Version 2: 2 clusters; Fix cluster label switching u,v; unif; fix w [0,1]; only force ordering of means in 1 sample; S2

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_S2 <- n[ , 2]
y_S2 <- y[ , 2]
m_S2 <- m[ , 2]
tcn_S2 <- tcn[ , 2]
w_S2 <- w[ ,2]
S <- 1
```

JAGS

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

test.data <- list("I" = I, "K" = K,
                  "y" = y_S2, "n" = n_S2,
                  "m" = m_S2, "tcn" = tcn_S2)
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)
```

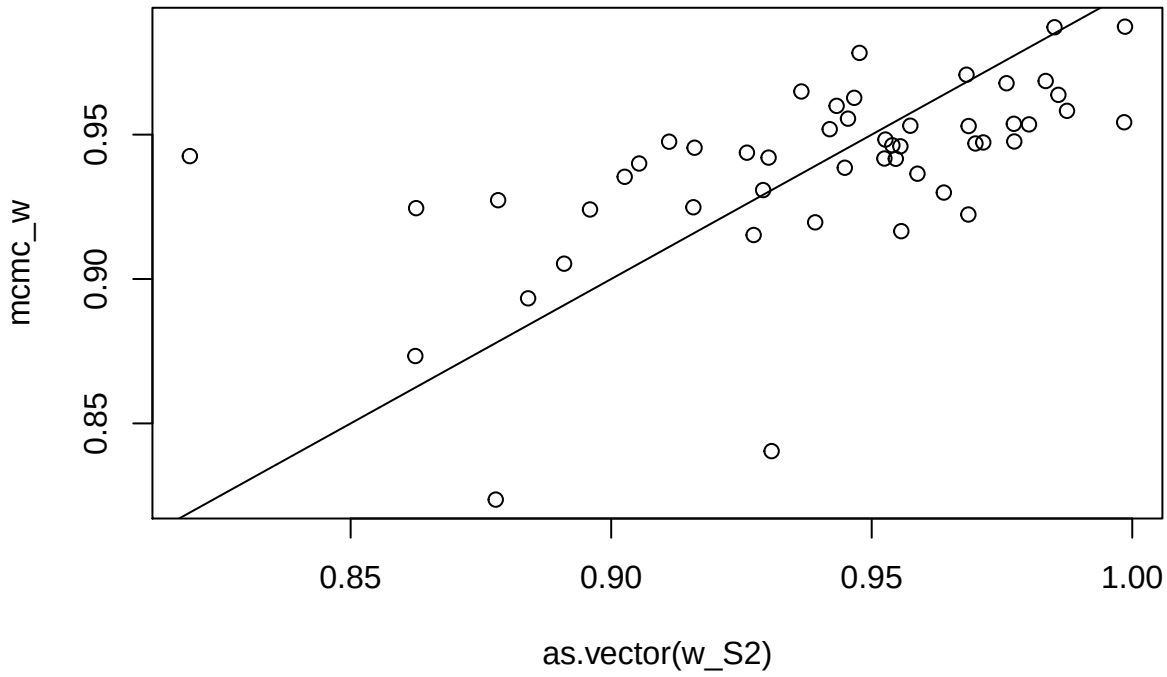
```
##        U[1]        U[2]        V[1]        V[2]        w[1]        w[2]
##    3.975986   75.751368  164.883082  287.334683 1000.000000  754.202444
##        w[3]        w[4]        w[5]        w[6]        w[7]        w[8]
```

```
