Project6

Jieran Sun, Hui Jeong (HJ) Jung, Gudmundur Björgvin Magnusson

2023-03-28

Problem 15

To prove that $E[\hat{g}(X)] = E[g(X)]$,

$$
\begin{aligned}
E[\hat{g}(X)] &= \int \hat{g}(x) f(x) dx && (1)\\
&= \int \left(\frac{1}{N} \sum_{i=1}^{N} g(X_i) \right) f(x) dx && (2)\\
&= \frac{1}{N} \sum_{i=1}^{N} \int g(X_i) f(x) dx && (3)\\
&= \frac{1}{N} \sum_{i=1}^{N} E[g(X_i)] && (4)\\
&= \frac{1}{N} N E[g(X)] \quad \text{(since } g(X_1), \ldots, g(X_N) \text{ are from the same distribution)} && (5)\\
&= E[g(X)] && (6)
\end{aligned}
$$

To further prove that $Var(\hat{g}(X)) = \frac{Var(g(X))}{N}$

$$
\begin{aligned}
\text{Var}(\hat{g}(X)) &= \text{Var} \left(\frac{1}{N} \sum_{i=1}^{N} g(X_i) \right) && (7)\\
&= \frac{1}{N^2} \text{Var} \left(\sum_{i=1}^{N} g(X_i) \right) && (8)\\
&= \frac{1}{N^2} \left(\sum_{i=1}^{N} \text{Var}(X_i) + \sum_{i,j=1}^{N} \text{Cov}(X_i, X_j) \right) \text{(from Bienaymé's identity)} && (9)\\
&= \frac{1}{N^2} N \text{Var}(X) = \frac{\text{Var}(X)}{N} && (10)
\end{aligned}
$$

If $X_1, ..., X_N$ is generated from a MCMC sampler, $E[\hat{g}(X)] = E[g(X)]$ will hold as the values X that are sampled will eventually converge to an identical distribution. However $Var(\hat{g}(X)) = \frac{Var(g(X))}{N}$ will most likely not hold as according to the nature of an MCMC sampler, the samples are not independent from each other and the current sample depends on the previous sample, and thus there will be some covariance resulting between two consecutive samples.

Question 16

(A) Expression for probability

First, we define the notations as $X = T, \neg X = F$,

First for:

$$P(C|R,S,W)$$

We have:

$$\frac{P(C)*P(S|C)*P(R|C)}{P(C)*P(S|C)*P(R|C)+P(\neg C)*P(S|\neg C)*P(R|\neg C)} = \frac{0.5*0.1*0.8}{0.5*0.1*0.8+0.5*0.5*0.2} = 0.4444$$

and for the other, we similarly have:

$$P(C|\neg R,S,W) = \frac{P(C)*P(S|C)*P(\neg R|C)}{P(C)*P(S|C)*P(\neg R|C)+P(\neg C)*P(S|\neg C)*P(\neg R|\neg C)} = \frac{0.5*0.1*0.2}{0.5*0.1*0.2+0.5*0.5*0.8} = 0.0476190$$

For Rain conditional probilities:

$$P(R|C,S,W)$$

We have:

$$\frac{P(R|C)*P(W|R,S)}{P(R|C)*P(W|R,S)+P(\neg R|C)*P(W|\neg R,S)} = \frac{0.8*0.99}{0.8*0.99+0.2*0.9} = 0.8148$$

and for

$$P(R|\neg C,S,W)$$

We have:

$$\frac{P(R|\neg C)*P(W|R,S)}{P(R|\neg C)*P(W|R,S)+P(\neg R|\neg C)*P(W|\neg R,S)} = \frac{0.2*0.99}{0.2*0.99+0.8*0.9} = 0.21568$$

