Version 3: 3 clusters

Simulate data

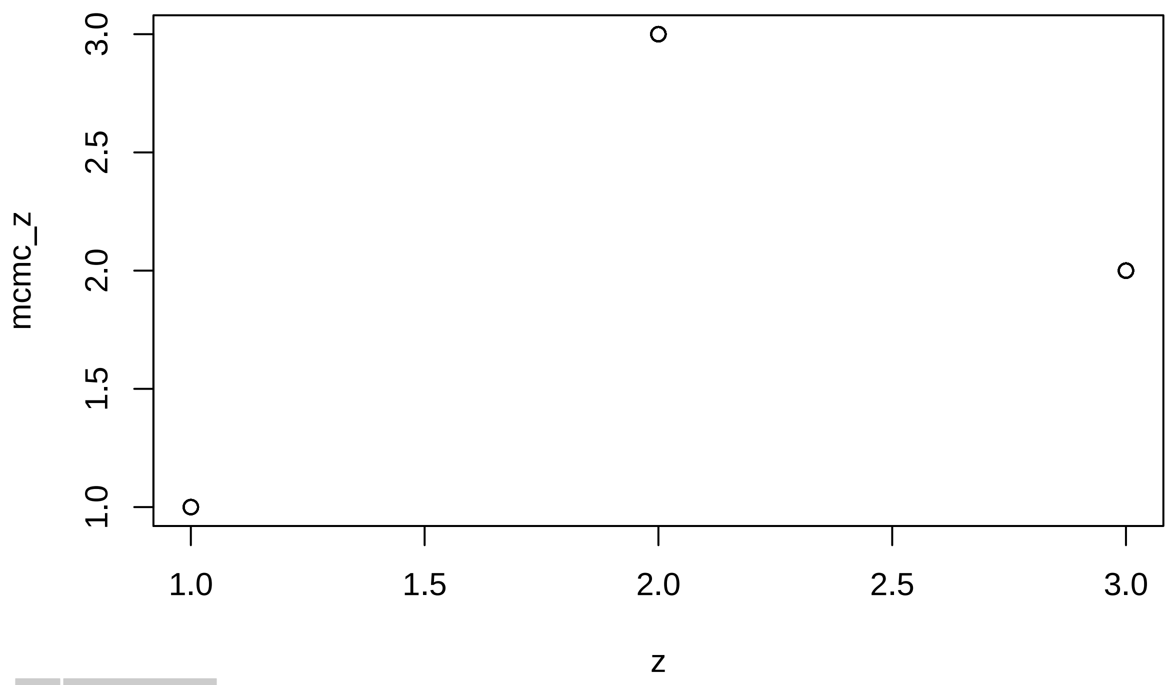
```r
I <- 50
K <- 3
S <- 10

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

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)
}
colnames(a) <- colnames(b) <- paste0("sample", seq_len(ncol(a)))
# reorder a,b matrices to match ordering of means (U) in S1
U <- a/(a+b)
V <- a+b
ix <- order(U[, 1])
U.ordered <- U[ix, ]
a.ordered <- a[ix, ]
b.ordered <- b[ix, ]
V.ordered <- V[ix, ]

#pi <- as.vector(rdirichlet(1, rep(1, K)))
pi <- c(0.2, 0.3, 0.5)
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])
  }
}

test.data <- list("I" = I, "S" = S, "K" = K,
                  "y" = y, "n" = n,
                  "m" = m, "tcn" = tcn)
```

JAGS

```
jags.file <- file.path(models.dir, "v3_no_constraints.jags")
inits <- list(".RNG.name" = "base::Wichmann-Hill",
              ".RNG.seed" = 123)
jags.m <- jags.model(jags.file, test.data,
                     n.chains = 1,
                     inits = inits,
                     n.adapt = 1000)
```

