Capture-Recapture Example (Tutorial)

Leah South

Our running example is about the capture and recapture of the bird species called the European Dipper (*Cinclus cinclus*). Marzolin (1988) collected the data based on the capture and recapture of this species over six years.

Below are the R packages we'll be using in this document.

```r
library(MASS) # multivariate normal
library(coda) # assessing convergence and sample quality
library(psych) # bivariate plots
```

The MCMC Algorithm

We'll be looking at MALA (asymptotically un-baised) and ULA (biased) approaches for inference on our example. The focus is on post-processing and bias rather than implementation of algorithms. You can find the relevant code for MALA and ULA implementations for reference.

```r
source('MCMC_fn.R')
source('ULA_fn.R')
```

The Statistical Model

The parameters for the model are ϕ_i and p_k where $i = 1, \ldots, 6$ and $k = 2, \ldots, 7$. ϕ_i represents the probability of survival from year i to year $i+1$ and p_k represents the probability of being captured in year k.

The likelihood for the model is given below, and based on data D_i for the number of birds released in year i and y_{ik} for the number of animals caught in year k out of the number released in year i. Here $d_i = D_i - \sum_{k=i+1}^{7} y_{ik}$ is the number released in year i that are never caught. The corresponding probability of a bird being released in year i and never being caught is $\chi_i = 1 - \sum_{k=i+1}^{7} \phi_i p_k \prod_{m=i+1}^{k-1} \phi_m (1 - p_m)$, which is a function of the model parameters. The likelihood is given by

$$f(y|\theta) \propto \prod_{i=1}^{6} \chi_i^{d_i} \prod_{k=i+1}^{7} \left[\phi_i p_k \prod_{m=i+1}^{k-1} \phi_m (1 - p_m) \right]^{y_{ik}},$$

where $\theta = (\phi, p)$, $\phi = (\phi_1, ..., \phi_6)$, $p = (p_2, ..., p_7)$ and $y = \{y_{ik} : i = 1, \ldots, 6, k = 2, \ldots, 7\}$. Due to parameter identifiability issues, the parameters ϕ_6 and p_7 are combined as $\phi_6 p_7$ leading to a total of eleven parameters.

The prior for each component of θ is set to be $\mathcal{U}(0, 1)$, and all components are independent *a priori*. For the RW proposal, the j-th parameter $\theta[j]$ is transformed using $\tilde{\theta}[j] = \log(\theta[j]/(1 - \theta[j]))$ for $j = 1, \ldots, 11$. The implied prior density for $\tilde{\theta}[j]$ is then $e^{\tilde{\theta}[j]}/(1 + e^{\tilde{\theta}[j]})^2$, for $j = 1, \ldots, 11$.

Read in the liklihood functions and tuning parameters for algorithms

```
load("recapture_ULA_bettertuning.RData")
load("recapture_MALA_tuning.RData")

# Names of the 11 variables
varNames <- paste0("theta",1:11) #### FIX-LATER
```

Multiple Chains

Getting the samples

Let's run 10 chains with a common starting point

```
initial <- c(0.35, -0.66, -1.74, 2.5, -0.67, -0.59, 2.38, 2.52, 1.2, 5.08,
1.3)
set.seed(2)
n_reps <- 10 # number of chains
its <- 500 # number of MCMC iterations
chains_ULA <- chains_MALA <- samples_ULA <- samples_MALA <- list()
for (i in 1:n_reps){

  # Running MALA
  single_chain_mala <- MALA_fn(d = 11, initial = initial, covmala = cov_rw, h = h_mala,
                           iters = its,der_loglike = der_loglike,
                           der_logprior = der_logprior,
                           options = options, varNames = varNames)
  chains_MALA[[i]] <- single_chain_mala
  samples_MALA[[i]] <- single_chain_mala$samples

  # Running ULA
  single_chain_ula <- ULA_fn(d = 11, initial = initial, cov_ULA = cov_ula, h = h_ula,
                           iters = its,der_loglike = der_loglike,
                           der_logprior = der_logprior,
                           options = options, varNames = varNames)
  chains_ULA[[i]] <- single_chain_ula
  samples_ULA[[i]] <- single_chain_ula$samples

}
ula_noburnin <- as.mcmc.list(samples_ULA)
mala_noburnin <- as.mcmc.list(samples_MALA)
save(ula_noburnin,mala_noburnin,chains_MALA,chains_ULA, file = "ten_chains.RData") # For future reuse
```

Compare the KSD

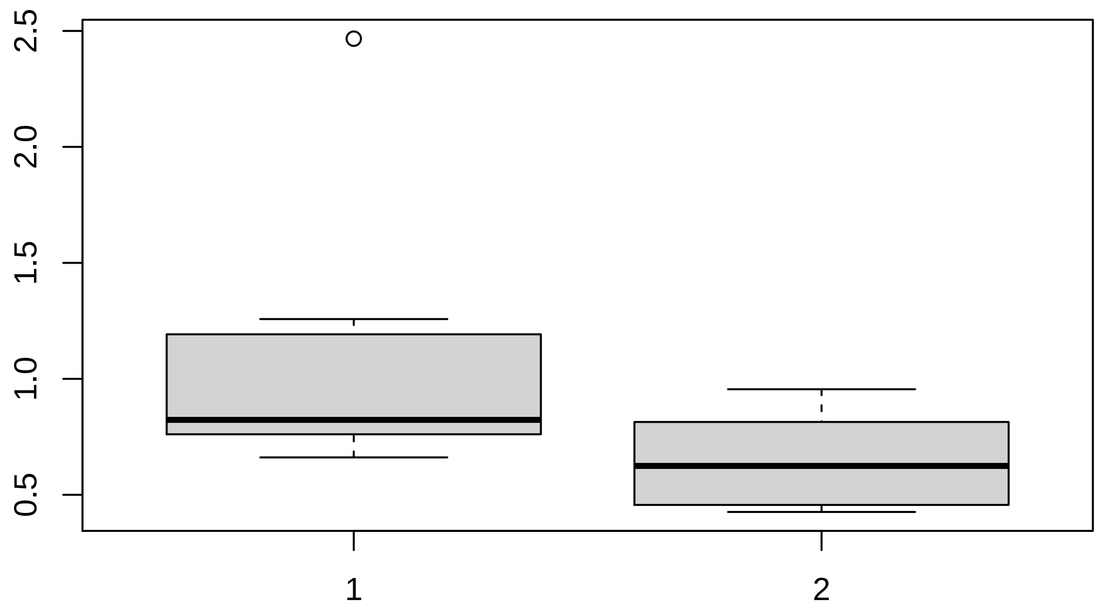
Sourcing in the KSD code and load the KSD package

```
source('KSD.R')
library(KSD)
```

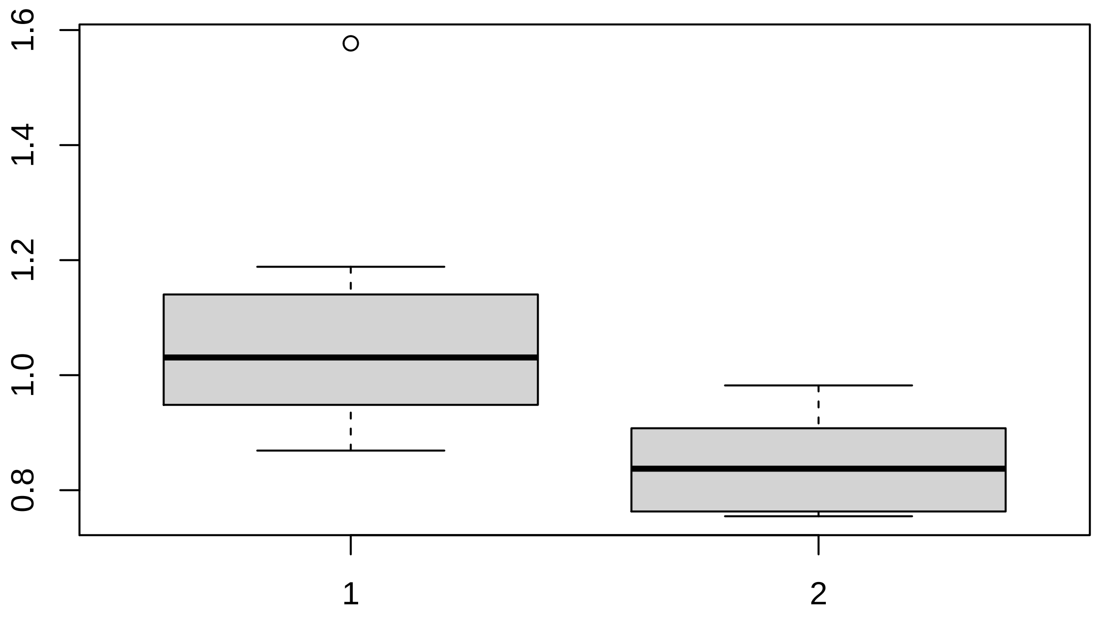
```
## Warning: package 'KSD' was built under R version 4.0.5
```

