Cu-MSDSp Example

```
library(rjson)
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.26.13 (Stan version 2.26.1)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

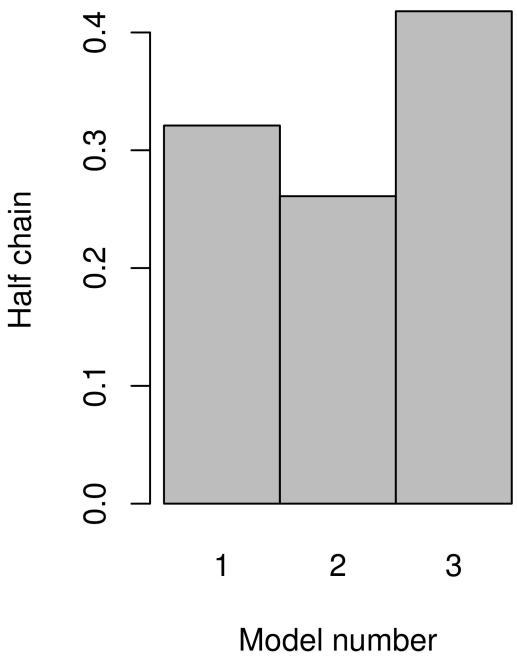
```
library(rstantools)
```

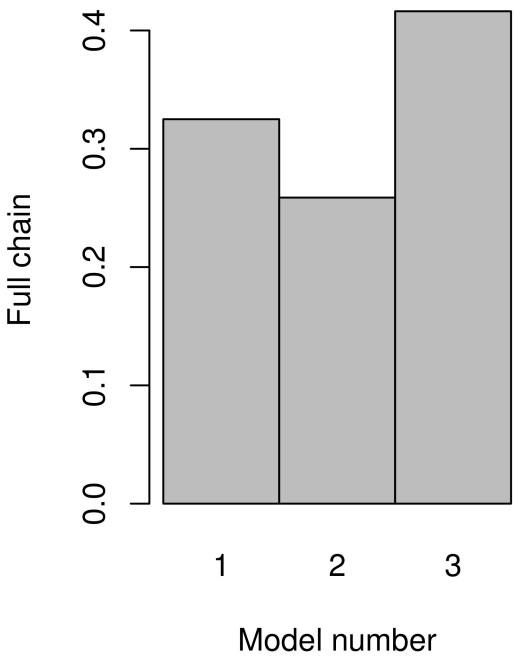
```
## This is rstantools version 2.2.0
```

```
####functions block
###  https://stackoverflow.com/questions/10296866/
###  finding-the-column-number-and-value-the-of-second-highest-value-in-a-row
maxn <- function(n) function(x) order(x, decreasing = TRUE)[n]
max1 <- maxn(1)