##  858.001393  654.634937  864.694545 1000.000000  846.950086 1000.000000
##        w[9]       w[10]       w[11]       w[12]       w[13]       w[14]
##  866.876913 1000.000000  758.488840  378.708191  802.212175 1000.000000
##       w[15]       w[16]       w[17]       w[18]       w[19]       w[20]
##  856.535379 1000.000000 1000.000000 1000.000000  849.721823  870.682276
##       w[21]       w[22]       w[23]       w[24]       w[25]       w[26]
##  910.608545  844.556110 1000.000000  803.964893 1000.000000 1000.000000
##       w[27]       w[28]       w[29]       w[30]       w[31]       w[32]
## 1000.000000 1000.000000  922.000428 1000.000000  785.650537 1106.095567
##       w[33]       w[34]       w[35]       w[36]       w[37]       w[38]
## 1000.000000 1000.000000  525.640372 1000.000000  894.857599  906.586993
##       w[39]       w[40]       w[41]       w[42]       w[43]       w[44]
##  723.646398  904.893604  810.599906  871.005674 1000.000000 1000.000000
##       w[45]       w[46]       w[47]       w[48]       w[49]       w[50]
##  643.529206 1000.000000 1000.000000  904.149414  804.524368  187.479915
##        z[1]        z[2]        z[3]        z[4]        z[5]        z[6]
##   61.438593   82.095702   41.207265   62.636671   85.044144   35.625770
##        z[7]        z[8]        z[9]       z[10]       z[11]       z[12]
##   36.506402   78.793929   41.596140   56.643939   44.711275  106.408566
##       z[13]       z[14]       z[15]       z[16]       z[17]       z[18]
##   77.352278   53.030275  115.528026   29.146340   68.305897   28.283122
##       z[19]       z[20]       z[21]       z[22]       z[23]       z[24]
##   63.323487   82.279672   51.046873   28.722250   39.533794   41.423986
##       z[25]       z[26]       