Evaluate the KSD on each of the chains for MALA and ULA.

```
KSD_MALA_gaussian <- KSD_ULA_gaussian <- rep(NaN,n_reps)
KSD_MALA_imq <- KSD_ULA_imq <- rep(NaN,n_reps)
for (i in 1:n_reps){
    samples <- as.matrix(chains_MALA[[i]]$samples)
  gradients <- chains_MALA[[i]]$der_loglike + chains_MALA[[i]]$der_logprior

  KSD_MALA_gaussian[i] <- KSD(samples, gradients)$ksd
  KSD_MALA_imq[i] <- imqKSD(samples, gradients)

  samples <- as.matrix(chains_ULA[[i]]$samples)
  gradients <- chains_ULA[[i]]$der_loglike + chains_ULA[[i]]$der_logprior

  KSD_ULA_gaussian[i] <- KSD(samples, gradients)$ksd
  KSD_ULA_imq[i] <- imqKSD(samples, gradients)
}
save(KSD_MALA_gaussian,KSD_ULA_gaussian,KSD_MALA_imq,KSD_ULA_imq, file = "ksd_ten_chains.RData")
boxplot(KSD_MALA_gaussian,KSD_ULA_gaussian)
```



```
boxplot(KSD_MALA_imq,KSD_ULA_imq)
```



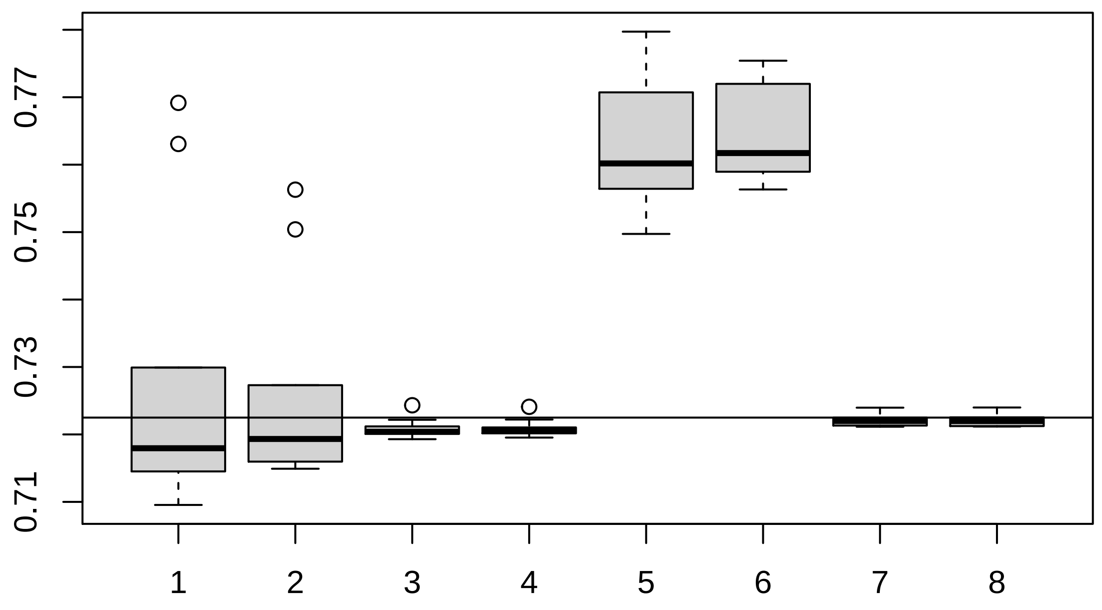
Estimating expectations with control variates

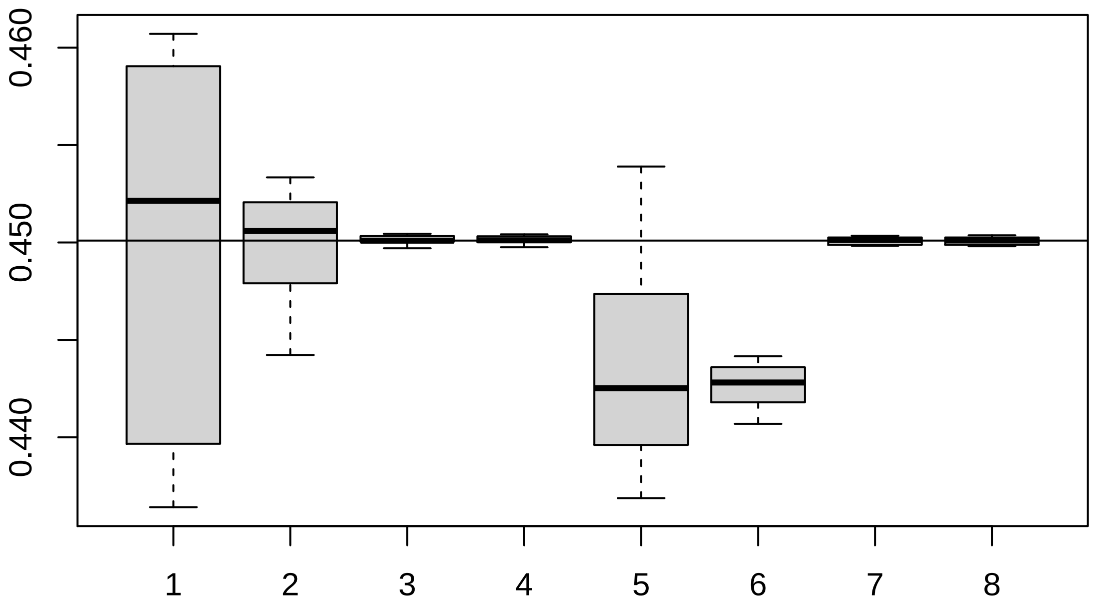
Now we'll estimate the posterior expectation of our parameters. The parameters are transformed using $\tilde{\theta}[j] = \log(\theta[j]/(1-\theta[j]))$ for $j = 1, \ldots, 11$. To transform back $\tilde{\theta}[j]$ we use $e^{\tilde{\theta}[j]}/(1+e^{\tilde{\theta}[j]})^2$, for $j = 1, \ldots, 11$.