#function for calculating median squared error for the estimated centers
center_mse <- function(centers,estimates){
  X <- list()
  for (i in 1:ncol(centers)) {
    X[[i]] <- apply(estimates,2,function(x) mean((x-centers[,i])^2) )
  }
  return(X)
}
```

Plotting the model posterior distributions

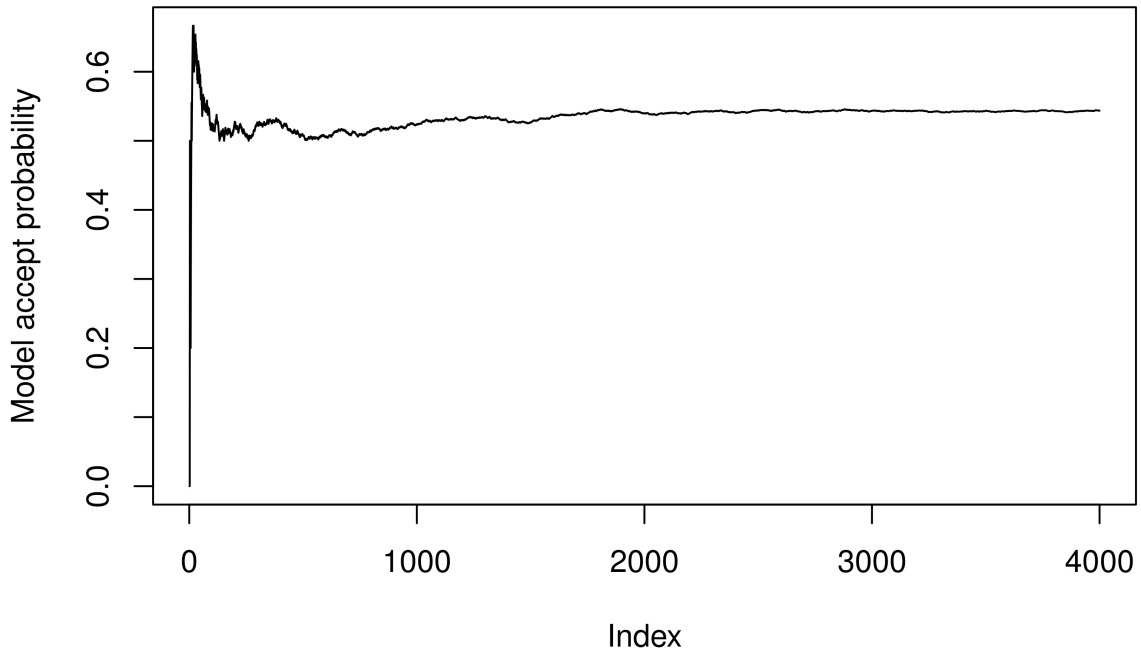
```
modelIndexChain1 <- read.csv("./src_example/modelSelection/modelIndexChain1.csv",header = F)
par(mfrow=c(1,2))
barplot(table(modelIndexChain1$V1[2001:4000])/(length(modelIndexChain1$V1[2001:4000])),
        space = 0,main = "",ylab = "Half chain", names.arg = c(1,2,3),xlab = "Model number")
barplot(table(modelIndexChain1$V1)/(length(modelIndexChain1$V1)),space = 0,main = "",
        ylab = "Full chain", names.arg = c(1,2,3),xlab = "Model number")
```





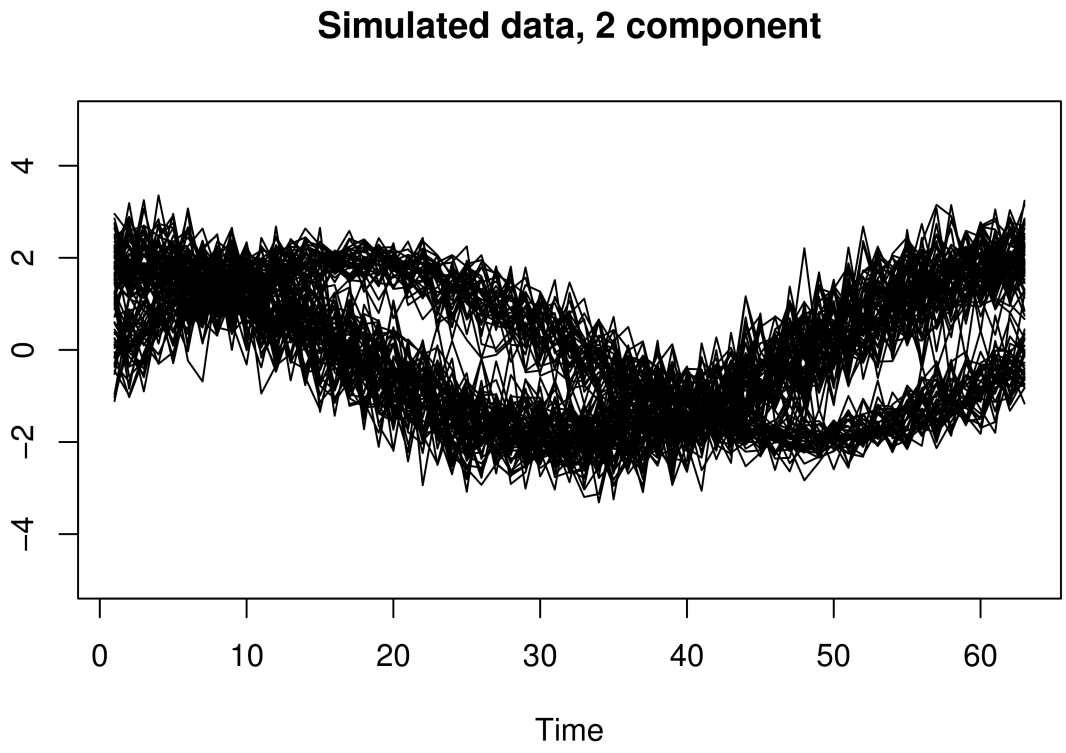
You can check if your MCMC chain converged by checking the modelAcceptChain

