Project 6 Sampling and variational inference

Team C - Minghang Li, Xiaocheng Yang, Xinyi Chen

April 09, 2024

Problem 15: Monte Carlo estimation of an expected value

Proof that $\mathbb{E}[\hat{g}(\mathbf{X})] = \mathbf{E}[g(X)]$

Proof. It's almost trivial that $\mathbb{E}[g(X)] = \mathbb{E}[g(X_i)]$ (because they are i.i.d from the same probability distribution as X).

$$\begin{aligned}
\mathbb{E}[\hat{g}(\mathbf{X})] &= \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right] \\
&= \frac{1}{N}\mathbb{E}\left[\sum_{i=1}^{N} g(X_i)\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[g(X_i)] \\
&= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[g(X)] \\
&= \frac{N}{N}\mathbb{E}[g(X)] \\
&= \mathbb{E}[g(X)]
\end{aligned}$$

□

Proof that $Var(\hat{g}(\mathbf{X})) = \frac{Var(g(X))}{N}$

Proof. By Bienayme's identity, we know that for pairwise independent variables, we have $Var\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} Var(X_i)$.

And we also have $Var(g(X_i)) = Var(g(X))$ (by the same reason stated in the first subquestion, they are i.i.d with the same variance as $Var(g(X))$).

The proof then follows:

$$
\begin{aligned}
Var(\hat{g}(X)) &= Var\left(\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right) \\
&= \frac{1}{N^2} Var\left(\sum_{i=1}^{N} g(X_i)\right) \\
&= \frac{1}{N^2}\sum_{i=1}^{N} Var(g(X_i)) \\
&= \frac{1}{N^2}\sum_{i=1}^{N} Var(g(X)) \\
&= \frac{N}{N^2} Var(g(X)) \\
&= \frac{Var(g(X))}{N}
\end{aligned}
$$

□

Problem 16: Sampling in the Rain Network

(a) Derive the expressions

All "$= T$"s are grayed out in the derivation for the ease of my brain to interpret.

Recall that given a Bayesian network with nodes in set X, for a certain node x we have $P(x|X_{\setminus x}) = P(x|MB(x))$.

From the structure of Markove Chain we can know that $S \perp R|C$

With the knowledge learned at hand, we can derive the expressions:

(a). Derive $P(C_{=T}|R_{=T}, S_{=T}, W_{=T})$

$$
\begin{aligned}
P(C_{=T}|R_{=T}, S_{=T}, W_{=T}) &= P(C_{=T}|R_{=T}, S_{=T}) && \text{(because } MB(C) = \{R, S\}\text{)} \\
&= \frac{P(R_{=T}, S_{=T}|C_{=T})P(C_{=T})}{P(R_{=T}, S_{=T})} \\
&= \frac{P(R_{=T}|C_{=T})P(S_{=T}|C_{=T})P(C_{=T})}{\sum_C P(R_{=T}|C)P(S_{=T}|C)P(C)} && \text{(because } S \perp R|C\text{)}
\end{aligned}
$$

From the Bayesian network we know that:

$$
\begin{aligned}
P(C_{=T}) &= 0.5 \\
P(R_{=T}|C_{=T}) &= 0.8 \\
P(S_{=T}|C_{=T}) &= 0.1 \\
P(R_{=T}, S_{=T}) &= P(R_{=T}|C_{=T})P(S_{=T}|C_{=T})P(C_{=T}) + P(R_{=T}|C_{=F})P(S_{=T}|C_{=F})P(C_{=F}) \\
&= 0.8 \cdot 0.1 \cdot 0.5 + 0.2 \cdot 0.5 \cdot 0.5 \\
&= 0.09
\end{aligned}
$$