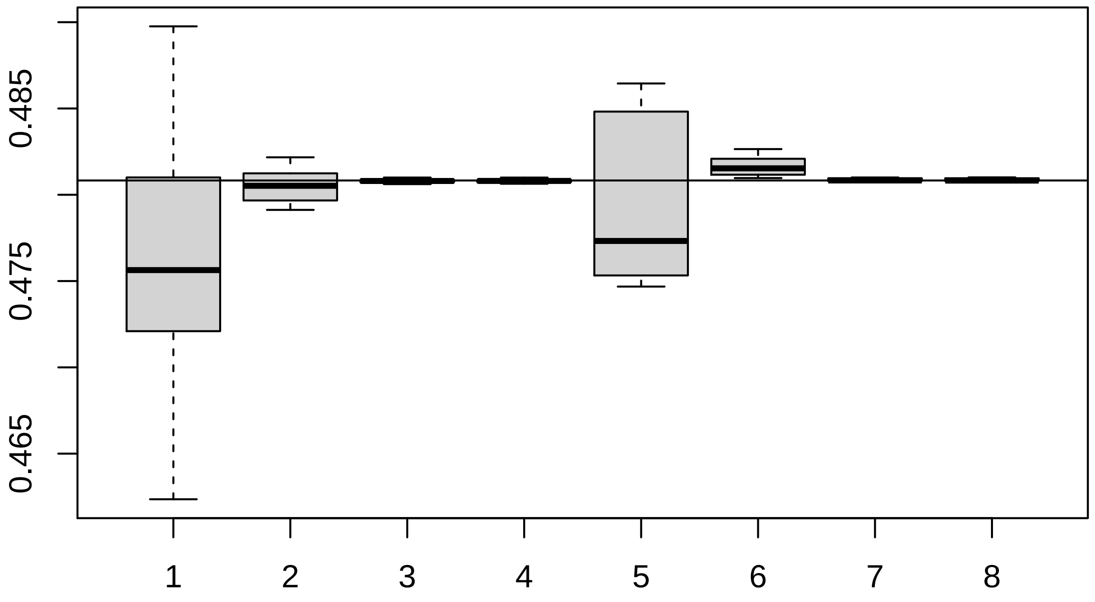
```
library(ZVCV)
```

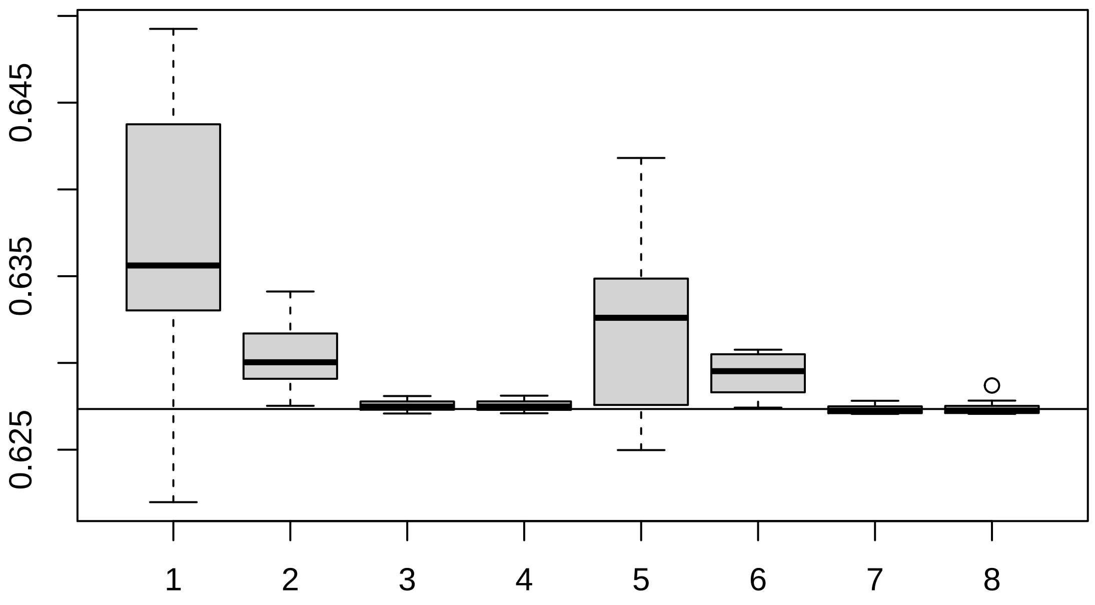
```
## Warning: package 'ZVCV' was built under R version 4.0.4
```

```
Vanilla_MALA <- ZV1_MALA <- CF_MALA <- SECF_MALA <- matrix(NaN,nrow=n_reps,ncol=d)
Vanilla_ULA <- ZV1_ULA <- CF_ULA <- SECF_ULA <- matrix(NaN,nrow=n_reps,ncol=d)
for (i in 1:n_reps){
  samples <- as.matrix(chains_MALA[[i]]$samples)
  gradients <- chains_MALA[[i]]$der_loglike + chains_MALA[[i]]$der_logprior
  integrand <- 1/(1+exp(-samples))

  # In order: vanilla estimate, zero-variance control variates
  # with a first order polynomila, control functionals and
  # semi-exact control functionals with a first order polynomial
  Vanilla_MALA[i,] <- colMeans(integrand)
  ZV1_MALA[i,] <- zvcv(integrand, samples, gradients,
                       options = list(polyorder = 1, regul_reg = FALSE))$expectation
  CF_MALA[i,] <- CF_crossval(integrand, samples, gradients, kernel_function = "RQ",
                             sigma_list = list(0.001,0.01,0.1,1,10), folds = 2)$expectation
  SECF_MALA[i,] <- SECF_crossval(integrand, samples, gradients, polyorder = 1, kernel_function = "RQ",
                                 sigma_list = list(0.001,0.01,0.1,1,10), folds = 2)$expectation
  # In order: vanilla estimate, zero-variance control variates
  # with a first order polynomila, control functionals and
  # semi-exact control functionals with a first order polynomial
  samples <- as.matrix(chains_ULA[[i]]$samples)
  gradients <- chains_ULA[[i]]$der_loglike + chains_ULA[[i]]$der_logprior
  integrand <- 1/(1+exp(-samples))

  Vanilla_ULA[i,] <- colMeans(integrand)
  ZV1_ULA[i,] <- zvcv(integrand, samples, gradients,
                      options = list(polyorder = 1, regul_reg = FALSE))$expectation
  CF_ULA[i,] <- CF_crossval(integrand, samples, gradients, kernel_function = "RQ",
                            sigma_list = list(0.001,0.01,0.1,1,10), folds = 2)$expectation
  SECF_ULA[i,] <- SECF_crossval(integrand, samples, gradients, polyorder = 1, kernel_function = "RQ",
                                sigma_list = list(0.001,0.01,0.1,1,10), folds = 2)$expectation
}
load("Recapture_goldstandard.RData")
# Boxplots of the estimates
for (j in 1:11){
  boxplot(Vanilla_MALA[,j],ZV1_MALA[,j],CF_MALA[,j],SECF_MALA[,j],
          Vanilla_ULA[,j],ZV1_ULA[,j],CF_ULA[,j],SECF_ULA[,j])
  abline(h=gold_standard[j])
}
```