```
modelAcceptChain1 <- read.csv("./src_example/modelSelection/AcceptChain1.csv",header = F)
plot(modelAcceptChain1[[1]],type="l",ylab="Model accept probability",main="")
```



Loading the data

```
data <- as.list(fromJSON(file="./src_example/data/data.json"))
series <- t(matrix(unlist(data$z),ncol = 63, nrow = 100))
#png(file ="./figures/figure2-0.png",width = 480 )
ts.plot(series,ylim=c(-5,5),main="Simulated data, 2 component")
#dev.off()
```



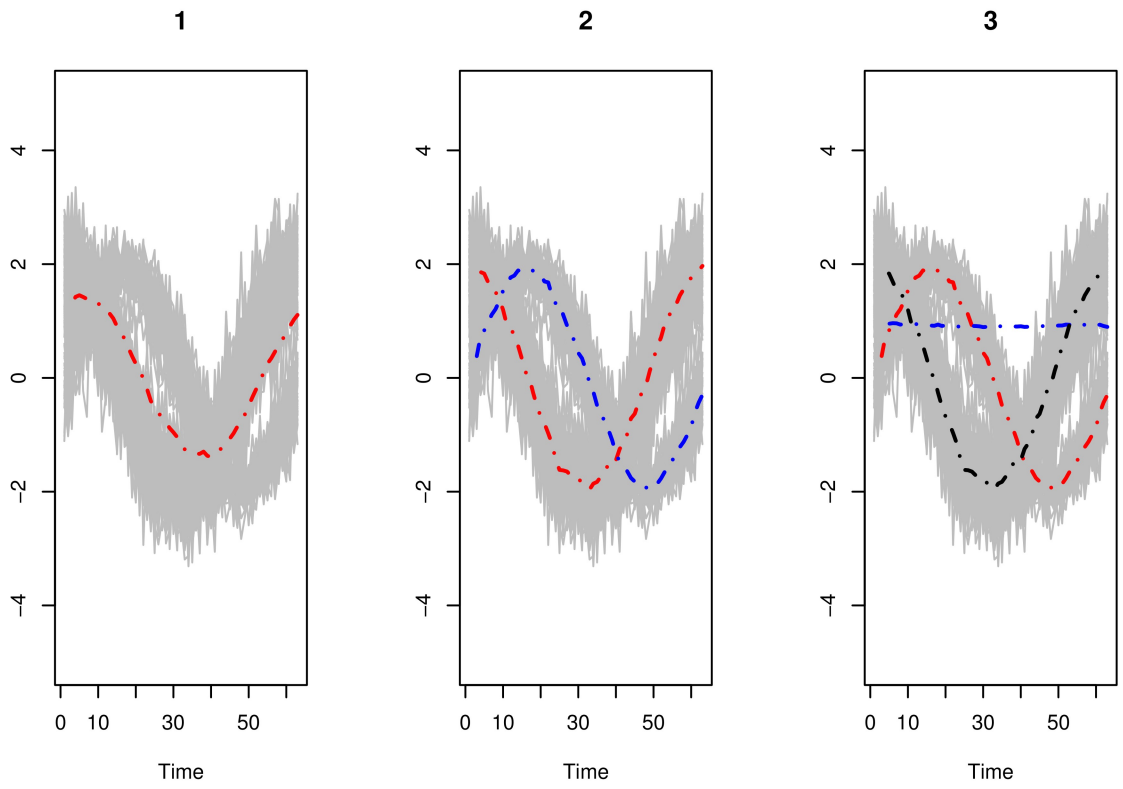
Calculating the estimates for the parameters