Plug in the values, $P(C_{=T}|R_{=T}, S_{=T}, W_{=T})$ is hence calculated (by the following code blocks) to be $\frac{4}{9}$

```
P_C_T <- 0.5
P_C_F <- 0.5
P_R_T_given_C_T <- 0.8
P_R_T_given_C_F <- 0.2
P_S_T_given_C_T <- 0.1
P_S_T_given_C_F <- 0.5

P_R_T_and_S_T <- P_R_T_given_C_T * P_S_T_given_C_T * P_C_T +
                 P_R_T_given_C_F * P_S_T_given_C_F * P_C_F
```

```
P_C_T_give_R_T_and_S_T <- (P_R_T_given_C_T * P_S_T_given_C_T * P_C_T) / P_R_T_and_S_T
P_C_T_give_R_T_and_S_T
```

```
## [1] 0.4444444
```

2. Derive $P(C_{=T}|R_{=F}, S_{=T}, W_{=T})$

$$
\begin{aligned}
P(C_{=T}|R_{=F}, S_{=T}, W_{=T}) &= P(C_{=T}|R_{=F}, S_{=T}) && \text{(because } MB(C) = \{R, S\}\text{)} \\
&= \frac{P(R_{=F}, S_{=T}|C_{=T})P(C_{=T})}{P(R_{=F}, S_{=T})} \\
&= \frac{P(R_{=F}|C_{=T})P(S_{=T}|C_{=T})P(C_{=T})}{\sum_C P(R_{=F}|C)P(S_{=T}|C)P(C)} && \text{(because } S \perp R|C\text{)}
\end{aligned}
$$

We know that S and R are binary variables that only take `True` or `False`, i.e., $P(R_{=T}) + P(R_{=F}) = 1$ and $P(S_{=T}) + P(S_{=F}) = 1$.

Plug in the values, $P(C_{=T}|R_{=F}, S_{=T}, W_{=T})$ is hence calculated (by the following code blocks) to be $\frac{1}{21}$

```
P_R_F_given_C_T <- 1 - P_R_T_given_C_T
P_R_F_given_C_F <- 1 - P_R_F_given_C_T
P_R_F_and_S_T <- P_R_F_given_C_T * P_S_T_given_C_T * P_C_T +
                 P_R_F_given_C_F * P_S_T_given_C_F * P_C_F
```

```
P_C_T_give_R_F_and_S_T <- (P_R_F_given_C_T * P_S_T_given_C_T * P_C_T) / P_R_F_and_S_T
P_C_T_give_R_F_and_S_T
```

```
## [1] 0.04761905
```

3. Derive $P(R_{=T}|C_{=T}, S_{=T}, W_{=T})$

In the following derivation the entities that are canceled out inside the condition probability are either because $S \perp R|C$ or $P(W|X_{\setminus W}) = P(W|MB(W))$.

$$
\begin{aligned}
P(R_{=T}|C_{=T},S_{=T},W_{=T}) &= \frac{P(R_{=T},C_{=T},S_{=T},W_{=T})}{P(C_{=T},S_{=T},W_{=T})} \\
&= \frac{P(W_{=T}|R_{=T},C_{=T},S_{=T})P(R_{=T},C_{=T},S_{=T})}{P(W_{=T}|C_{=T},S_{=T})P(C_{=T},S_{=T})} \\
&= \frac{P(W_{=T}|R_{=T},\cancel{C_{=T}},S_{=T})P(R_{=T}|C_{=T},\cancel{S_{=T}})\cancel{P(C_{=T},S_{=T})}}{P(W_{=T}|C_{=T},S_{=T})\cancel{P(C_{=T},S_{=T})}} \\
&= \frac{P(W_{=T}|R_{=T},S_{=T})P(R_{=T}|C_{=T})}{\sum_R P(W_{=T},R|C_{=T},S_{=T})} \\
&= \frac{P(W_{=T}|R_{=T},S_{=T})P(R_{=T}|C_{=T})}{\sum_R P(W_{=T}|R,\cancel{C_{=T}},S_{=T})P(R|C_{=T},\cancel{S_{=T}})} \qquad \text{(Trying to make use of } P(W|R,S)\text{)} \\
&= \frac{P(W_{=T}|R_{=T},S_{=T})P(R_{=T}|C_{=T})}{\sum_R P(W_{=T}|R,S_{=T})P(R|C_{=T})}
\end{aligned}
$$