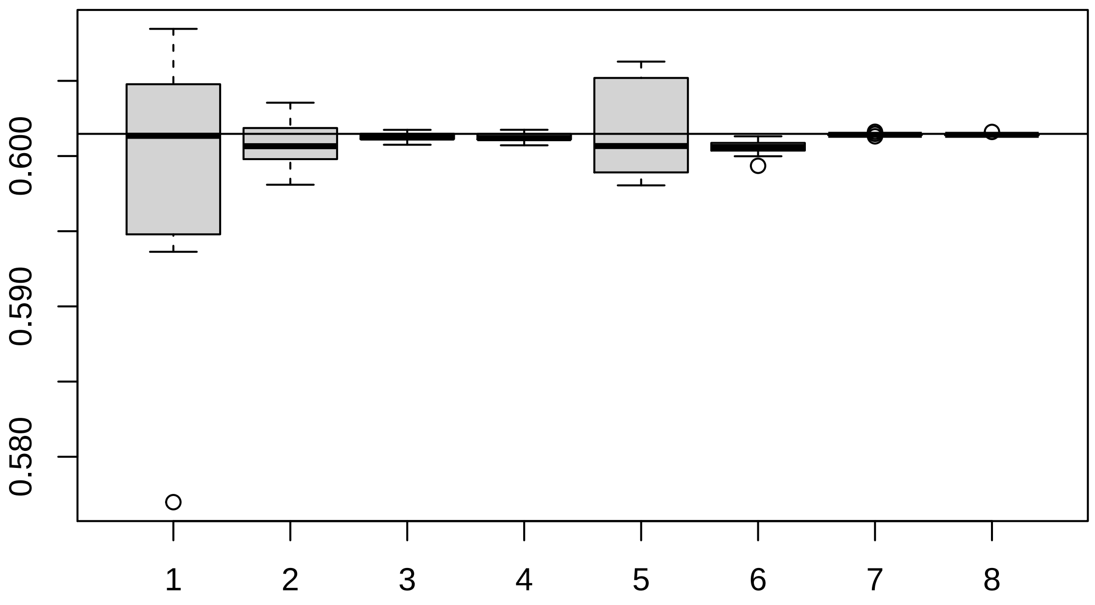


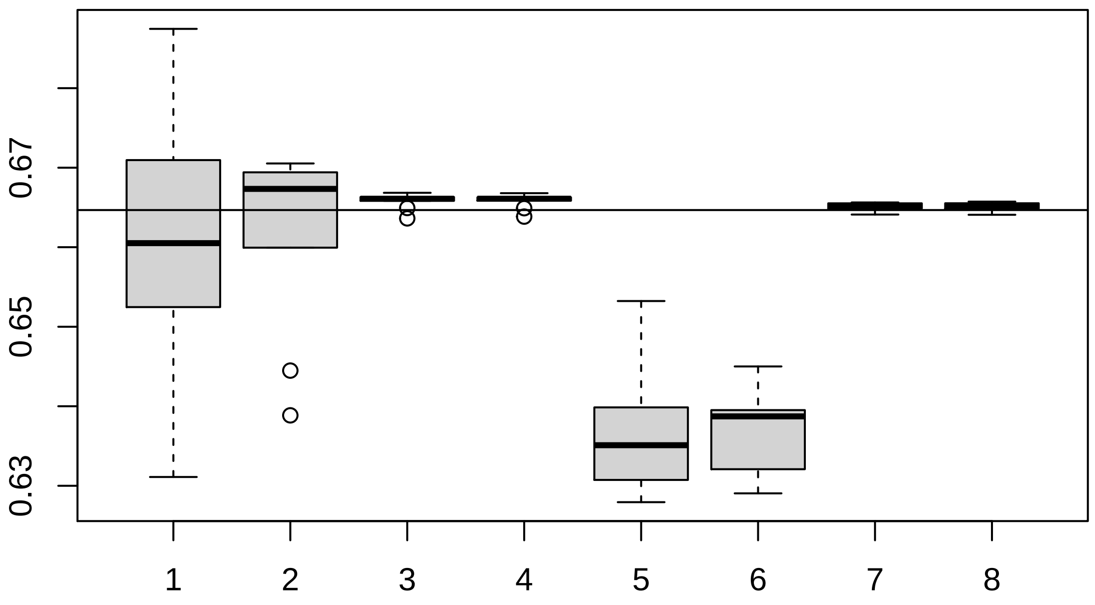

0.460
0.450
0.440
1
2
3
4
5
6
7
8

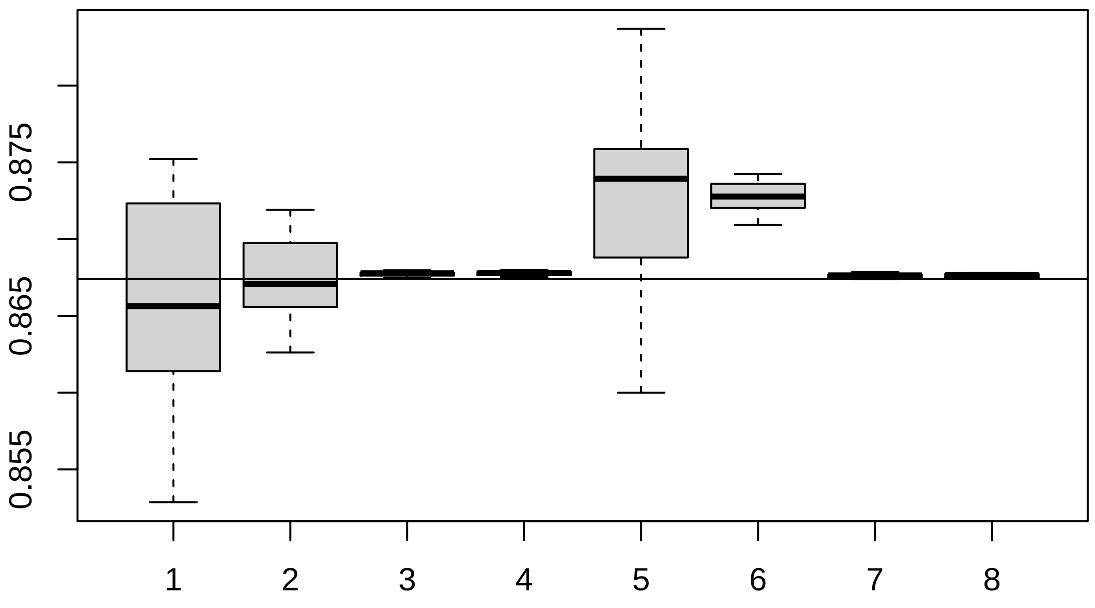

0.465
0.475
0.485
1
2
3
4
5
6
7
8

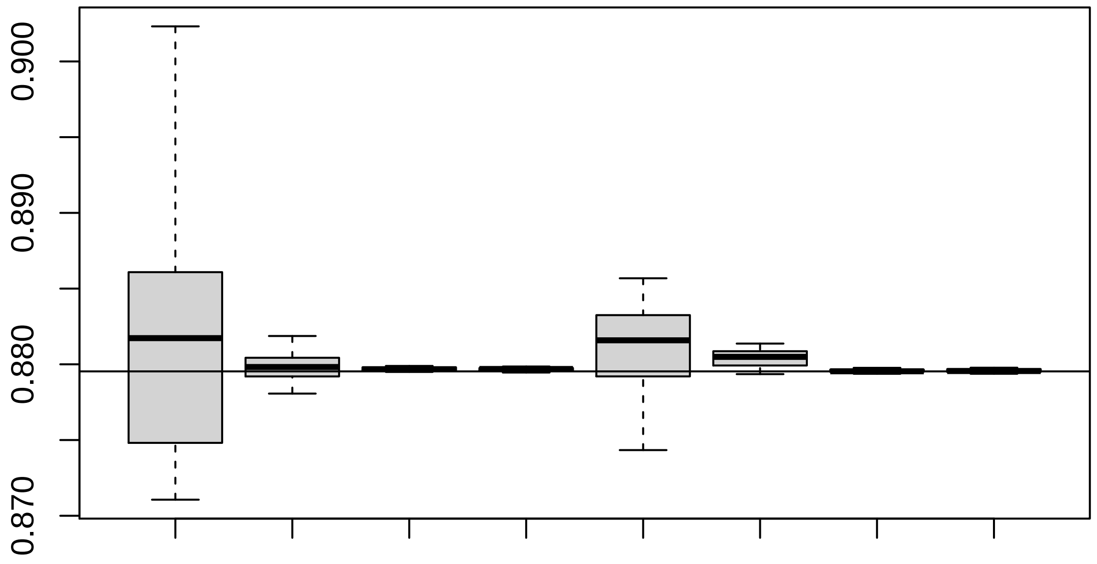


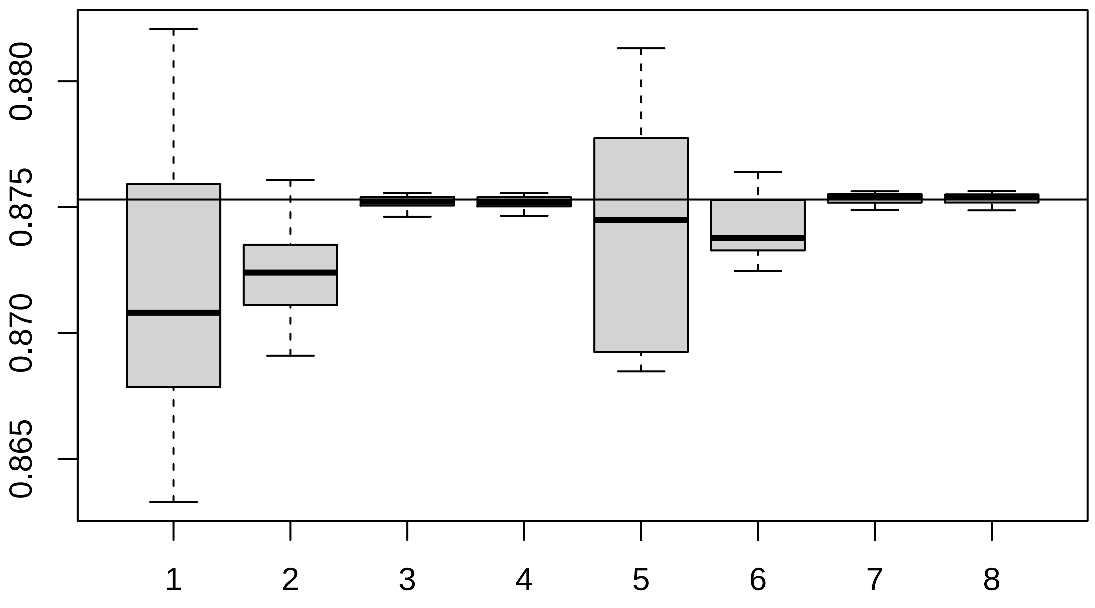
0.645
0.635
0.625
1
2
3
4
5
6
7
8

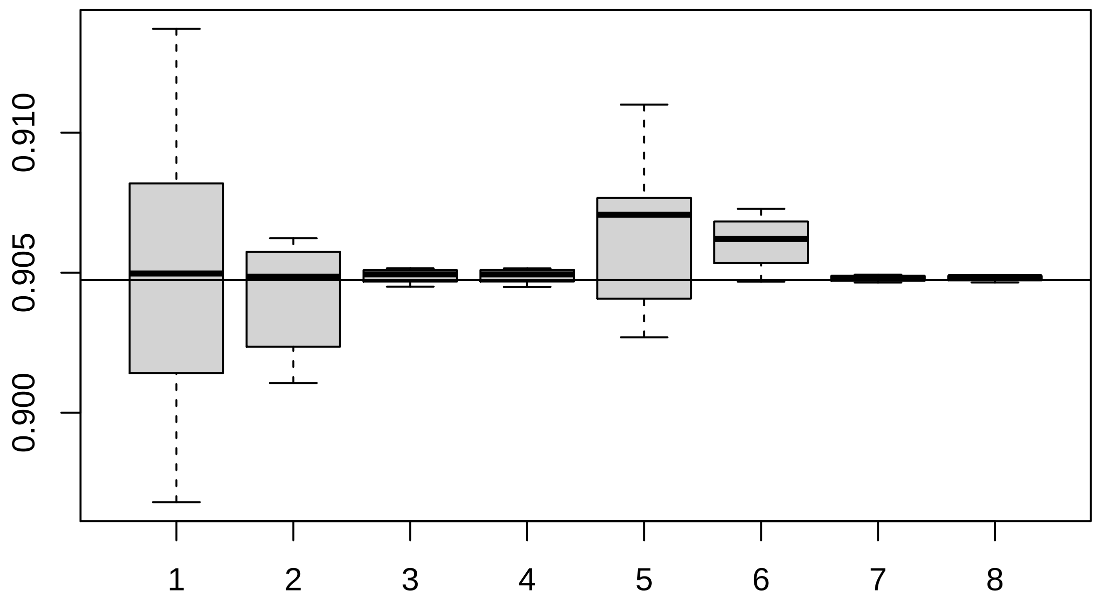

0.580
0.590
0.600
1
2
3
4
5
6
7
8

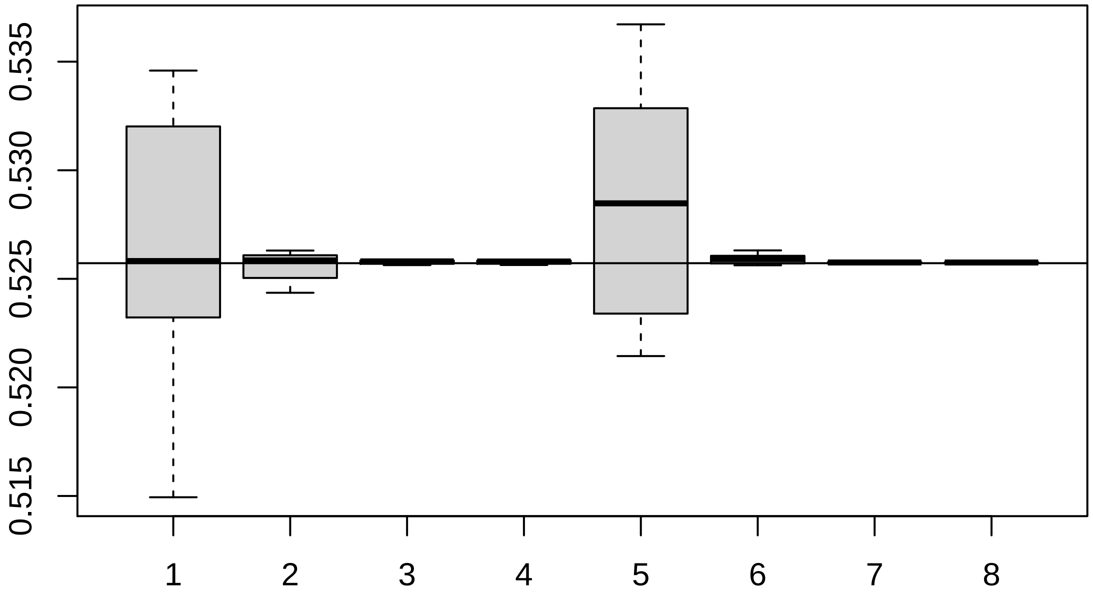

0.63
0.65
0.67
1
2
3
4
5
6
7
8


0.855
0.865
0.875
1
2
3
4
5
6
7
8


0.870
0.880
0.890
0.900
1
2
3
4
5
6
7
8


0.865
0.870
0.875
0.880
1
2
3
4
5
6
7
8


0.900
0.905
0.910
1
2
3
4
5
6
7
8



Investigate tuning parameter for ULA

Try increasing and decreasing the parameter h_ula.

```r
set.seed(2)
h_value <- 0.5
# Running ULA
single_chain_ula <- ULA_fn(d = 11, initial = initial, cov_ULA = cov_ula, h = h_value,
                            iters = its,der_loglike = der_loglike,
                            der_logprior = der_logprior,
                            options = options, varNames = varNames)
samples <- as.matrix(single_chain_ula$samples)
gradients <- single_chain_ula$der_loglike + single_chain_ula$der_logprior

KSD(samples, gradients)$ksd
```

```
## [1] 2.943607
```

```r
imqKSD(samples, gradients)
```

```
## [1] 1.710645
```

```r
h_value <- 1