z[27]       z[28]       z[29]       z[30]
##   32.556134   46.978255   73.674053   66.677082   36.277009   41.397802
##       z[31]       z[32]       z[33]       z[34]       z[35]       z[36]
##   63.581079   73.298563   65.614865   68.066840   48.651516   68.866810
##       z[37]       z[38]       z[39]       z[40]       z[41]       z[42]
##   39.223534   77.605132   47.634927   50.600416   90.421165   41.176262
##       z[43]       z[44]       z[45]       z[46]       z[47]       z[48]
##   57.674543   66.020137  170.328688   75.156088   77.065664   64.880520
##       z[49]       z[50]
##   28.158238  172.161250
```

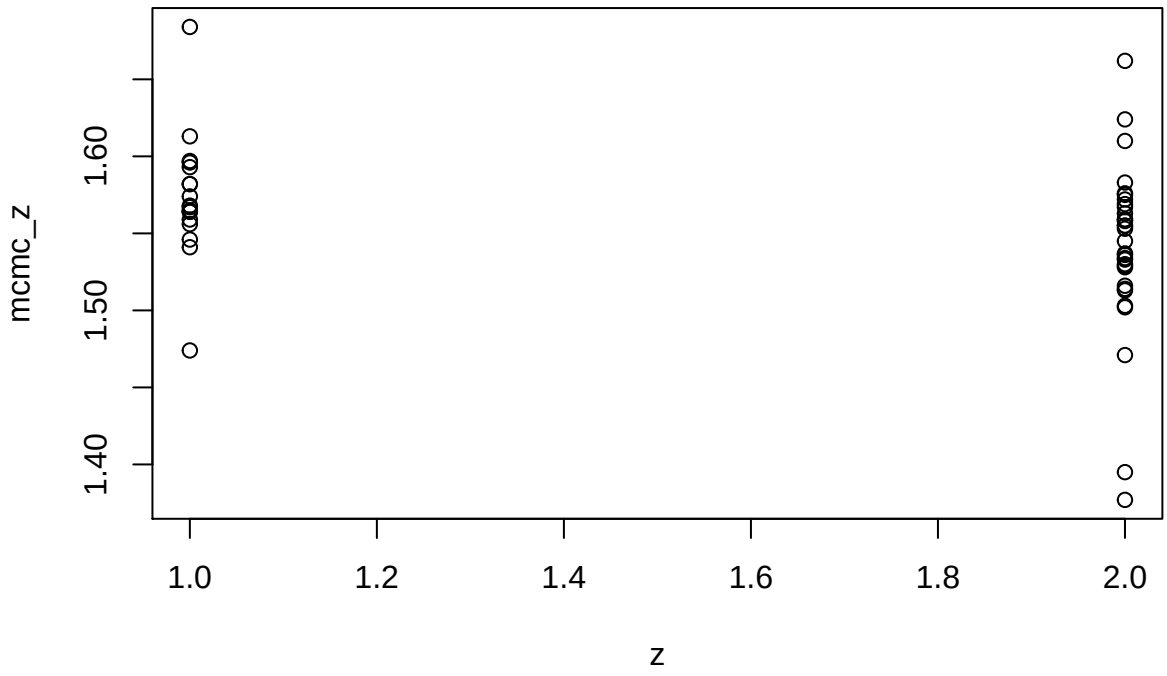
```
pdf(file.path(trace.dir, paste0(runName, "_trace.pdf")))
plot(samps)
dev.off()
```

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

```
mcmc_vals <- s$statistics