From the Bayesian network we know that

$$
\begin{aligned}
P(W_{=T}|R_{=T},S_{=T}) &= 0.99 \\
P(R_{=T}|C_{=T}) &= 0.8 \\
\sum_R P(W_{=T}|R,S_{=T})P(R|C_{=T}) &= P(W_{=T}|R_{=T},S_{=T})P(R_{=T}|C_{=T}) + P(W_{=T}|R_{=F},S_{=T})P(R_{=F}|C_{=T}) \\
&= 0.99 \cdot 0.8 + 0.9 \cdot (1-0.8) \\
&= 0.972
\end{aligned}
$$

Plug in the values, we can find that $P(R_{=T}|C_{=T},S_{=T},W_{=T}) = \frac{22}{27}$

```
P_W_T_given_R_T_and_S_T <- 0.99
P_W_T_given_R_F_and_S_T <- 0.9
P_W_T_given_R_T_and_S_F <- 0.9
P_W_T_given_R_F_and_S_F <- 0.1
```

```
P_W_T_given_S_T_and_C_T <- P_W_T_given_R_T_and_S_T * P_R_T_given_C_T +
                           P_W_T_given_R_F_and_S_T * P_R_F_given_C_T
```

```
P_R_T_given_C_T_and_S_T_and_W_T <- (P_W_T_given_R_T_and_S_T * P_R_T_given_C_T) /
                                   P_W_T_given_S_T_and_C_T
P_R_T_given_C_T_and_S_T_and_W_T
```

```
## [1] 0.8148148
```

4. Derive $P(R_{=T}|C_{=F},S_{=T},W_{=T})$

Following similar derivation as $P(R_{=T}|C_{=T},S_{=T},W_{=T})$

$$
\begin{aligned}
P(R_{=T}|C_{=F},S_{=T},W_{=T}) & \\
&= \frac{P(W_{=T}|R_{=T},S_{=T})P(R_{=T}|C_{=F})}{\sum_R P(W_{=T}|R,S_{=T})P(R|C_{=F})} \\
&= \frac{0.99 \cdot 0.2}{0.99 \cdot 0.2 + 0.9 \cdot (1-0.2)} \\
&= \frac{11}{51}
\end{aligned}
$$

```
P_R_T_given_C_F_and_S_T_and_W_T <- (P_W_T_given_R_T_and_S_T * P_R_T_given_C_F) /
                                  (P_W_T_given_R_T_and_S_T * P_R_T_given_C_F +
                                   P_W_T_given_R_F_and_S_T * P_R_F_given_C_F)
P_R_T_given_C_F_and_S_T_and_W_T
```

```
## [1] 0.2156863
```

(b). Implement the Gibbs sampler for the Bayesian network

```
sample_CgivenR <- function(R){
  C <- 0
  if (R == F) {
    C <- rbern(1, p=P_C_T_give_R_F_and_S_T)
  } else {
    C <- rbern(1, p=P_C_T_give_R_T_and_S_T)
  }
  return(C)
}

sample_RgivenC <- function(C){
  R <- 0
  if (C == F) {
    R <- rbern(1, p=P_R_T_given_C_F_and_S_T_and_W_T)
  } else {
    R <- rbern(1, p=P_R_T_given_C_T_and_S_T_and_W_T)
  }
  return(R)
}
```

```
set.seed(42)
```

```
gibbs_sampler <- function(niter) {
  C <- rep(0,niter)
  R <- rep(0,niter)
  C[1]=1
  R[1]=1
  foreach (i=2:niter) %do% {
    C[i] <- sample_CgivenR(R[i-1])
    R[i] <- sample_RgivenC(C[i])
  }
  res <- data.frame(C=C,R=R)
  return(res)
}
```

```
samples <- gibbs_sampler(100)
res <- table(samples) / 100
res
```

```
##    R
## C      0    1
##   0 0.61 0.20
##   1 0.02 0.17
```

