Version 2: 2 clusters; Fix cluster label switching u,v; unif; fix w [0,1]; only force ordering of means in 1 sample; S4

Simulate data

```
I <- 50
K <- 2
S <- 10

# choose diffuse priors for gamma
a_gamma <- 2
b_gamma <- 10

avrg <- a_gamma * b_gamma
std.dv <- sqrt(a_gamma*b_gamma^2)
g_range = seq(0, avrg + 5*std.dv, 0.01)
g_y = dgamma(g_range, a_gamma, rate = 1/b_gamma)
#plot(g_range, g_y, type = "l", ylim=c(0, max(g_y) + 0.01))

set.seed(123)

a <- matrix(NA, nrow=K, ncol=S)
b <- matrix(NA, nrow=K, ncol=S)
for (s in 1:S) {
  a[, s] <- rgamma(K, a_gamma, rate = 1/b_gamma)
  b[, s] <- rgamma(K, a_gamma, rate = 1/b_gamma)
}

# reorder a,b matrices to match ordering of means (U) in S1
U <- a/(a+b)
V <- a+b
U.ordered <- U[order(U[,1]), ]
a.ordered <- a[order(U[,1]), ]
b.ordered <- b[order(U[,1]), ]
V.ordered <- V[order(U[,1]), ]

pi <- as.vector(rdirichlet(1, rep(1, K)))
z <- sample(1:K, size = I, replace = T, prob = pi)

w <- matrix(NA, nrow=I, ncol=S)
for (s in 1:S) {