mcmc_w <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "w", "Mean"]
plot(as.vector(w_S2), mcmc_w, type = "p")
abline(a=0, b=1)
```



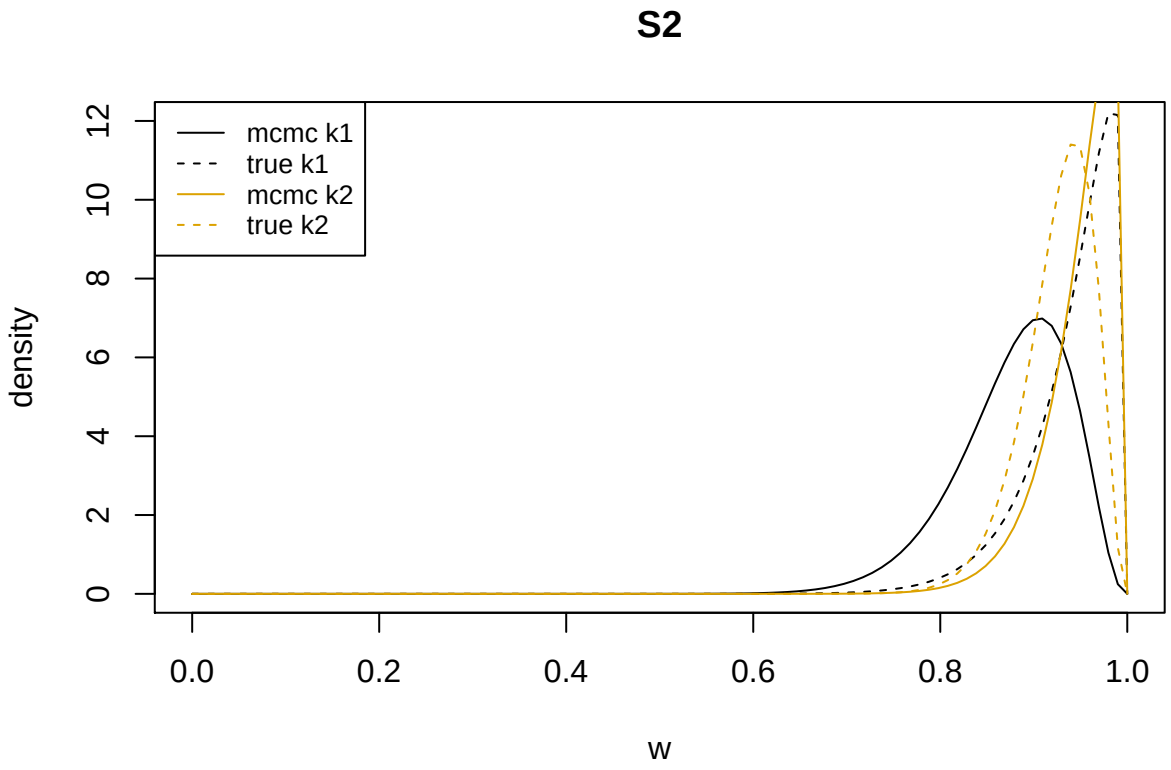

```
mcmc_z <- as.vector(mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "z", "Mean"])
#mcmc_z <- round(mcmc_z, 0)
plot(z, mcmc_z, type = "p")
```



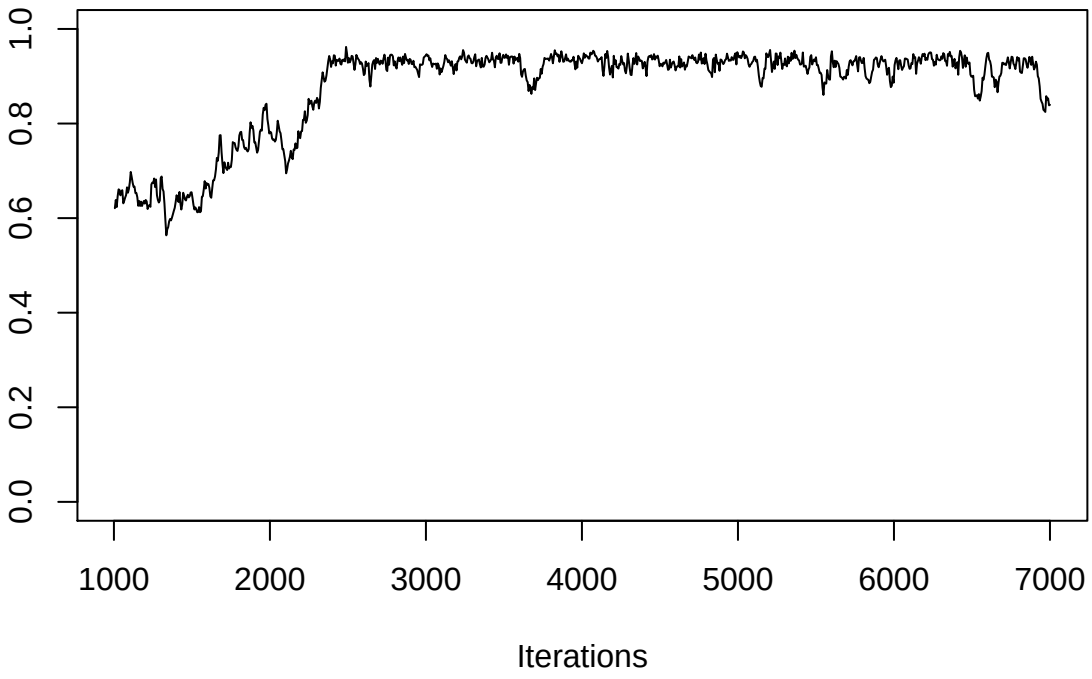

```
mcmc_U <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "U", "Mean"]
mcmc_U <- matrix(mcmc_U, nrow=K)
mcmc_V <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "V", "Mean"]
mcmc_V <- matrix(mcmc_V, nrow=K)

mcmc_U
```

```
##           [,1]
## [1,] 0.8771262
## [2,] 0.9504203
```

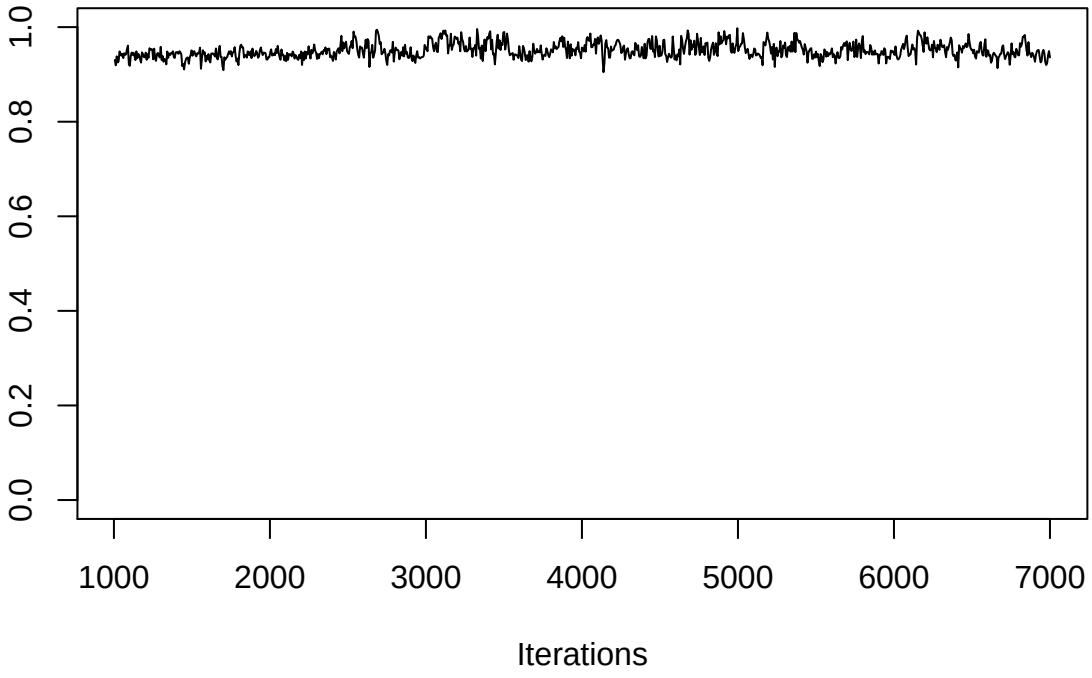
```r
p <- seq(0, 1, length = 100)
colors <- c("#000000", "#DCA200", "#8FA7ED", "#9D847A", "#A47901")
for (k in 1:K) {
  if (k == 1) {
    # plot mcmc mean U,V