(c). Estimate the marginal probability of rain ($P(R_{=T}|S_{=T}, W_{=T})$)

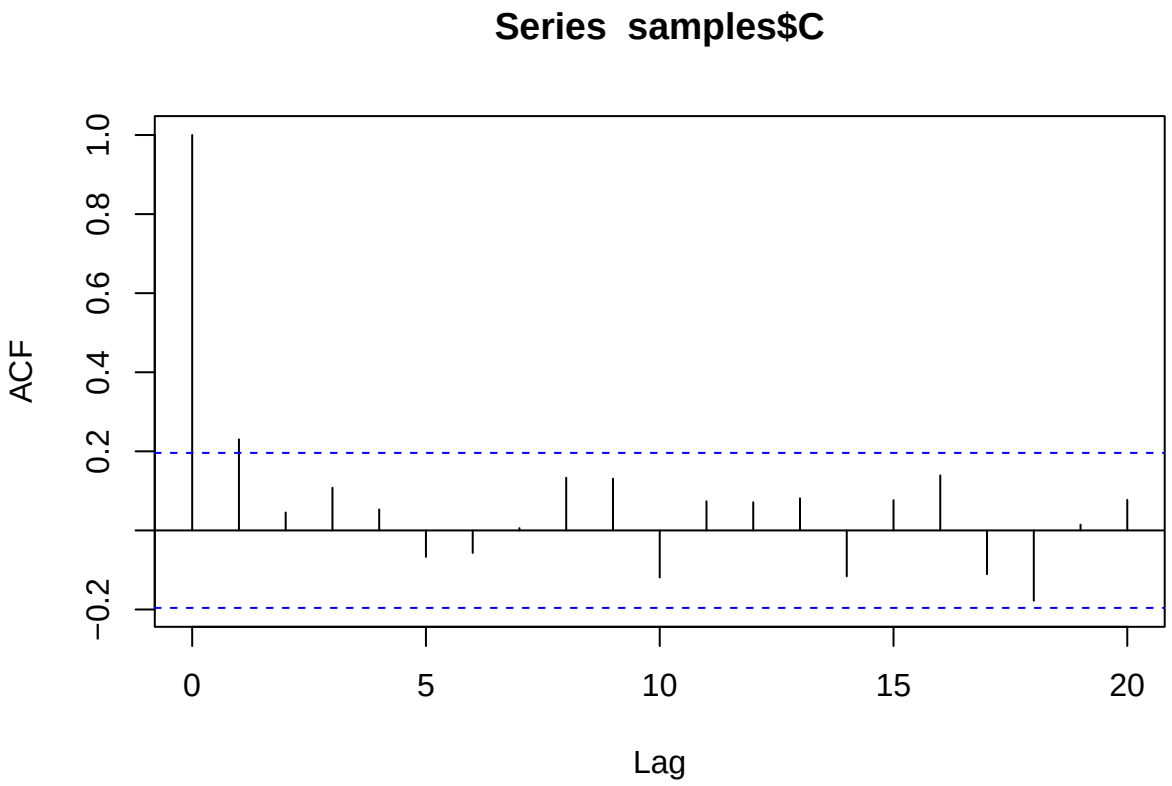
Just sum over the $R = T$ column of the result table:

```
P_R_T_given_S_T_and_W_T <- colSums(res)[[2]]
P_R_T_given_S_T_and_W_T
```

```
## [1] 0.37
```

(d). Use the R function `acf()` to provide plots and estimates of the auto-correlation functions for the samples of both variables Rain and Cloudy. Provide estimates of the ESS.