  w[, s] <- rbeta(I, a.ordered[,s][z], b.ordered[,s][z])
}

tcn <- matrix(2, nrow=I, ncol=S)
m <- matrix(rep(sample(1:2, size = I, replace = T), S), nrow=I, ncol=S)

calcTheta <- function(m, tcn, w) {
```

```
  (m * w) / (tcn * w + 2*(1-w))
}
theta <- calcTheta(m, tcn, w)

n <- replicate(S, rpois(I, 100))
y <- matrix(NA, nrow=I, ncol=S)
for (i in 1:I) {
  for (s in 1:S) {
    y[i, s] <- rbinom(1, n[i, s], theta[i,s])
  }
}

#n_S2 <- subset(n, select = 2)
#y_S2 <- subset(y, select = 2)
#m_S2 <- subset(m, select = 2)
#tcn_S2 <- subset(tcn, select = 2)
n_S4 <- n[ , 4]
y_S4 <- y[ , 4]
m_S4 <- m[ , 4]
tcn_S4 <- tcn[ , 4]
w_S4 <- w[ ,4]
S <- 1
```

JAGS

```
jags.file <- file.path(models.dir, "v2_uv_unif_fix2_1sample.jags")

test.data <- list("I" = I, "K" = K,
                  "y" = y_S4, "n" = n_S4,
                  "m" = m_S4, "tcn" = tcn_S4)
jags.m <- jags.model(jags.file, test.data,
                     n.chains = 1,
                     inits = list(".RNG.name" = "base::Wichmann-Hill",