```
data1 <- read.csv("./src_example/goldStandardChains/1/1GSC1.csv")
data2 <- read.csv("./src_example/goldStandardChains/2/2GSC1.csv")
data3 <- read.csv("./src_example/goldStandardChains/3/3GSC1.csv")

fit1_means <- apply(data1,2,median)
fit2_means <- apply(data2,2,median)
fit3_means <- apply(data3,2,median)

l1_1 <- paste('l.',1:63,sep='')
l1 <- paste('l.1.',1:63,sep='')
l2 <- paste('l.2.',1:63,sep='')
l3 <- paste('l.3.',1:63,sep='')
```

Plotting the estimated mixture component centers

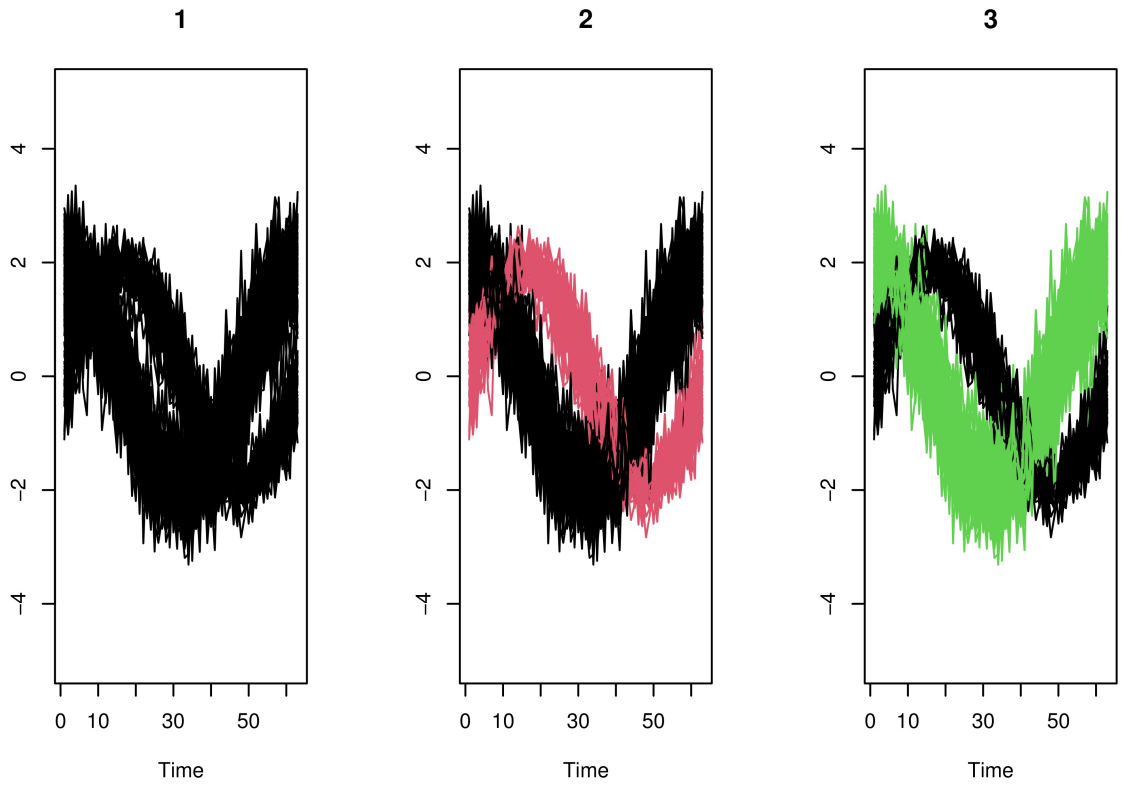
```
par(mfrow=c(1,3))

#1
L1_1 <-fit1_means[l1_1]

ts.plot(series, col="gray",ylim=c(-5,5),main="1")
```

```
lines(1:63,L1_1,"l",col="red", lwd=2 ,lty=4)