(B) Gibbs sampler

We set up the Gibbs sampler and run the first Gibbs sample for 100 steps.

```
set.seed(2023)

# From the descrete probabilities we can tell the joint distribution of C and R given W,S = T
# Each probability case is calculated based on Bayesian network joint probability function

gibbsOldDONOTUSE <- function(n){

  # row is R = (F,T), column is C = (F,T)
  jointMat <- matrix(c(0.18, 0.009, 0.0495, 0.0396), ncol = 2)
  jointMat <- jointMat/sum(jointMat)
```

```
  # initiation
  CProb <- rep(NA, n)
  RProb <- rep(NA, n)

  CProb[1] <- rbinom(1,1,0.5)
  RProb[1] <- rbinom(1,1,0.5)

  # Assigning probability in each round
  for (i in 2:n){
    if (runif(1) <= 0.5) {
      CProb[i] <- rbinom(1, 1, jointMat[RProb[i-1] + 1, 2]/sum(jointMat[,2]))
      RProb[i] <- RProb[i-1]
    } else {
      RProb[i] <- rbinom(1, 1, jointMat[2, CProb[i-1] + 1]/sum(jointMat[2,]))
      CProb[i] <- CProb[i-1]
    }
  }

  resultDT <- data.table(Cloudy = CProb, Rain = RProb)
  return(resultDT)
}

gibbsRain <- function(number_samples){

    # Constants
    P_C = 0.5
    P_S_g_C = matrix(c(1,0,0.1,0.5,0.9,0.5),ncol = 3)
    P_R_g_C = matrix(c(1,0,0.8,0.2,0.2,0.8),ncol = 3)
    P_G_g_S_R = matrix(c(1,1,0,0,1,0,1,0,
                         0.99,0.90,0.90,0.01,0.01,0.1,0.1,0.99),ncol = 4)

    # INITALIZE

    states <- matrix(T, number_samples,
                     ncol = 2,
                     dimnames = list(c(),c("Rain","Cloudy")))


    for (i in 1:(number_samples-1)) {
      if (runif(1) >= 0.5) {
        # Update Rain
        a <- P_R_g_C[2-states[i,"Cloudy"],2]*P_G_g_S_R[1,3]
        b <- a + P_R_g_C[2-states[i,"Cloudy"],3]*P_G_g_S_R[2,3]

        states[i+1,"Rain"] <- runif(1) <= a/b
        states[i+1,"Cloudy"] <- states[i,"Cloudy"]

      } else {
        # Update Cloudy
        a <- P_C*P_S_g_C[1,2]*P_R_g_C[1,3-states[i,"Rain"]]
        b <- a + (1-P_C)*P_S_g_C[2,2]*P_R_g_C[2,3-states[i,"Rain"]]

        states[i+1,"Cloudy"] <- runif(1) <= a/b
```

```
        states[i+1,"Rain"] <- states[i,"Rain"]

  }
}
return (states)
}

gibbSamples <- gibbsRain(100)
```