    plot(p, dbeta(p, mcmc_U[k] * mcmc_V[k], (1-mcmc_U[k])*mcmc_V[k]),
         main = "S2",
         ylab = "density", xlab = "w", type = "l", col = colors[k],
         ylim = c(0, 12))
    # plot truth
    lines(p, dbeta(p, a.ordered[k,2], b.ordered[k,2]), 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,2], b.ordered[k,2]), type = "l", col = colors[k], lty=2)
  }
}
```



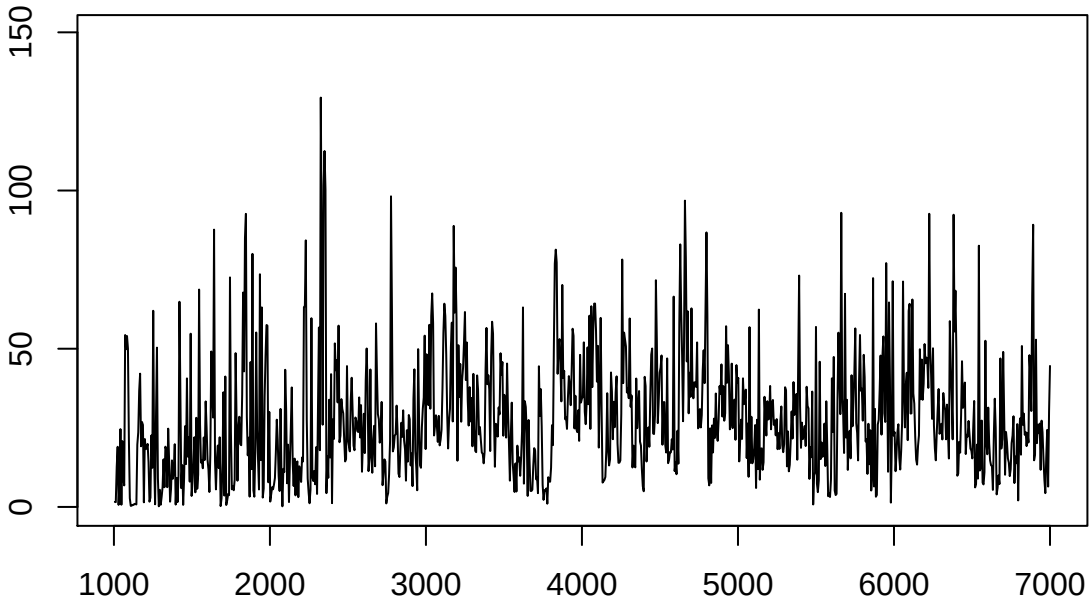
```r
U_trace <- samps[[1]][, 1:2]
V_trace <- samps[[1]][, 3:4]
traceplot(U_trace[,1], ylim = c(0,1))
```



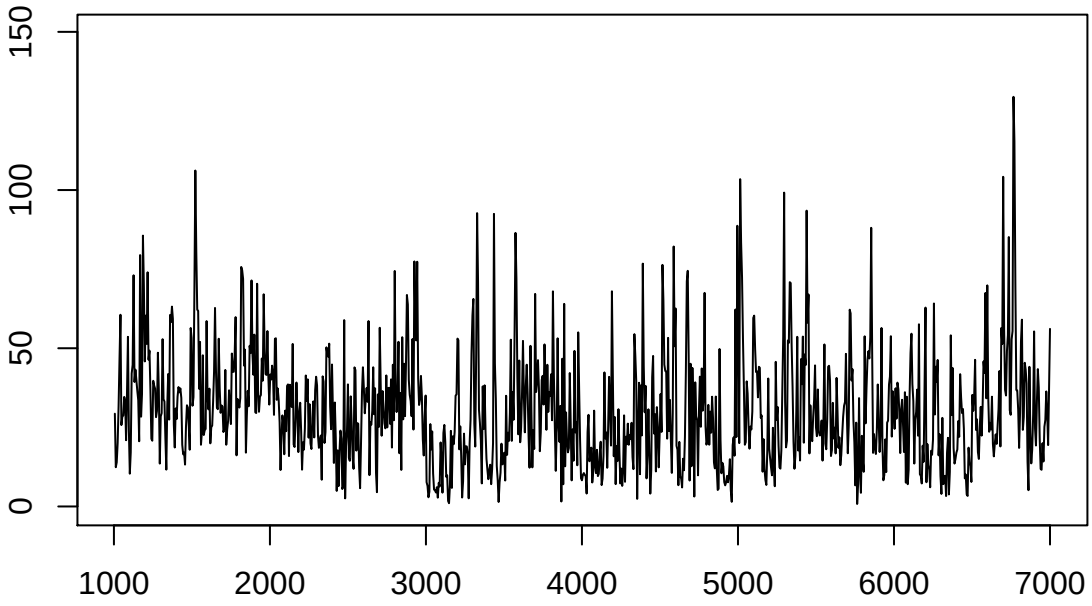

```r
traceplot(U_trace[,2], ylim = c(0,1))
```



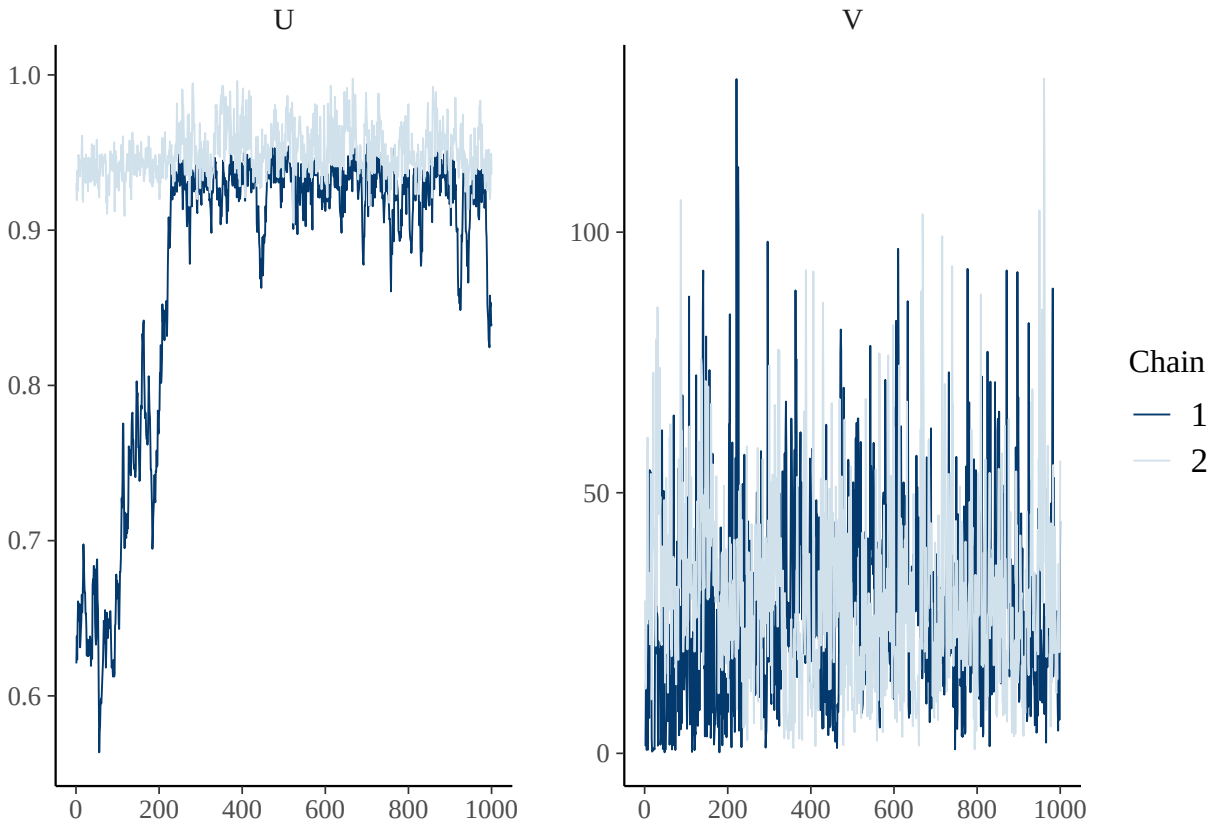

```r
traceplot(V_trace[,1], ylim = c(0,max(V_trace)+20))
```




```
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
1.0
0.9
0.8
0.7
0.6
100
50
0
0
200
400
600
800
1000
Chain
1
2