                                  ".RNG.seed" = 123))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 50
##    Unobserved stochastic nodes: 105
##    Total graph size: 899
##
## Initializing model
```

```
params <- c("z", "w", "U", "V")
samps <- coda.samples(jags.m, params, n.iter=6000, thin = 6)
s <- summary(samps)
effectiveSize(samps)
```

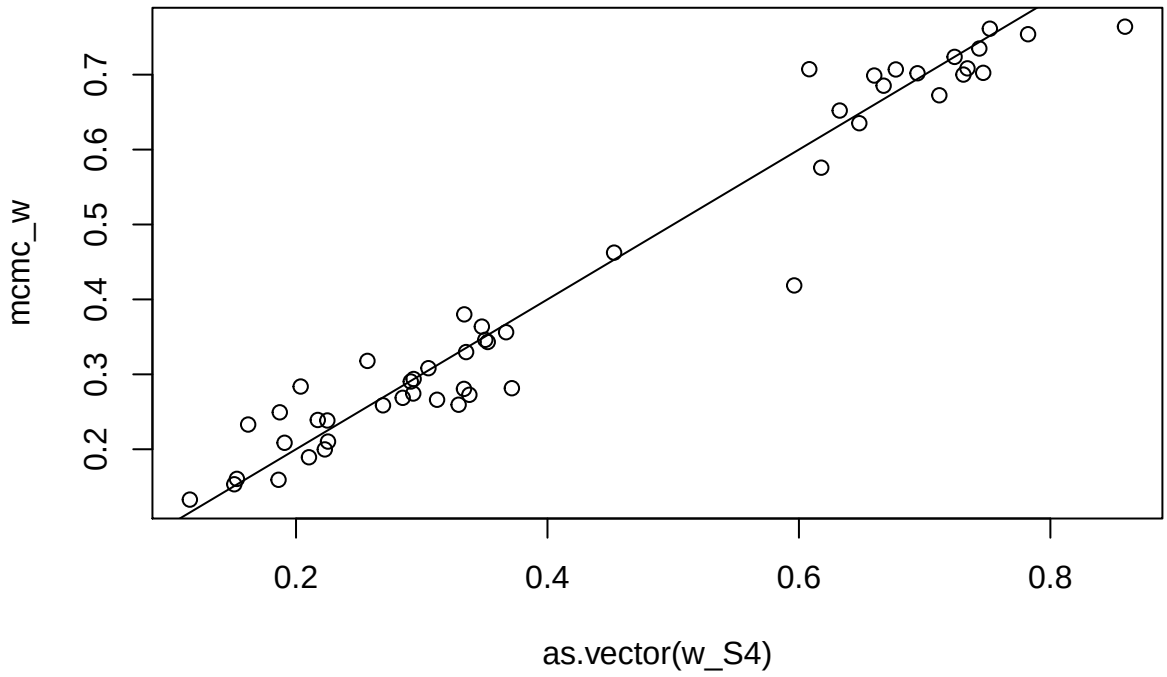
```
##      U[1]      U[2]      V[1]      V[2]      w[1]      w[2]      w[3]
##  369.6086  510.9256  621.2790  395.9346 1000.0000 1000.0000  883.2969
##      w[4]      w[5]      w[6]      w[7]      w[8]      w[9]     w[10]
```

```
## 1000.0000  817.4796  581.6779 1000.0000  889.2857 1000.0000  733.2111
##     w[11]     w[12]     w[13]     w[14]     w[15]     w[16]     w[17]
## 1000.0000  776.0705 1000.0000 1000.0000 1000.0000 1000.0000  848.6387
##     w[18]     w[19]     w[20]     w[21]     w[22]     w[23]     w[24]
## 1000.0000 1157.0446 1000.0000 1000.0000  476.2898  962.6961 1000.0000
##     w[25]     w[26]     w[27]     w[28]     w[29]     