(C) Marginal probability

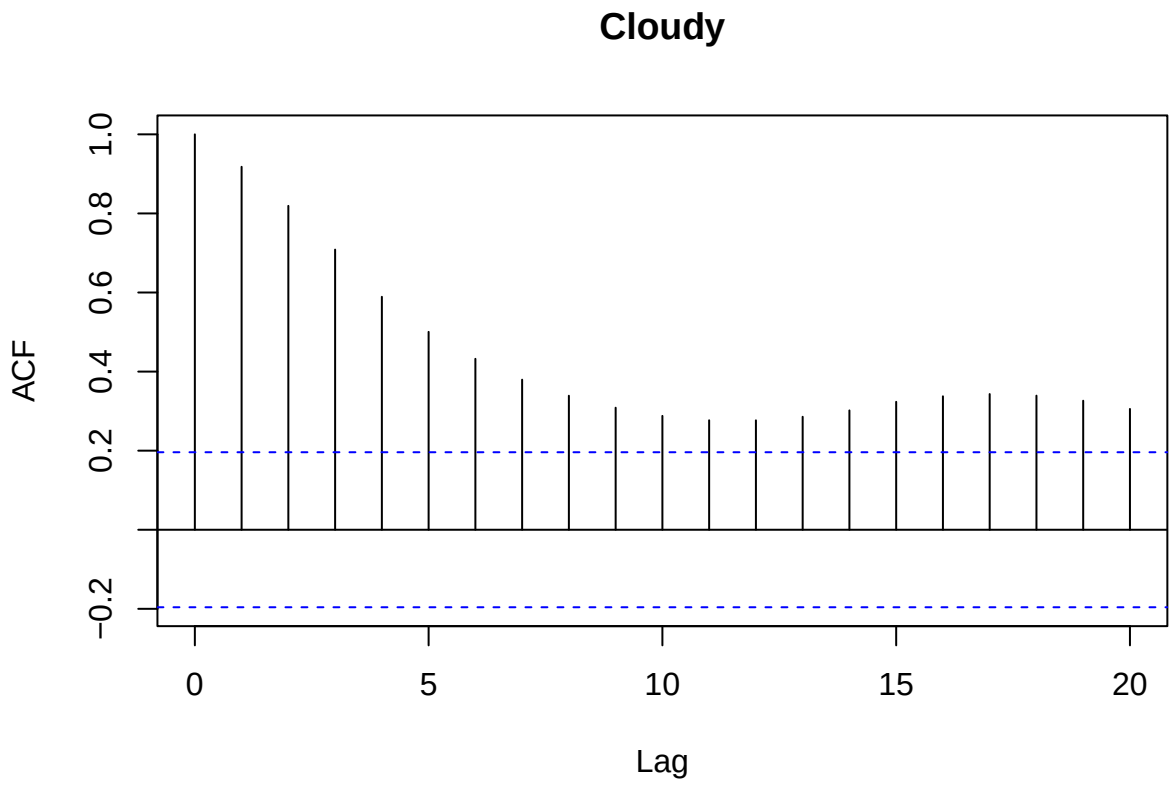
```
RMarg <- mean(gibbSamples[, "Rain"])
cat(sprintf("The margina probability is %s", RMarg))
```

```
## The margina probability is 0.23
```