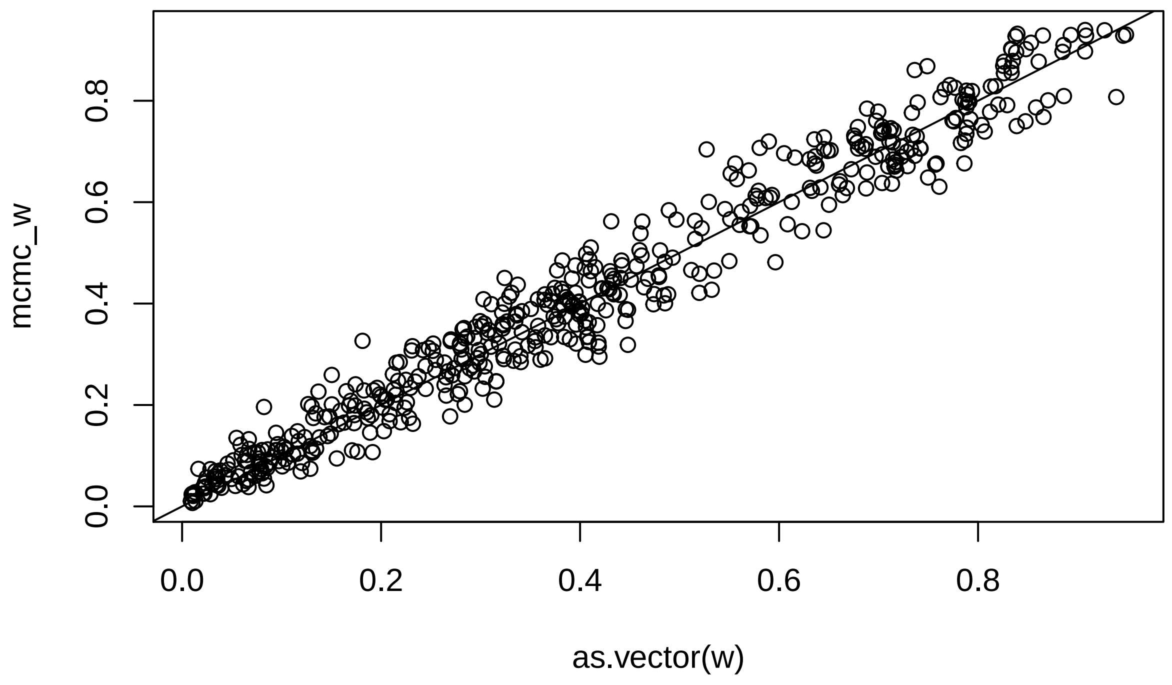
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 500
##    Unobserved stochastic nodes: 611
##    Total graph size: 8531
##
## Initializing model
```