w[30]     w[31]
## 1000.0000 1000.0000  908.4275 1000.0000 1000.0000 1000.0000 1000.0000
##     w[32]     w[33]     w[34]     w[35]     w[36]     w[37]     w[38]
## 1000.0000 1049.8637  881.3053 1108.4798 1000.0000 1000.0000  877.5472
##     w[39]     w[40]     w[41]     w[42]     w[43]     w[44]     w[45]
## 1000.0000 1000.0000  788.2947  424.0328 1000.0000 1000.0000 1000.0000
##     w[46]     w[47]     w[48]     w[49]     w[50]      z[1]      z[2]
## 1000.0000  769.7950 1103.4107 1000.0000 1000.0000  700.5075  546.4961
##      z[3]      z[4]      z[5]      z[6]      z[7]      z[8]      z[9]
##  812.3743  604.7060  686.7910  480.1575  737.3282 1000.0000  492.6095
##     z[10]     z[11]     z[12]     z[13]     z[14]     z[15]     z[16]
##  502.0442  769.2462  573.9012 1000.0000 1000.0000 1000.0000  864.4790
##     z[17]     z[18]     z[19]     z[20]     z[21]     z[22]     z[23]
##  251.8960 1000.0000  489.2545  790.9361  813.4572  358.9591  664.4798
##     z[24]     z[25]     z[26]     