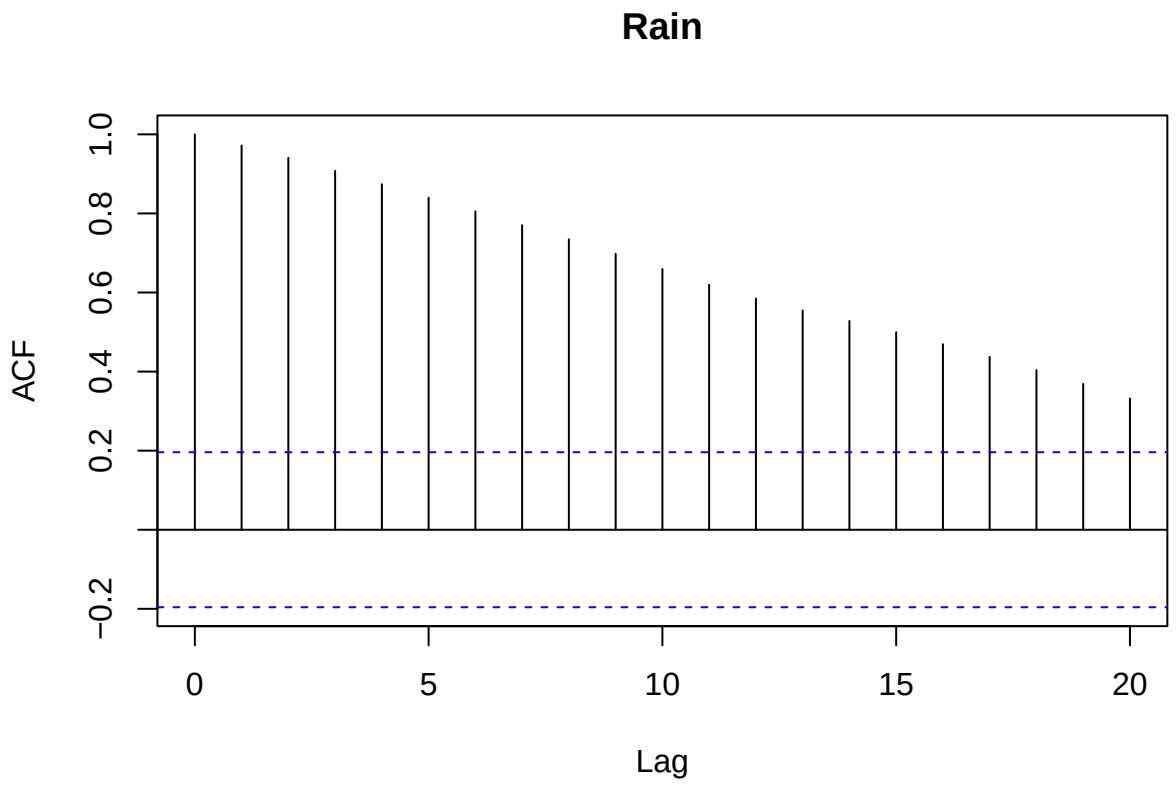
(D) ESS estimation

We first compute the rolling estimate for distribution and subsequently feed that into acf function and return the ESS estimation.

```
## Method 1 defining function --------------------------
rollingEstimate <- function(gibbSamples){
  cloudy <- 1*(gibbSamples[, "Cloudy"])
  rain <- 1*(gibbSamples[, "Rain"])

  rollingEstimateCloudy <- cumsum(cloudy) / seq_along(cloudy)
  rollingEstimateRain <- cumsum(rain) / seq_along(rain)

  return(list(rollingEstimateCloudy,rollingEstimateRain))
}

rollEsts <- rollingEstimate(gibbSamples)

cloudy_acf <- acf(rollEsts[[1]],main = "Cloudy")
```




```
rain_acf   <- acf(rollEsts[[2]], main = "Rain")
```



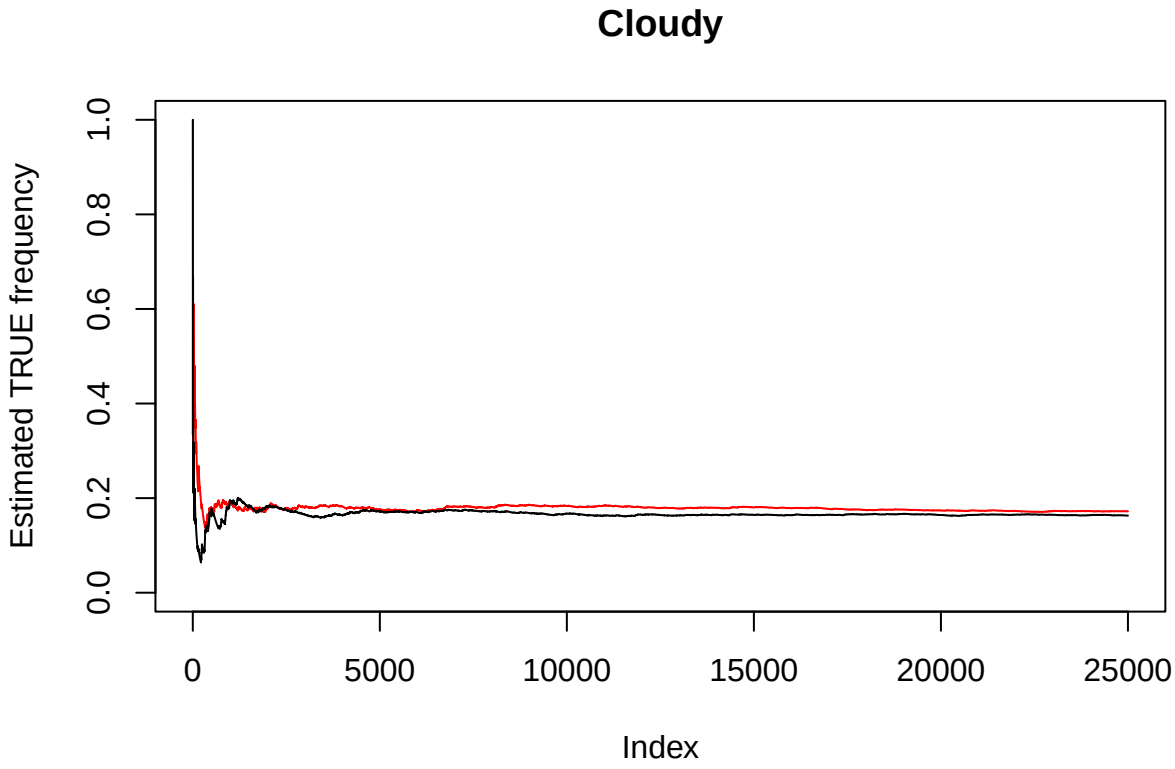

```
## Method 2 data table --------------------------------

gibbSamplesDT <- as.data.table(gibbSamples)
gibbSamplesDT[, RFreq := cumsum(Rain)/seq_len(.N)]
gibbSamplesDT[, CFreq := cumsum(Cloudy)/seq_len(.N)]

cloudyACF <- acf(gibbSamplesDT$CFreq, plot = FALSE)
rainACT   <- acf(gibbSamplesDT$RFreq, plot = FALSE)

## ESS estimation --------------------------------------
ESSCloudy <- 100/(1+2*sum(cloudyACF$acf))
ESSRain   <- 100/(1+2*sum(rainACT$acf))

cat(sprintf("
Cloudy ESS: %s, \n
Rain ESS: %s", ESSCloudy, ESSRain))
```

```
##
## Cloudy ESS: 5.05092202502421,
##
## Rain ESS: 3.4486214161745
```