```
params <- c("z", "w", "U", "V")
samps <- coda.samples(jags.m, params, n.iter=10000, thin=7)
jags_df <- ggs(samps)
s <- summary(samps)
#effectiveSize(samps)
#pdf(file.path(trace.dir, paste0(runName, "_trace.pdf")))
#plot(samps)
#dev.off()
```



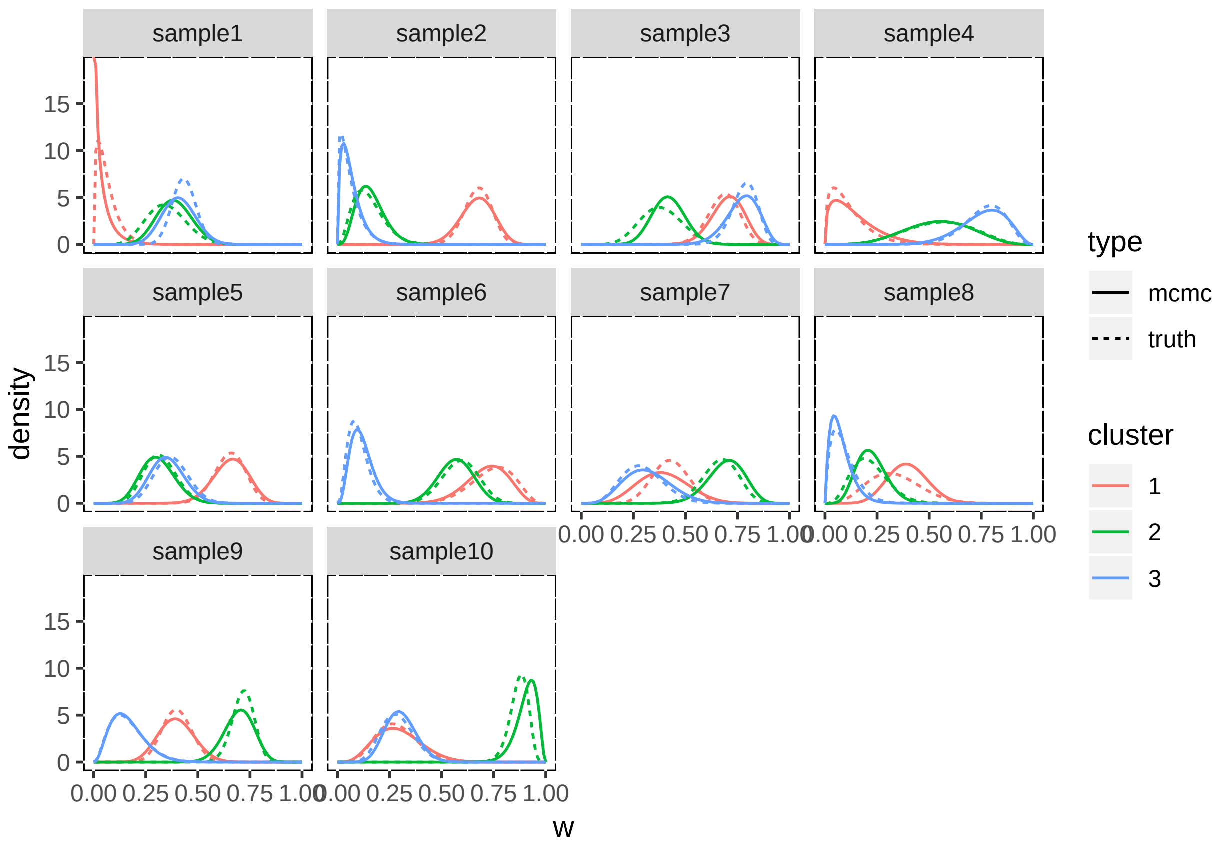
z	mcmc_z
1	1
2	3
3	2

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



```r
# 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)

# p <- seq(0, 1, length = 100)
# colors <- c("#000000", "#DCA200", "#8FA7ED", "#9D847A", "#A47901")
# for (s in 1:S) {
#   for (k in 1:K) {
#     if (k == 1) {
#       # plot mcmc mean U,V
#       plot(p, dbeta(p, mcmc_U[k,s] * mcmc_V[k,s], (1-mcmc_U[k,s])*mcmc_V[k,s]),
#            main = paste0("S", s),
#            ylab = "density", xlab = "w", type = "l", col = colors[k],
#            ylim = c(0, 20))
#       # plot truth
#       lines(p, dbeta(p, a.ordered[k,s], b.ordered[k,s]), type = "l", col = colors[k], lty=2)
#       # add legend
#       allU <- round(as.vector(rbind(mcmc_U[,s], U.ordered[,s])), digits = 2)
#       legend(x = "topleft",
#              legend = paste0(c("mean k", "true k"), rep(1:K, each=2), ", U=", allU),
#              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,s] * mcmc_V[k,s], (1-mcmc_U[k,s])*mcmc_V[k,s]),
#             type = "l", col = colors[k])
#       # plot truth
#       lines(p, dbeta(p, a.ordered[k,s], b.ordered[k,s]), type = "l", col = colors[k], lty=2)
#     }
#   }
# }
```

```r
mcmc_U <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "U", "Mean"]
mcmc_U <- matrix(mcmc_U, nrow=K)
colnames(mcmc_U) <- paste0("sample", 1:S)
mcmc_V <- mcmc_vals[substr(rownames(mcmc_vals), 1, 1) == "V", "Mean"]
mcmc_V <- matrix(mcmc_V, nrow=K)
colnames(mcmc_V) <- paste0("sample", 1:S)

mcmc_U2 <- mcmc_U %>%
    as_tibble() %>%
    mutate(cluster=z_mapping$mcmc_z,
           type="mcmc") %>%
    gather("sample", "U", -c(cluster, type)) %>%
    mutate(sample=factor(sample, levels=colnames(mcmc_U)))