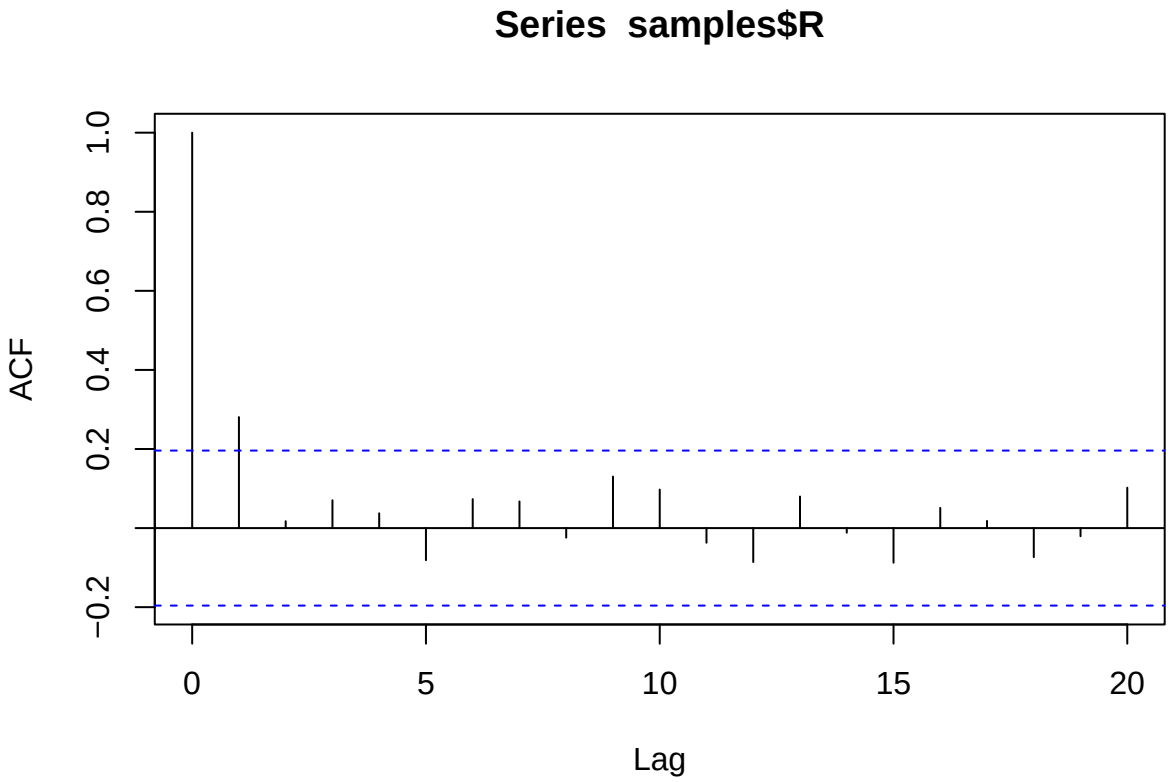
```
acf_C <- acf(samples$C)
```




```
ESS_C <- acf_C$n.used / (1 + 2 *sum(acf_C$acf))
ESS_C
```

```
## [1] 23.89455
```

```
acf_R <- acf(samples$R)
```




```
ESS_R <- acf_R$n.used / (1 + 2 *sum(acf_R$acf))
ESS_R
```

```
## [1] 23.7731
```

(e) Draw 50,000 samples

```
samples_50k_run1 <- gibbs_sampler(50000)
res_50k_run1 <- table(samples_50k_run1) / 50000
res_50k_run1
```

```
##    R
## C         0       1
##   0 0.64286 0.18140
##   1 0.03272 0.14302
```

```
samples_50k_run2 <- gibbs_sampler(50000)
res_50k_run2 <- table(samples_50k_run2) / 50000
res_50k_run2
```

```
##    R
## C         0       1
##   0 0.64556 0.17990
##   1 0.03204 0.14250
```