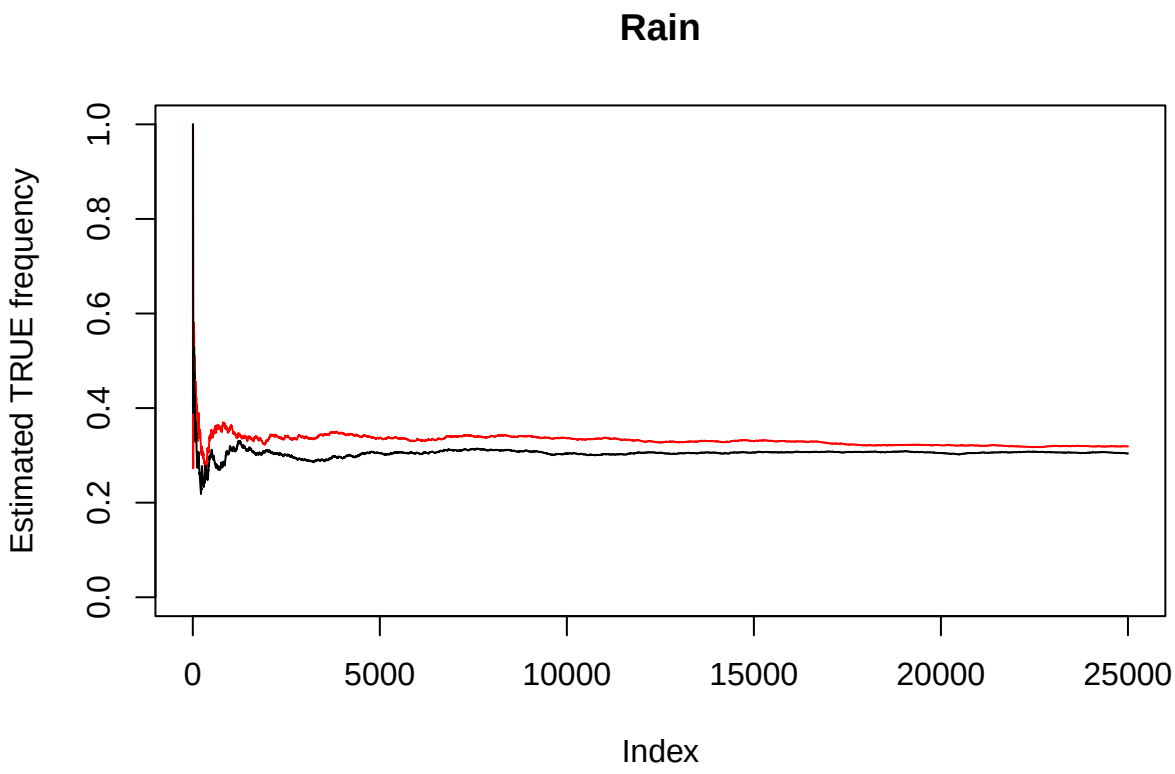
(E & F) Draw 50k sample and run for burn-in time