z[27]     z[28]     z[29]     z[30]
##  722.4212  505.1167  644.3205  493.3241  485.3308  512.4778  443.0167
##     z[31]     z[32]     z[33]     z[34]     z[35]     z[36]     z[37]
##  569.2367  483.7014  557.5588 1000.0000 1000.0000  595.7060 1000.0000
##     z[38]     z[39]     z[40]     z[41]     z[42]     z[43]     z[44]
## 1000.0000    0.0000    0.0000  692.6165  401.4921 1000.0000  535.2766
##     z[45]     z[46]     z[47]     z[48]     z[49]     z[50]
##  739.6677 1000.0000  608.9926  883.0945  535.2766  589.5112
```

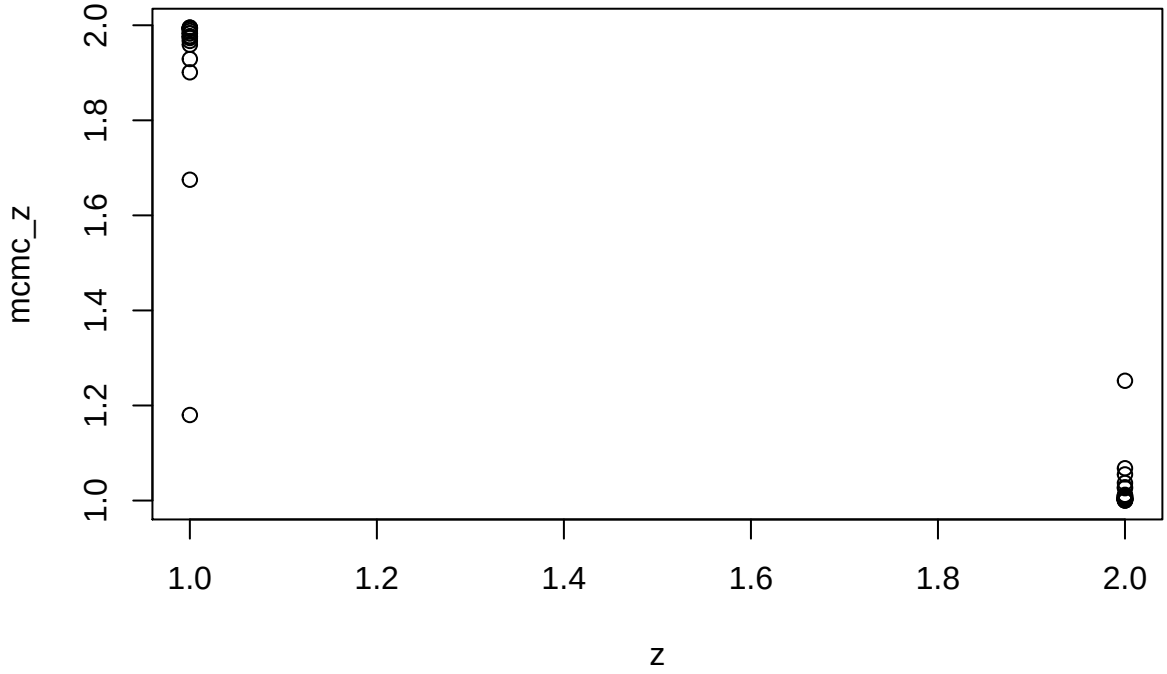
```
pdf(file.path(trace.dir, paste0(runName, "_trace.pdf")))
plot(samps)
dev.off()
```

```
## pdf
##   2
```

```
mcmc_vals <- s$statistics
mcmc_w <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "w", "Mean"]
plot(as.vector(w_S4), mcmc_w, type = "p")
abline(a=0, b=1)
```



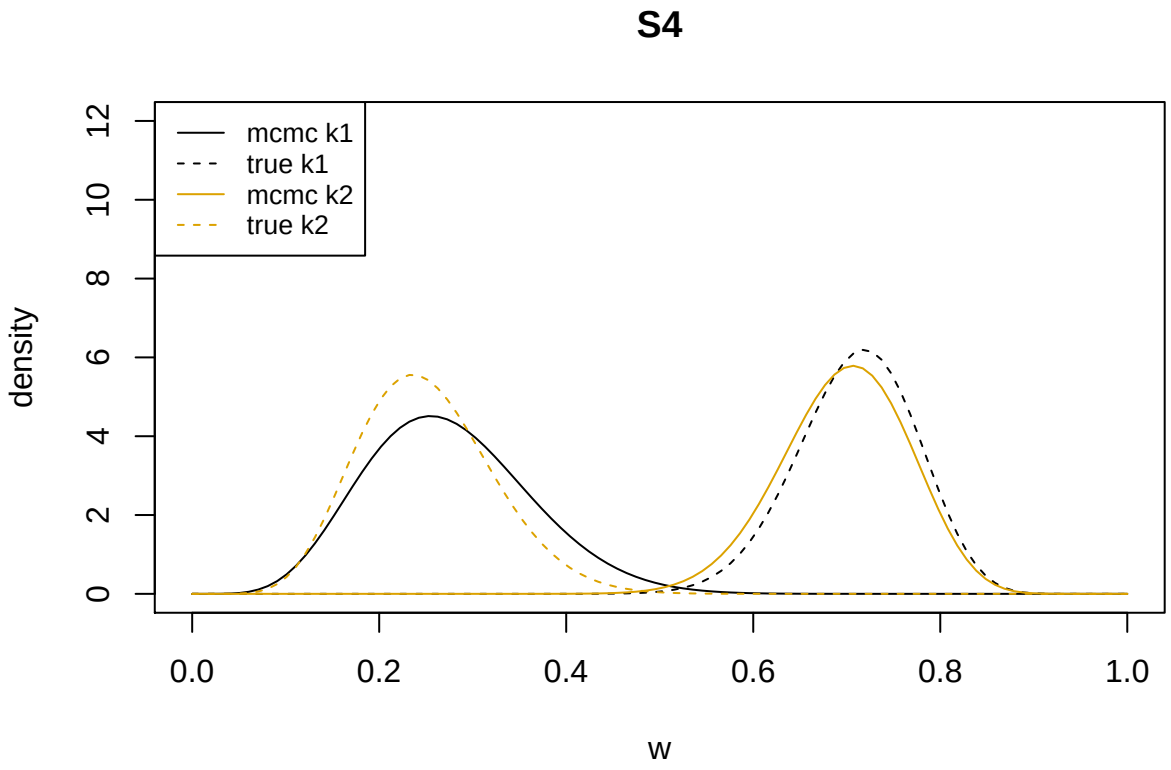

```
mcmc_z <- as.vector(mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "z", "Mean"])
#mcmc_z <- round(mcmc_z, 0)