#2
L2_1 <- fit2_means[l1]
L2_2 <- fit2_means[l2]

ts.plot(series, col="gray",ylim=c(-5,5),main="2")
lines(1:63,L2_1,"l",col="red", lwd=2 ,lty=4)
lines(1:63,L2_2,"l",col="blue", lwd=2 ,lty=4)

#3
L3_1 <- fit3_means[l1]
L3_2 <- fit3_means[l2]
L3_3 <- fit3_means[l3]

ts.plot(series, col="gray",ylim=c(-5,5),main="3")
lines(1:63,L3_1,"l",col="red", lwd=2 ,lty=4)
lines(1:63,L3_2,"l",col="blue", lwd=2 ,lty=4)
lines(1:63,L3_3,"l",col="black", lwd=2 ,lty=4)
```



Assignment and plotting the clusters

```
par(mfrow=c(1,3))
ts.plot(series,main="1",ylim = c(-5,5))

dist_to_L2_1 <- c()
for (i in 1:ncol(series)) {
  dist_to_L2_1[i] <- fit2_means["theta.1"]/sqrt(fit2_means["epsilon.1"])*
                     exp(-0.5*(sum((L2_1 - series[,i])^2))/fit2_means["epsilon.1"])
}
dist_to_L2_2 <- c()
for (i in 1:ncol(series)) {
  dist_to_L2_2[i] <- fit2_means["theta.2"]/sqrt(fit2_means["epsilon.2"])*
                     exp(-0.5*(sum((L2_2 - series[,i])^2))/fit2_means["epsilon.2"])
}

Ldist2 <- data.frame(dist_to_L2_1,dist_to_L2_2)
assignments2 <- c(apply(Ldist2,1,max1))

plot(1:nrow(series),series[,1],type = "l",col=assignments2[1],xlab = "Time",
     ylab = "",ylim = c(-5,5),main="2")
for (i in 1:ncol(series)) {
  lines(1:nrow(series),series[,i],type = "l",col=assignments2[i])
}

dist_to_L3_1 <- c()
for (i in 1:ncol(series)) {
  dist_to_L3_1[i] <- fit3_means["theta.1"]/sqrt(fit3_means["epsilon.1"])*
                     exp(-0.5*(sum((L3_1 - series[,i])^2))/fit3_means["epsilon.1"])
}
dist_to_L3_2 <- c()
for (i in 1:ncol(series)) {
  dist_to_L3_2[i] <- fit3_means["theta.2"]/sqrt(fit3_means["epsilon.2"])*
                     exp(-0.5*(sum((L3_2 - series[,i])^2))/fit3_means["epsilon.2"])
}
dist_to_L3_3 <- c()
for (i in 1:ncol(series)) {
  dist_to_L3_3[i] <- fit3_means["theta.3"]/sqrt(fit3_means["epsilon.3"])*
                     exp(-0.5*(sum((L3_3 - series[,i])^2))/fit3_means["epsilon.3"])
}

Ldist3 <- data.frame(dist_to_L3_1,dist_to_L3_2,dist_to_L3_3)
assignments3 <- c(apply(Ldist3,1,max1))

plot(1:nrow(series),series[,1],type = "l",col=assignments3[1],xlab = "Time",
     ylab = "",ylim = c(-5,5),main="3")
for (i in 1:ncol(series)) {
  lines(1:nrow(series),series[,i],type = "l",col=assignments3[i])
}
```



Mixture weight estimates (The true values in the simulation were 0.67 and 0.33, but the realized are 0.63 and 0.37)

Realised and estimated values

```
round(table(assignments2)/100,5);round(fit2_means[c("theta.1","theta.2")],5)
```