Looks better for more chains

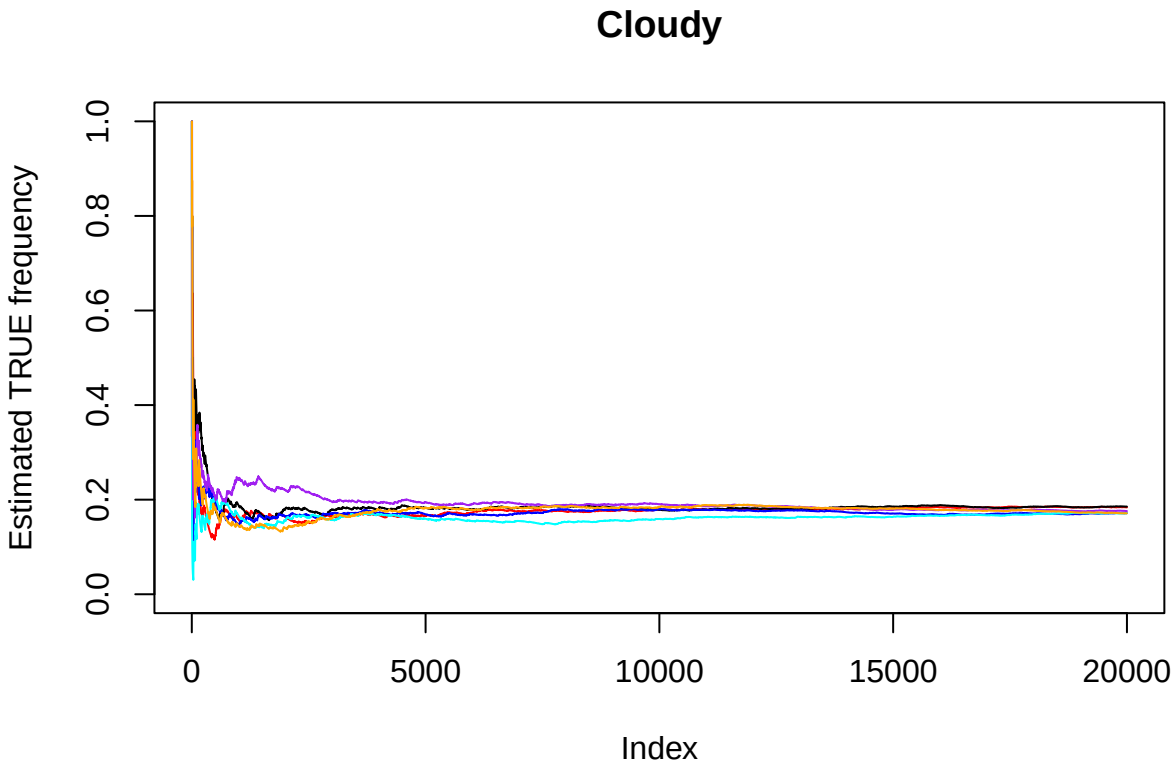
```r
# sample two 50k long chains

gibbSamples_50k_1 <- gibbsRain(50000)
gibbSamples_50k_2 <- gibbsRain(50000)

rollEsts_50k_1 <- rollingEstimate(gibbSamples_50k_1)
rollEsts_50k_2 <- rollingEstimate(gibbSamples_50k_2)

# plot chains
plot(rollEsts_50k_1[[1]][1:25000],type = "l",col="red",ylim = c(0,1), main = "Cloudy",
     ylab = "Estimated TRUE frequency")
lines(rollEsts_50k_2[[1]][1:25000])
```