plot(z, mcmc_z, type = "p")
```



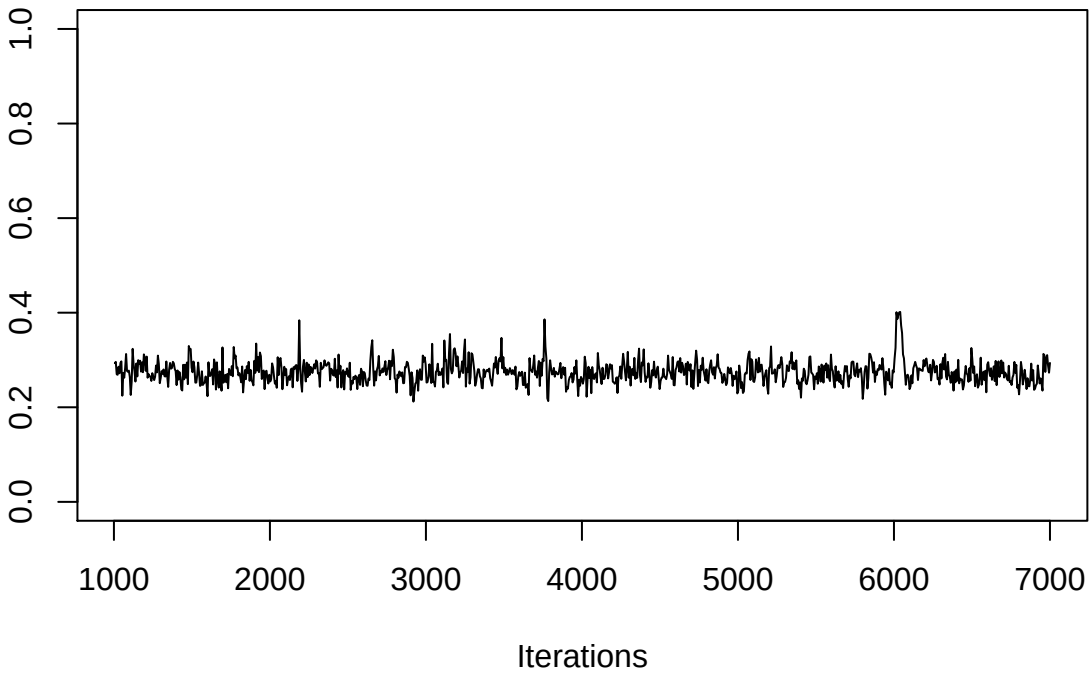

```
mcmc_U <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "U", "Mean"]
mcmc_U <- matrix(mcmc_U, nrow=K)
mcmc_V <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "V", "Mean"]
mcmc_V <- matrix(mcmc_V, nrow=K)

mcmc_U
```

```
##           [,1]
## [1,] 0.2744832
## [2,] 0.6973845
```

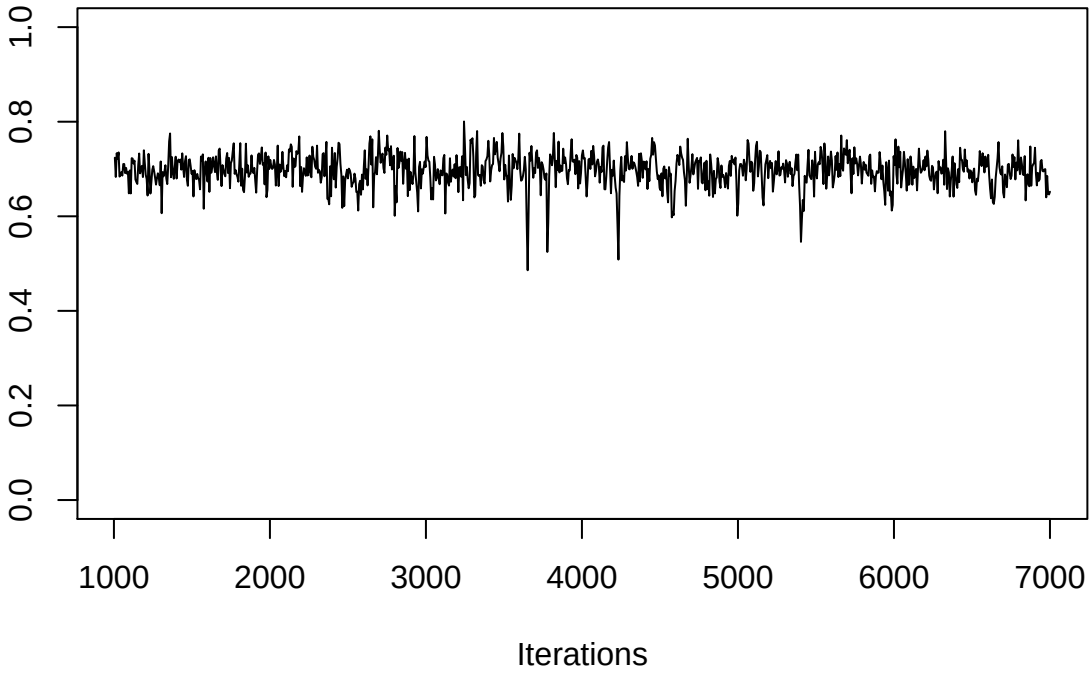
```r
p <- seq(0, 1, length = 100)
colors <- c("#000000", "#DCA200", "#8FA7ED", "#9D847A", "#A47901")
for (k in 1:K) {
  if (k == 1) {
    # plot mcmc mean U,V
    plot(p, dbeta(p, mcmc_U[k] * mcmc_V[k], (1-mcmc_U[k])*mcmc_V[k]),
         main = "S4",
         ylab = "density", xlab = "w", type = "l", col = colors[k],
         ylim = c(0, 12))
    # plot truth
    lines(p, dbeta(p, a.ordered[k,4], b.ordered[k,4]), type = "l", col = colors[k], lty=2)
    # add legend
    legend(x = "topleft",
           legend = paste0(c("mcmc k", "true k"), rep(1:K, each=2)),
           col = colors[rep(1:K, each=2)],