```
## assignments2
##    1    2
## 0.63 0.37
```

```
## theta.1 theta.2
## 0.62816 0.37184
```

```
round(table(assignments3)/100,5);round(fit3_means[c("theta.1","theta.2","theta.3")],5)
```

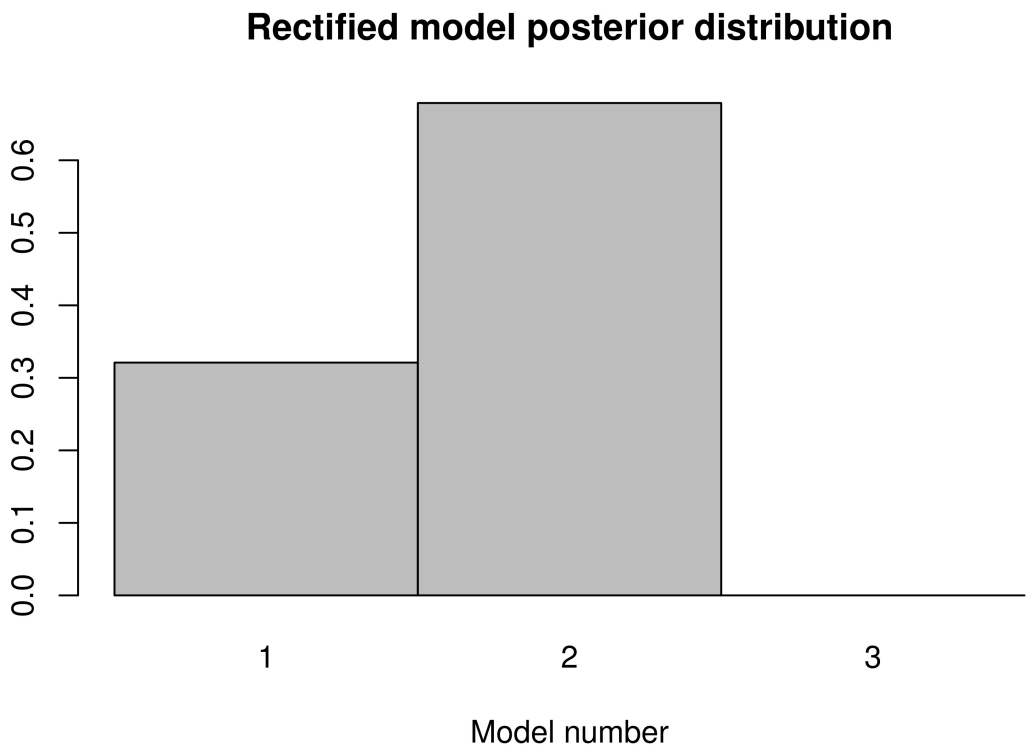
```
## assignments3
##    1    3
## 0.37 0.63
```

```
## theta.1 theta.2 theta.3
## 0.36911 0.00689 0.62138
```

Rectified model posterior distribution

Since there are empty components, we can rectify the model posterior distribution by summing the support model 3 got to the model 2 (from the half chain, the burn-in (typically the first half) is discarded).

```
barplot(c(0.321,0.261+0.418,0),space = 0,ylab = "", names.arg = c(1,2,3),
        xlab = "Model number", main ="Rectified model posterior distribution")
```




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

MSE of the estimated centers

```
x <- seq(0,2*pi,0.1)
centers2 <- data.frame(L1 = 2*sin(x),L2 = 2*cos(x))
estimates2.1 <- data.frame(L1_1)
estimates2.2 <- data.frame(L2_1,L2_2)
estimates2.3 <- data.frame(L3_1,L3_2,L3_3)

center_mse(centers2,estimates2.1)
```

```
## [[1]]
##     L1_1
## 1.562316
##
## [[2]]
##      L1_1
## 0.5690197
```

```
center_mse(centers2,estimates2.2)
```

```
## [[1]]
##        L2_1        L2_2
## 3.832934160 0.008205544
##
## [[2]]
##       L2_1       L2_2
## 0.00803746 3.84497009
```

```
center_mse(centers2,estimates2.3)
```

```
## [[1]]
##        L3_1        L3_2        L3_3
## 0.007204895 2.829084144 3.841779809
##
## [[2]]
##        L3_1        L3_2        L3_3
## 3.854768473 2.805883679 0.006823415
```

We see that the centres were estimated relatively accurately given the noise we simulated in to the data.