# Running ULA
single_chain_ula <- ULA_fn(d = 11, initial = initial, cov_ULA = cov_ula, h = h_value,
                            iters = its,der_loglike = der_loglike,
                            der_logprior = der_logprior,
                            options = options, varNames = varNames)
```

```
samples <- as.matrix(single_chain_ula$samples)
gradients <- single_chain_ula$der_loglike + single_chain_ula$der_logprior

KSD(samples, gradients)$ksd
```

```
## [1] 0.4114754
```

```
imqKSD(samples, gradients)
```

```
## [1] 0.7582476
```

```
h_value <- 1.6
# Running ULA
single_chain_ula <- ULA_fn(d = 11, initial = initial, cov_ULA = cov_ula, h = h_value,
                           iters = its,der_loglike = der_loglike,
                           der_logprior = der_logprior,
                           options = options, varNames = varNames)
samples <- as.matrix(single_chain_ula$samples)
gradients <- single_chain_ula$der_loglike + single_chain_ula$der_logprior

KSD(samples, gradients)$ksd
```

```
## [1] 6.967222
```

```
imqKSD(samples, gradients)
```

```
## [1] 1.599452
```

Investigate convergence for MALA

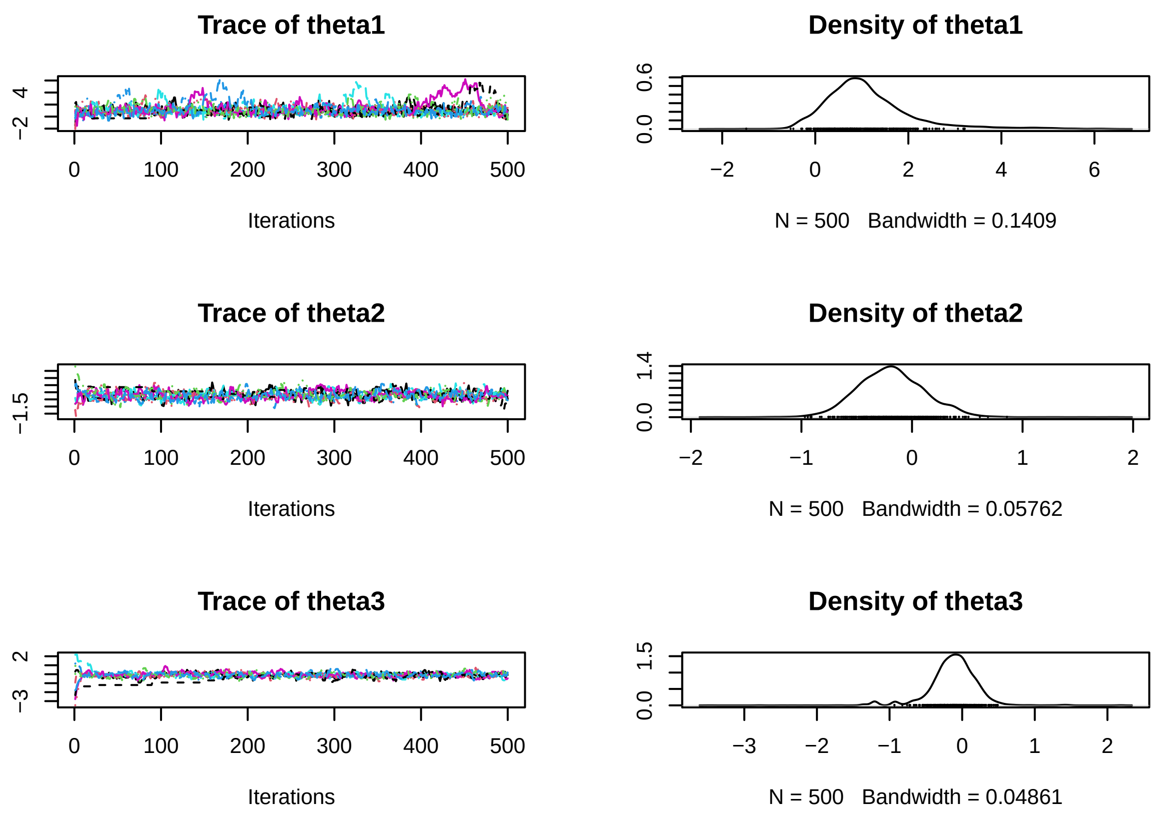
Try alternative initialisation points you may use the following to simulate from the prior:

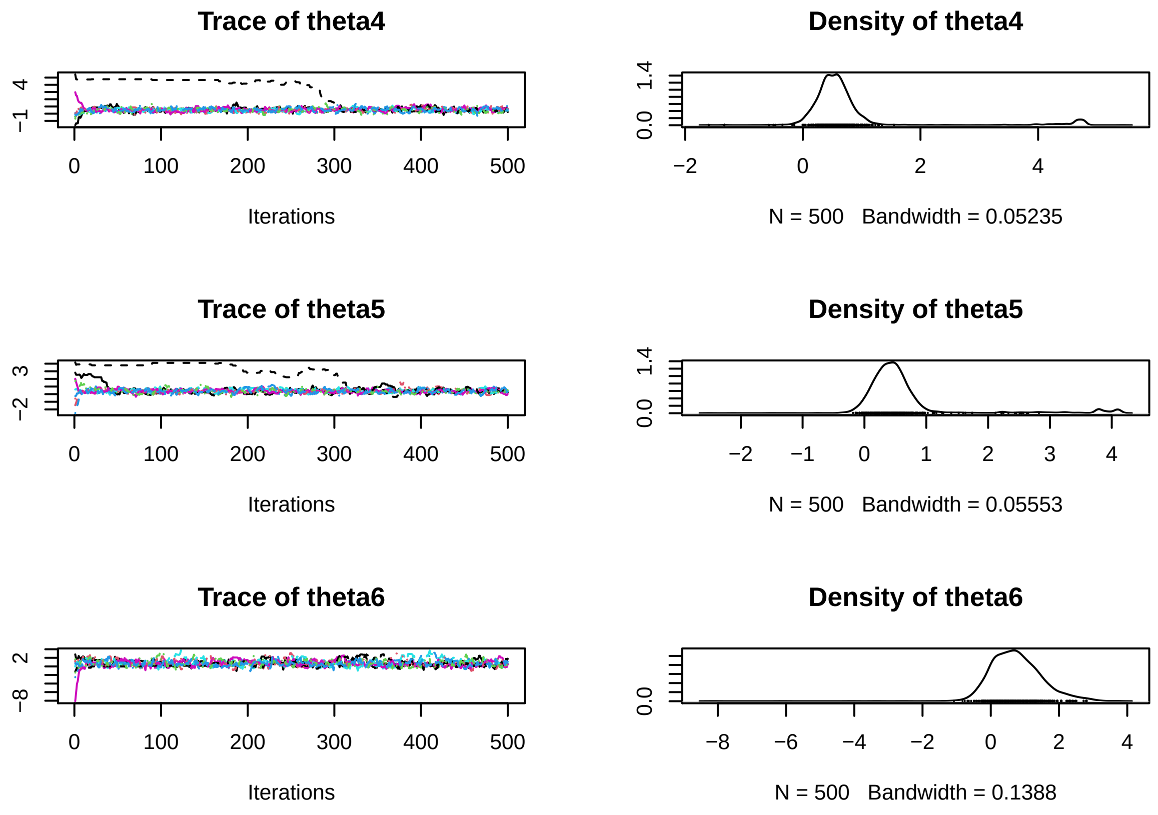
```
initial <- recapture_simprior(1, options)
```