           lty = rep(1:2, K),
           cex=0.8)
  } else {
    # plot mcmc mean U,V
    lines(p, dbeta(p, mcmc_U[k] * mcmc_V[k], (1-mcmc_U[k])*mcmc_V[k]),
          type = "l", col = colors[k])
    # plot truth
    lines(p, dbeta(p, a.ordered[k,4], b.ordered[k,4]), type = "l", col = colors[k], lty=2)
  }
}
```



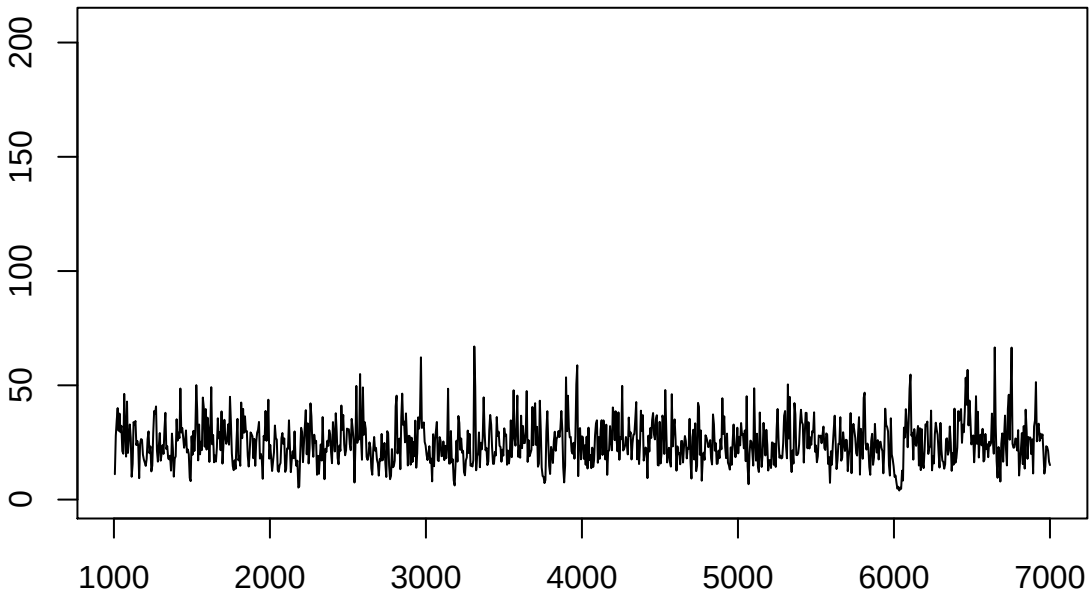
```r
U_trace <- samps[[1]][, 1:2]
V_trace <- samps[[1]][, 3:4]
traceplot(U_trace[,1], ylim = c(0,1))
```



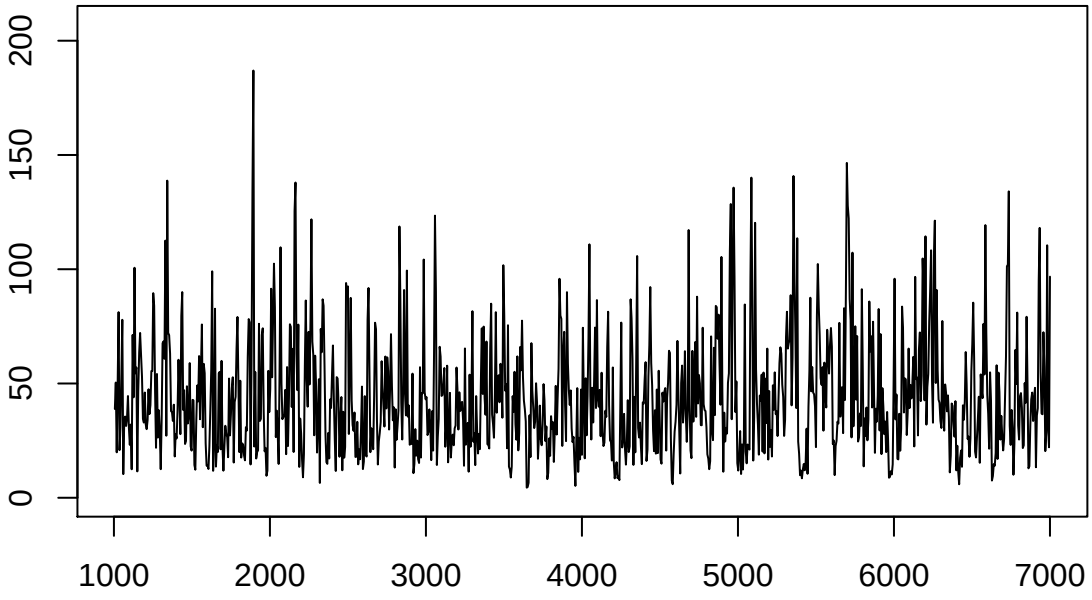

```r
traceplot(U_trace[,2], ylim = c(0,1))
```



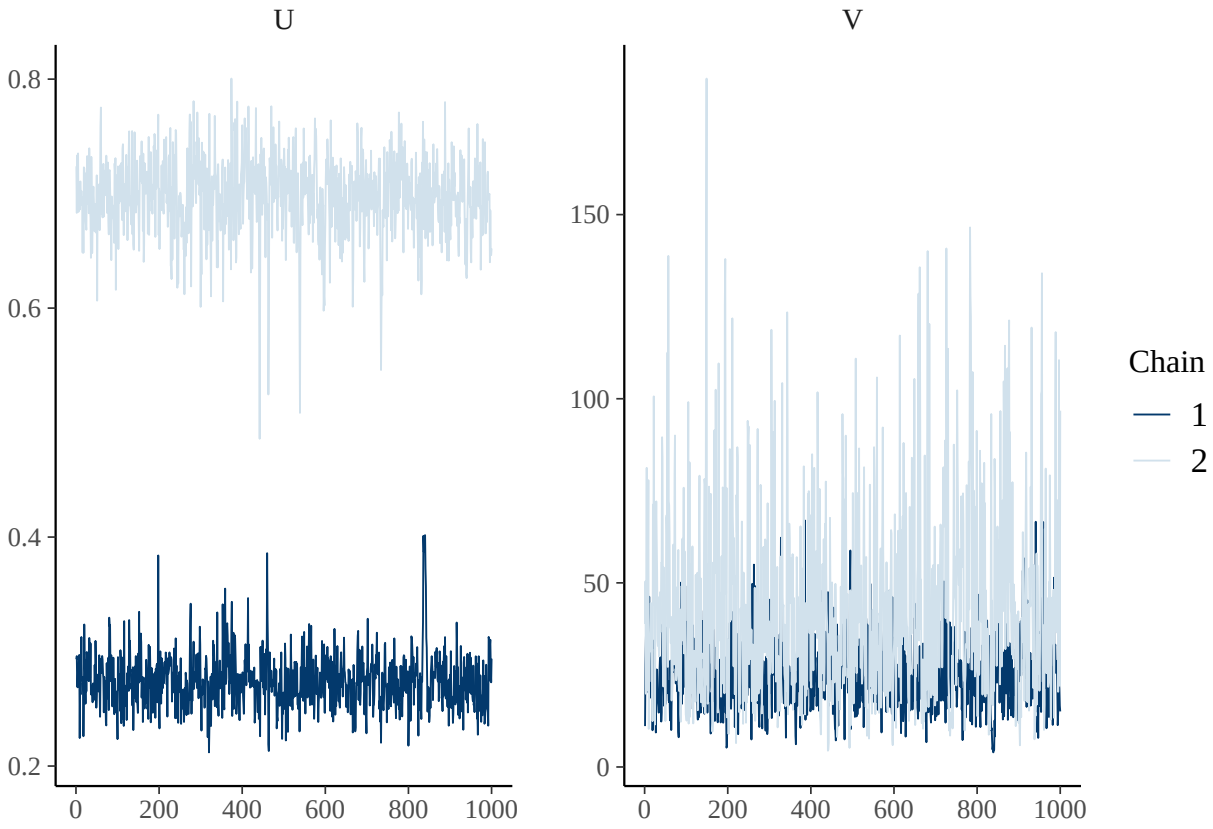

```r
traceplot(V_trace[,1], ylim = c(0,max(V_trace)+20))
```




```
traceplot(V_trace[,2], ylim = c(0,max(V_trace)+20))
```




```
x1 <- samps[[1]][, c(1,3)]
dimnames(x1)[[2]] <- c("U", "V")
x2 <- samps[[1]][, c(2,4)]
dimnames(x2)[[2]] <- c("U", "V")
UV_trace <- list(x1, x2)
mcmc_trace(UV_trace, regex_pars = c("U", "V"))
```


U
V
0.8
0.6
0.4
0.2
150
100
50
0
0
200
400
600
800
1000
Chain
1
2