mcmc_betas <- mcmc_V %>%
    as_tibble() %>%
    mutate(cluster=z_mapping$mcmc_z,
           type="mcmc") %>%
    gather("sample", "V", -c(cluster, type)) %>%
    mutate(sample=factor(sample, levels=colnames(mcmc_U))) %>%
    left_join(mcmc_U2, by=c("cluster", "type", "sample"))
mcmc_betas$a <- mcmc_betas$U * mcmc_betas$V
mcmc_betas$b <- (1 - mcmc_betas$U) * mcmc_betas$V

a2 <- a.ordered %>%
    as_tibble() %>%
    mutate(cluster=1:K,
           type="truth") %>%
    gather("sample", "a", -c(cluster, type)) %>%
    mutate(sample=factor(sample, levels=colnames(a.ordered)))

true_betas <- b.ordered %>%
    as_tibble() %>%
    mutate(cluster=1:K,
           type="truth") %>%
    gather("sample", "b", -c(cluster, type)) %>%
    mutate(sample=factor(sample, levels=colnames(b.ordered))) %>%
    left_join(a2, by=c("cluster", "type", "sample"))

betas <- bind_rows(true_betas, mcmc_betas)

dens_list <- list()
for(row.ix in 1:nrow(betas)) {
  x <- seq(0, 1, length = 100)
  val <- dbeta(x, shape1=betas[row.ix, ]$a, shape2=betas[row.ix, ]$b)
  dens_list[[row.ix]] <- data.frame(x=x, val=val,
                                    type=betas[row.ix, ]$type,
                                    cluster=betas[row.ix, ]$cluster,
                                    sample=betas[row.ix, ]$sample,
                                    stringsAsFactors = F)
}

dens_df <- bind_rows(dens_list)
dens_df$grp <- paste0("k", dens_df$cluster, ",", dens_df$sample, ",", dens_df$type)
dens_df$cluster <- as.factor(dens_df$cluster)

## beta distributions by sample
beta.plot <- ggplot(dens_df, aes(x=x, y=val, group=grp)) +
  geom_line(aes(color=cluster, lty=type)) +
  facet_wrap(~sample) +
  ylab("density") +
  xlab("w") +
  theme(panel.background=element_rect(fill="white", color="black"))
beta.plot
```



```
samples_matrix <- as.matrix(samps)

U_chains <- samples_matrix[, which(substr(colnames(samples_matrix), 1, 1) == "U")]

pdf(file.path(working.dir, paste0(runName, "_Uplots.pdf")))
for (s in 1:S) {
  s.char <- paste(s)
  if (nchar(s.char) == 1) {
    U_sub <- U_chains[, which(substr(colnames(U_chains), 5, 6) == paste0(s.char, "]"))]
  } else {
    U_sub <- U_chains[, which(substr(colnames(U_chains), 5, 7) == paste0(s.char, "]"))]
  }
  plot(density(as.vector(U_sub)),
       main = paste0("S", s, " U density; K=", K))
  legend(x = "topright",
         legend = paste0("k", 1:K, ", true U = ", round(U[,s], 2)))
}
dev.off()

pdf(file.path(working.dir, paste0(runName, "_Uplots_withTruth.pdf")))
for (s in 1:S) {
  s.char <- paste(s)
  if (nchar(s.char) == 1) {
    U_sub <- U_chains[, which(substr(colnames(U_chains), 5, 6) == paste0(s.char, "]"))]
  } else {
    U_sub <- U_chains[, which(substr(colnames(U_chains), 5, 7) == paste0(s.char, "]"))]
  }
```

```r
  plot(density(as.vector(U_sub)),
       main = paste0("S", s, " U density; K=", K))
  legend(x = "topright",
         legend = paste0("k", 1:K, ", true U = ", round(U[,s], 2)),
         col = colors[1:K],
         lty = "dashed")
  u <- round(U[,s], 2)
  for ( i in 1:length(u)) {
    abline(v = u[i], col = colors[i], lty = "dashed")
  }

}
dev.off()
```