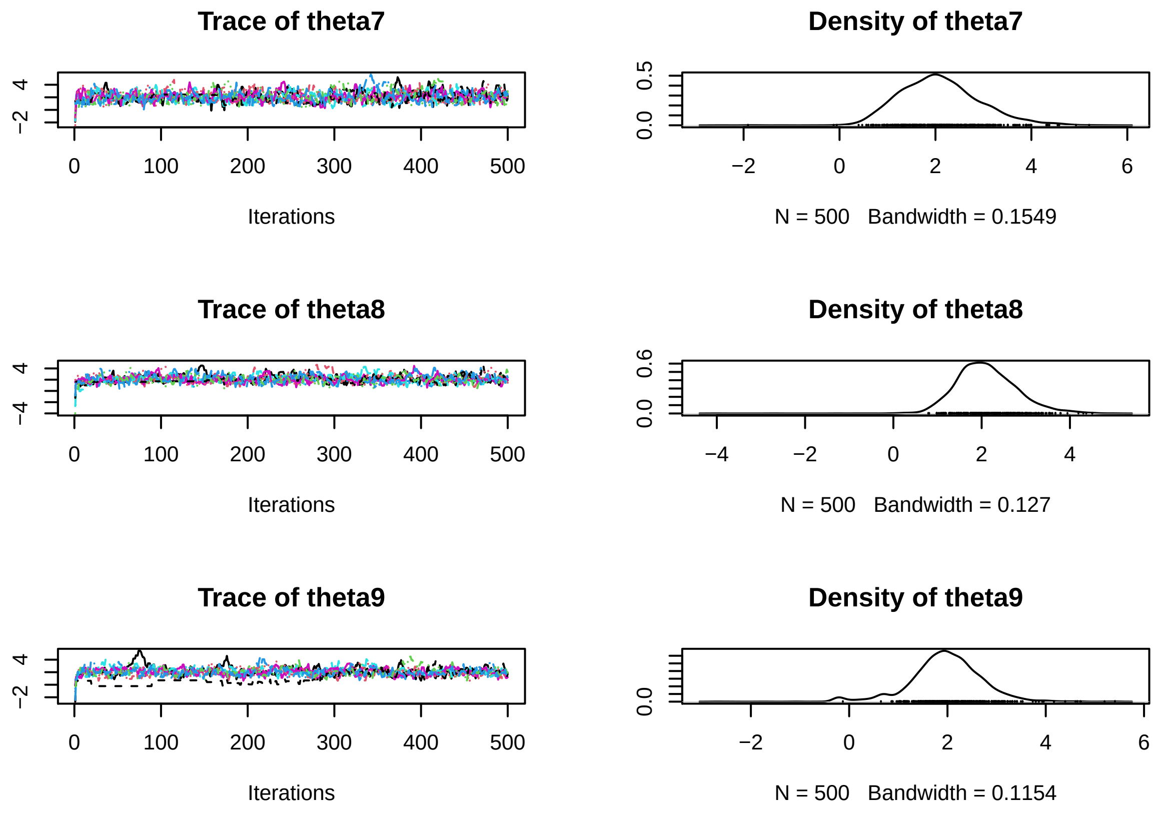
```
set.seed(2)
n_reps <- 10 # number of chains
its <- 500 # number of MCMC iterations
chains_ULA <- chains_MALA <- samples_ULA <-
  samples_MALA <- list()
for (i in 1:n_reps){
  initial <- recapture_simprior(1, options)
  # Running MALA
  single_chain_mala <- MALA_fn(d = 11, initial = initial, covmala = cov_rw, h = h_mala,
                           iters = its,der_loglike = der_loglike,
                           der_logprior = der_logprior,
                           options = options, varNames = varNames)
  chains_MALA[[i]] <- single_chain_mala
  samples_MALA[[i]] <- single_chain_mala$samples

  # Running ULA
  single_chain_ula <- ULA_fn(d = 11, initial = initial, cov_ULA = cov_ula, h = h_ula,
                           iters = its,der_loglike = der_loglike,
                           der_logprior = der_logprior,
                           options = options, varNames = varNames)
  chains_ULA[[i]] <- single_chain_ula
  samples_ULA[[i]] <- single_chain_ula$samples
}
ula_noburnin <- as.mcmc.list(samples_ULA)
mala_noburnin <- as.mcmc.list(samples_MALA)
```

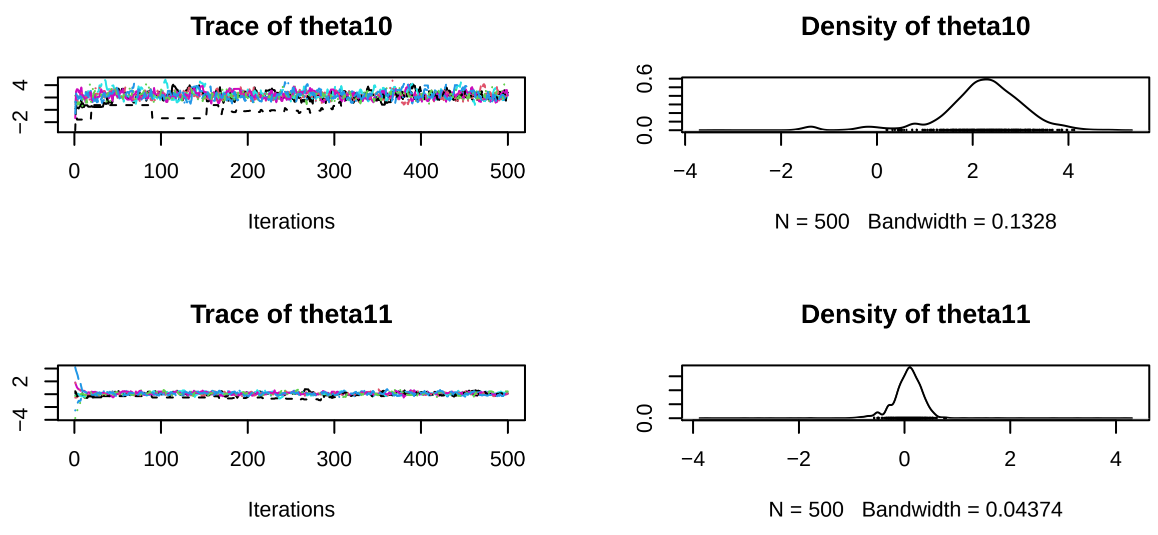
Investigate the convergence using multiple chains and Gelman & Rubin's $\hat{R}$ diagnostic. Work out a good burnin and investigate control variates on the burnin chain.

```
plot(mala_noburnin,smooth=FALSE)
```




Trace of theta4
Density of theta4
Iterations
N = 500   Bandwidth = 0.05235
Trace of theta5
Density of theta5
Iterations
N = 500   Bandwidth = 0.05553
Trace of theta6
Density of theta6
Iterations
N = 500   Bandwidth = 0.1388


Trace of theta7
Iterations
Density of theta7
N = 500 Bandwidth = 0.1549
Trace of theta8
Iterations
Density of theta8
N = 500 Bandwidth = 0.127
Trace of theta9
Iterations
Density of theta9
N = 500 Bandwidth = 0.1154



Rhat

The $\hat{R}$ diagnostic is not < 1.1 for all dimensions, so we have evidence of non-convergence.

