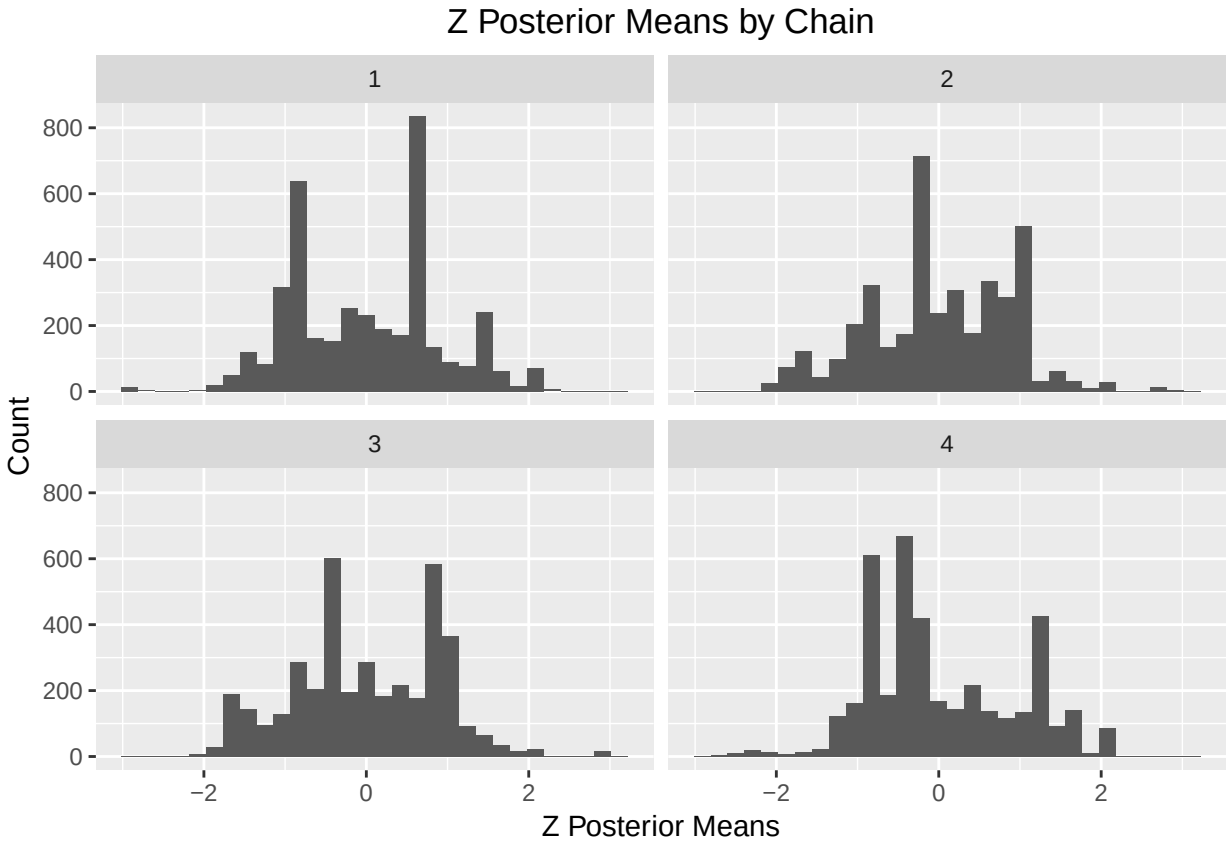

```
w_ind <- grep("^w", colnames(draws))
z_ind <- grep("^z", colnames(draws))
w_dat <- data.frame(chain1 = colMeans(draws[1:l,w_ind]),
                    chain2 = colMeans(draws[(l+1):(2*l),w_ind]),
                    chain3 = colMeans(draws[(2*l+1):(3*l),w_ind]),
                    chain4 = colMeans(draws[(3*l+1):(4*l),w_ind]))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