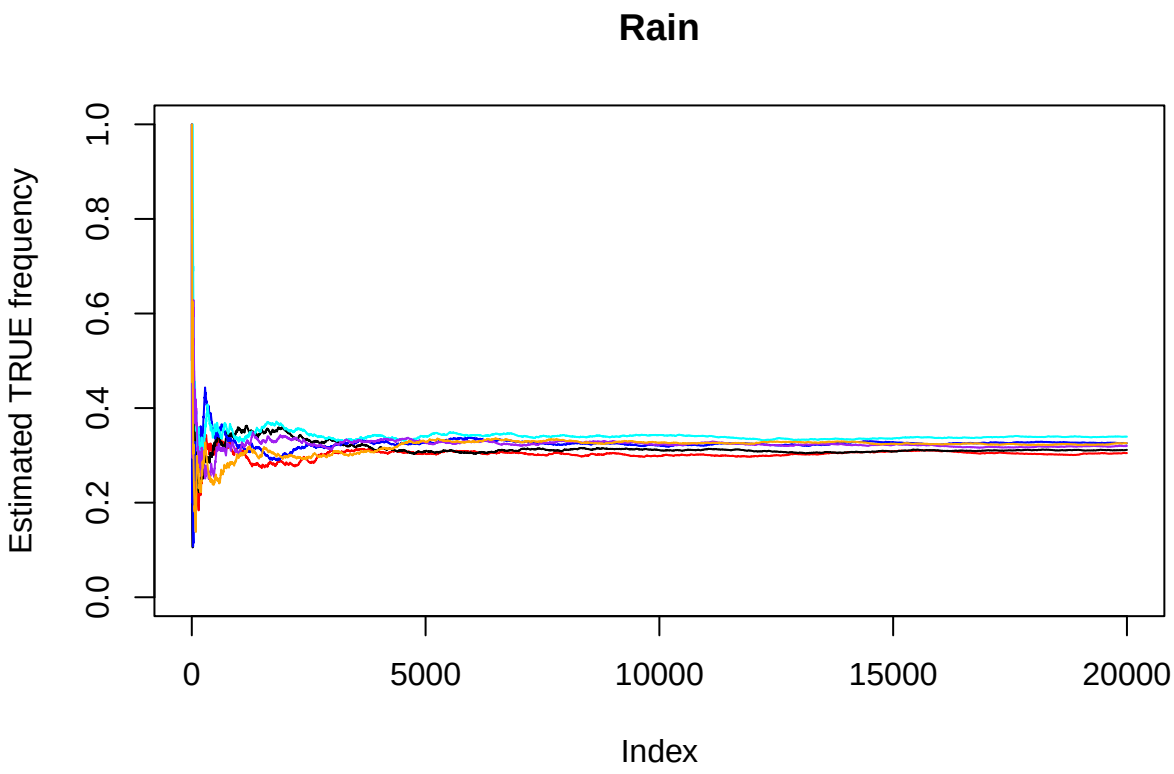

```r
plot(rollEsts_50k_1[[2]][1:25000],type = "l",col="red",ylim = c(0,1), main = "Rain",
     ylab = "Estimated TRUE frequency")
lines(rollEsts_50k_2[[2]][1:25000])
```




```
# plot more then two chains to check on a more general convergence burn-in time

plot(rollingEstimate(gibbsRain(20000))[[1]],type = "l",col="red",ylim = c(0,1),main = "Cloudy",
     ylab = "Estimated TRUE frequency")
for (col in c("black","blue","cyan","purple","orange")) {
  lines(rollingEstimate(gibbsRain(20000))[[1]],col = col)

}
```