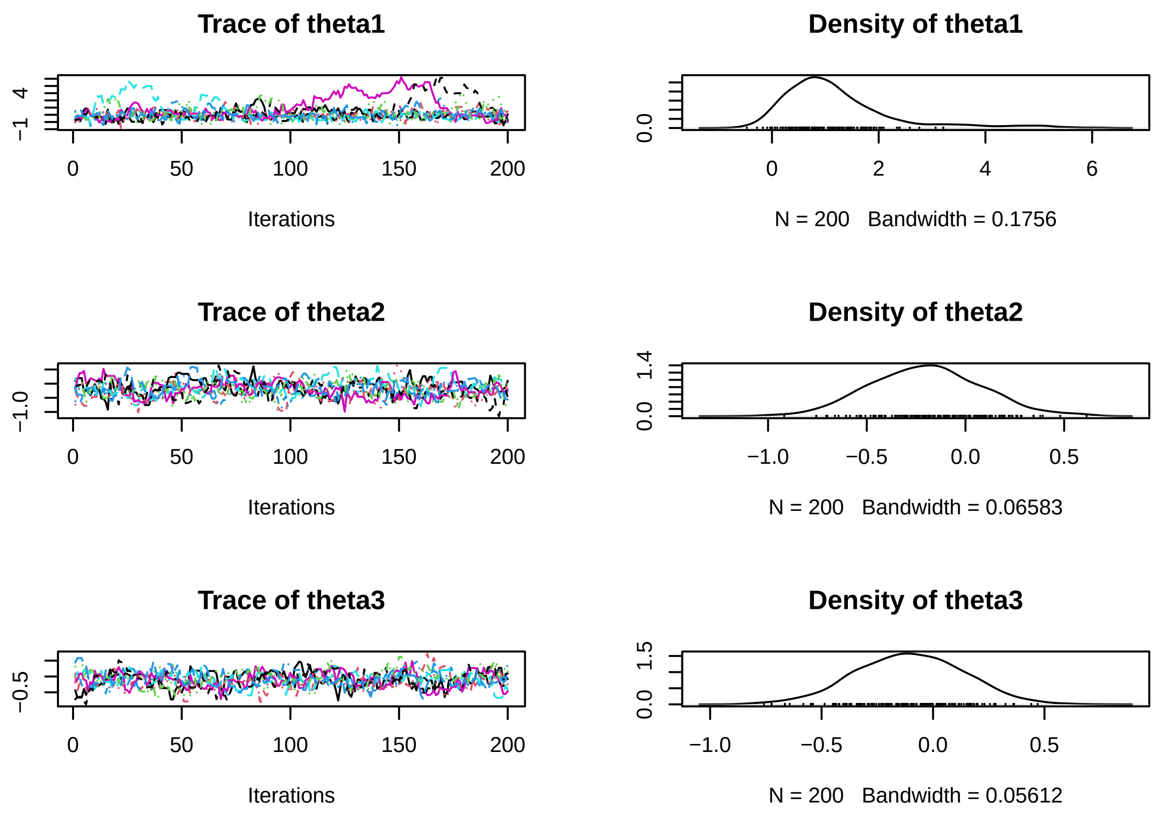
```
gelman.diag(mala_noburnin,autoburnin=FALSE)
```

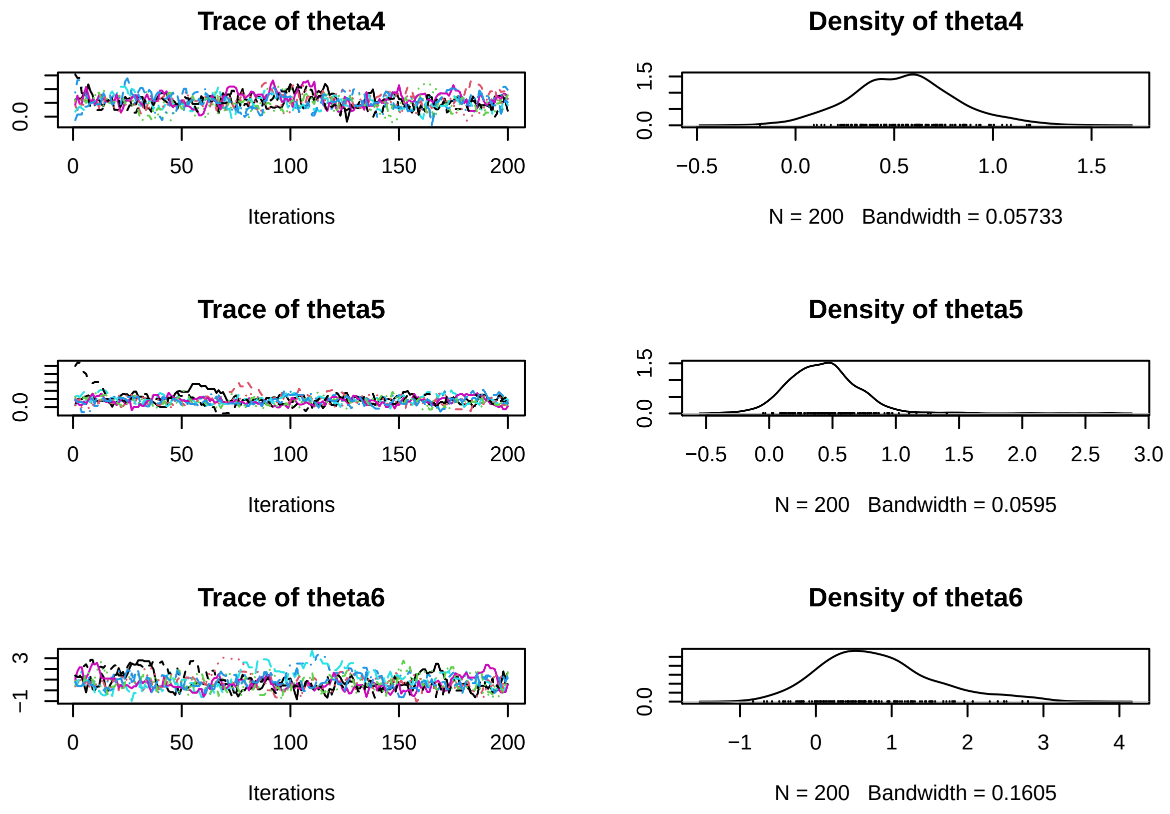
```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## theta1        1.03       1.04
## theta2        1.02       1.04
## theta3        1.11       1.22
## theta4        1.80       5.48
## theta5        1.70       3.80
## theta6        1.02       1.03
## theta7        1.01       1.02
## theta8        1.01       1.03
## theta9        1.17       1.33
## theta10       1.31       1.62
## theta11       1.12       1.24
##
## Multivariate psrf
##
## 1.57
```

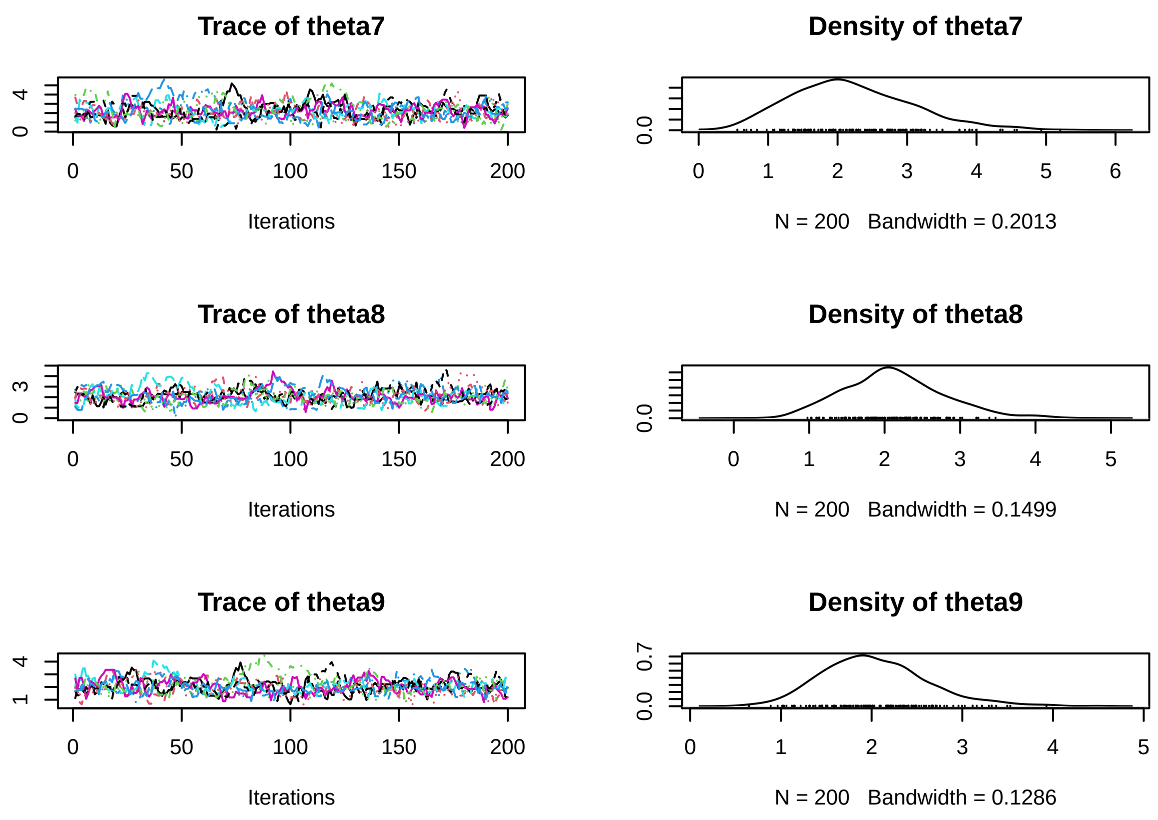
Removing Burn-In

Let's see if we still have evidence of non-convergence if we remove the first 500 samples as burn-in.