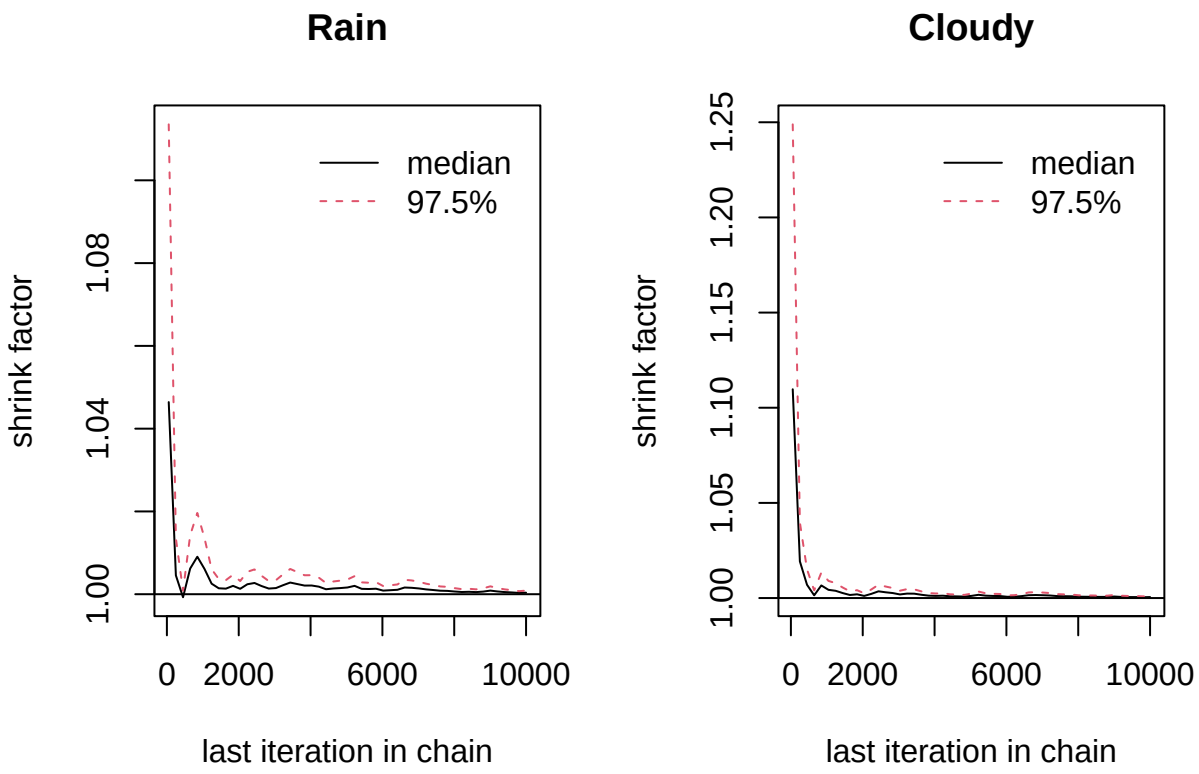

```
plot(rollingEstimate(gibbsRain(20000))[[2]],type = "l",col="red",ylim = c(0,1),main = "Rain",
     ylab = "Estimated TRUE frequency")
for (col in c("black","blue","cyan","purple","orange")) {
  lines(rollingEstimate(gibbsRain(20000))[[2]],col = col)

}
```



From the graphs above, it can be estimated that burn-in time around 8000 be a reasonable choice.

(G) Gelman and Rubin

```
mcmclist <- list()
for (i in 1:10) {
 mcmclist[[i]] <- mcmc(gibbsRain(10000)*1)
}
mcmclist <- mcmc.list(mcmclist)

gelman.plot(mcmclist)
```



From the gelman plot we can estimate that burn-in time around 8000 should be an ideal value.

(H) Re-estimation

To get a better estimate, we sample the distribution from 100 chains and then take the mean of that:

```
vec <- c()
vec2 <- c()
for (i in 1:100) {
  sample <- gibbsRain(8000)
  a <- colMeans(sample)
  vec[i] <- a[1]
  vec2[i] <- a[2]
}

cat(sprintf("
Cloudy prob: %s
Rain prob:  %s", mean(vec), mean(vec2)))
```

```
##
## Cloudy prob: 0.32155125
## Rain prob:  0.1756
```

(H) Analytical value

The analytical value of the marginal probability is

$$P(R|S,W) = \frac{\sum_C P(R=T,C,S=T,W=T)}{\sum_R \sum_C P(R,C,S=T,W=T)} = \frac{0.0396+0.009}{0.0396+0.009+0.0495+0.18} = 0.1747573$$