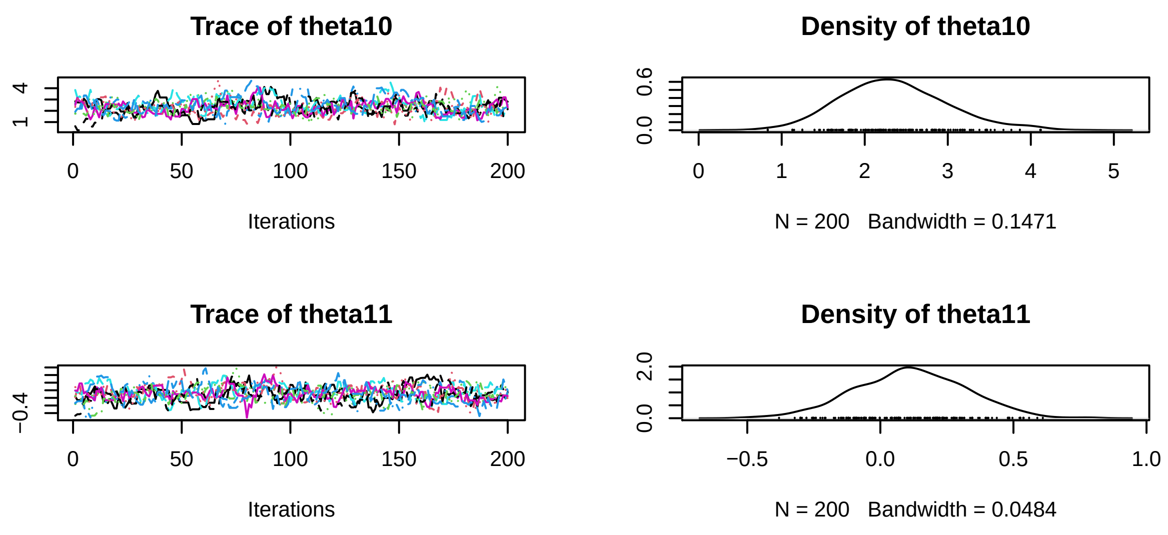
```
# Removing burn-in
burnin <- 300
its <- 500
mala_burnin <- list()
for (i in 1:n_reps){
  mala_burnin[[i]] <- mcmc(mala_noburnin[[i]][(burnin+1):(its),])
}
mala_burnin <- as.mcmc.list(mala_burnin)

# Plotting chains
plot(mala_burnin,smooth=FALSE)
```




Trace of theta4
0.0
0
50
100
150
200
Iterations
Density of theta4
1.5
0.0
−0.5
0.0
0.5
1.0
1.5
N = 200 Bandwidth = 0.05733
Trace of theta5
0.0
0
50
100
150
200
Iterations
Density of theta5
1.5
0.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
N = 200 Bandwidth = 0.0595
Trace of theta6
3
−1
0
50
100
150
200
Iterations
Density of theta6
0.0
−1
0
1
2
3
4
N = 200 Bandwidth = 0.1605



Trace of theta7
Density of theta7
Iterations
N = 200 Bandwidth = 0.2013
Trace of theta8
Density of theta8
Iterations
N = 200 Bandwidth = 0.1499
Trace of theta9
Density of theta9
Iterations
N = 200 Bandwidth = 0.1286



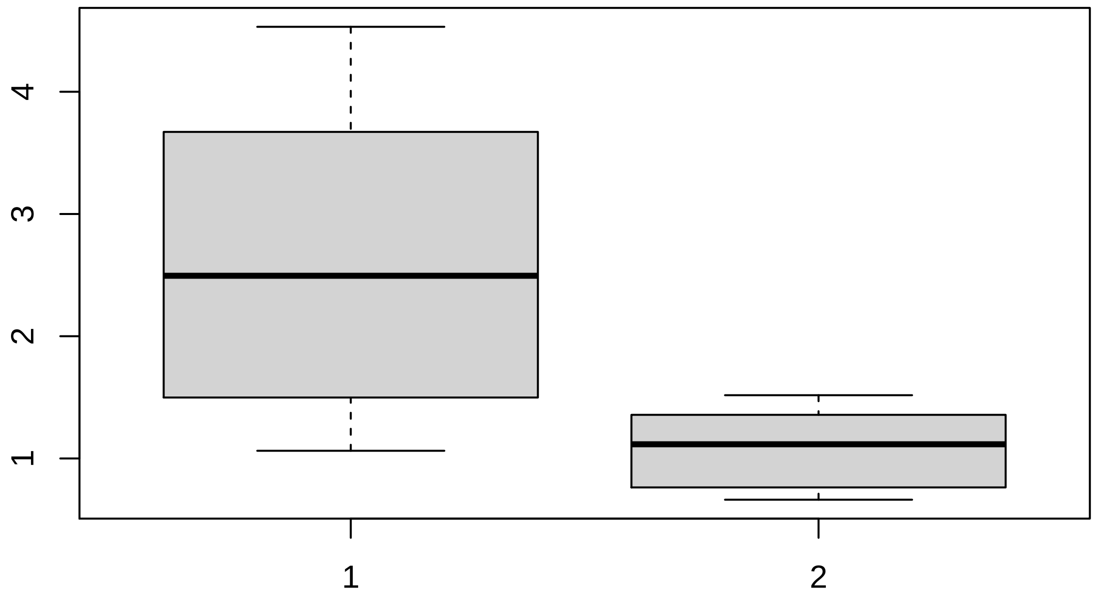
```
# Calculating Rhat
gelman.diag(mala_burnin,autoburnin=FALSE)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## theta1        1.09       1.19
## theta2        1.03       1.06
## theta3        1.02       1.05
## theta4        1.03       1.07
## theta5        1.04       1.08
## theta6        1.04       1.07
## theta7        1.03       1.05
## theta8        1.02       1.04
## theta9        1.03       1.06
## theta10       1.02       1.05
## theta11       1.01       1.03
##
## Multivariate psrf
##
## 1.09
```

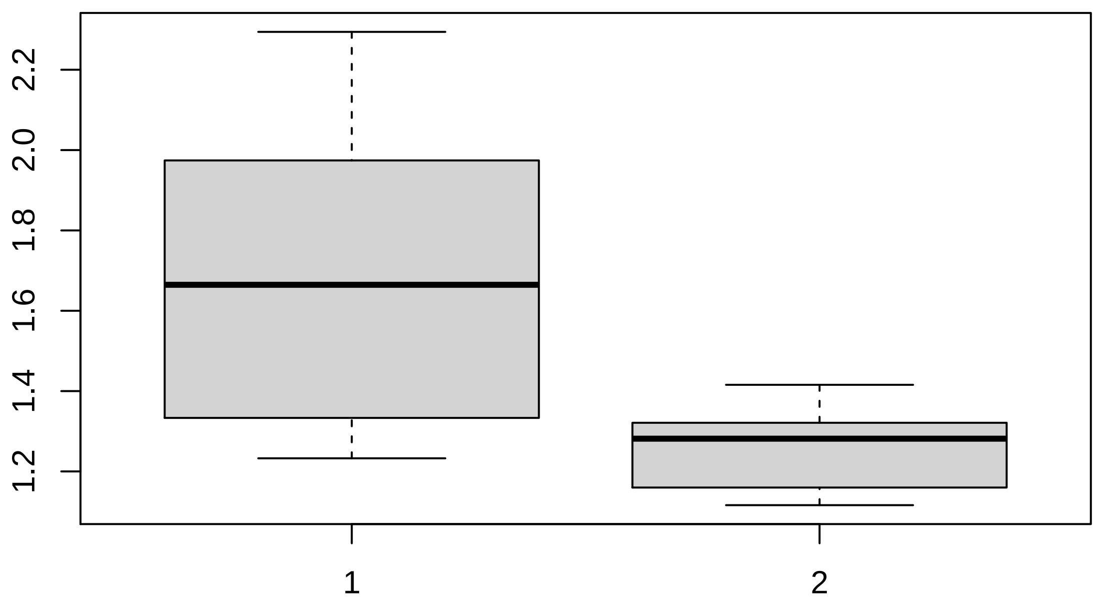
Investigate the KSD without bias:

```
burnin <- 300
KSD_MALA_gaussian <- KSD_ULA_gaussian <- rep(NaN,n_reps)
KSD_MALA_imq <- KSD_ULA_imq <- rep(NaN,n_reps)
```

```
for (i in 1:n_reps){
  samples <- as.matrix(chains_MALA[[i]]$samples[(burnin+1):(its),])
  gradients <- chains_MALA[[i]]$der_loglike[(burnin+1):(its),] + chains_MALA[[i]]$der_logprior[(burnin+

  KSD_MALA_gaussian[i] <- KSD(samples, gradients)$ksd
  KSD_MALA_imq[i] <- imqKSD(samples, gradients)

  samples <- as.matrix(chains_ULA[[i]]$samples[(burnin+1):(its),])
  gradients <- chains_ULA[[i]]$der_loglike[(burnin+1):(its),] + chains_ULA[[i]]$der_logprior[(burnin+1)

  KSD_ULA_gaussian[i] <- KSD(samples, gradients)$ksd
  KSD_ULA_imq[i] <- imqKSD(samples, gradients)
}
boxplot(KSD_MALA_gaussian,KSD_ULA_gaussian)
```



```
boxplot(KSD_MALA_imq,KSD_ULA_imq)
```


1.2
1.4
1.6
1.8
2.0
2.2
1
2