(f) Plot the relative frequenceis of $R = T$ and $C = T$ up to each iteration t against t, for two independent runs of the sampler

```r
calc_relative_freq <- function(samples) {
  n <- length(x = samples)
  rel_freq <- foreach(t=1:n, .combine = c) %dopar% {
    return(sum(samples[1:t]/t))
  }
  return(rel_freq)
}
```

```r
num_cores <- detectCores()
registerDoParallel(num_cores)
start_time <- Sys.time()
rel_freq_C_run1 <- calc_relative_freq(samples = samples_50k_run1$C)
rel_freq_C_run2 <- calc_relative_freq(samples = samples_50k_run2$C)
rel_freq_R_run1 <- calc_relative_freq(samples = samples_50k_run1$R)
rel_freq_R_run2 <- calc_relative_freq(samples = samples_50k_run2$R)
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 52.79544 secs
```

```r
stopImplicitCluster()
```

Plot the lines

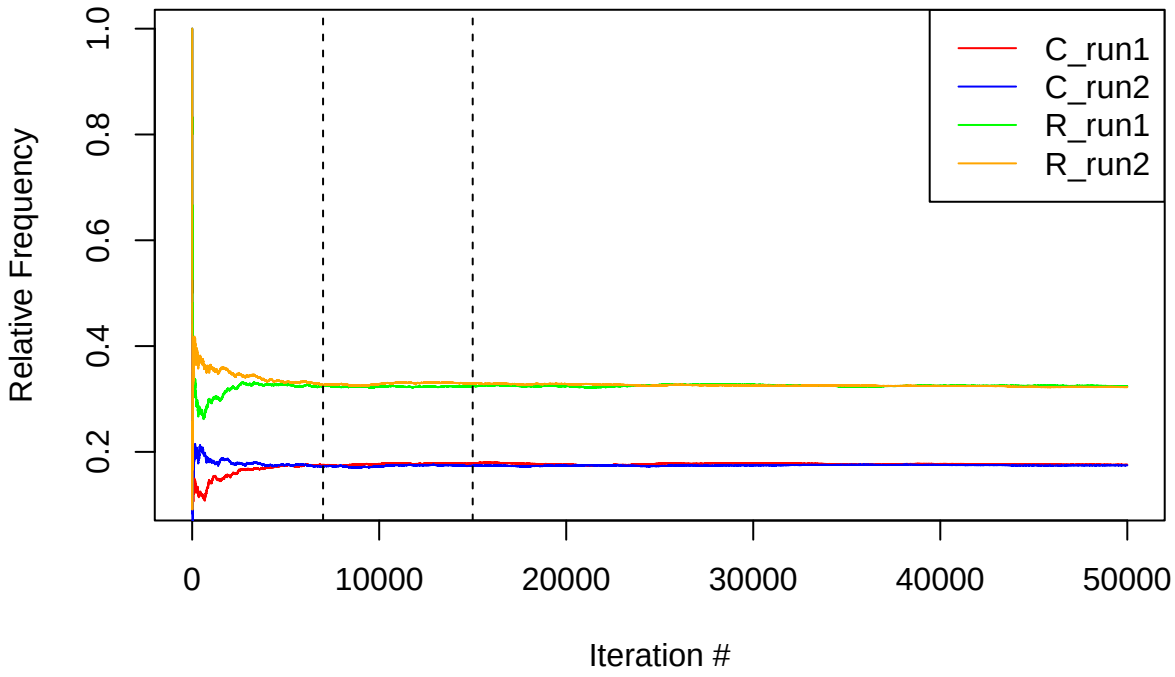
```r
x=1:50000
plot(x, rel_freq_C_run1, col = "red",
     type="l",
     xlab = "Iteration #",
     ylab = "Relative Frequency",
)
lines(x, rel_freq_C_run2, col = "blue")

# Add the third vector, rel_freq_R_run1, to the plot
lines(x, rel_freq_R_run1, col = "green")

# Add the fourth vector, rel_freq_R_run2, to the plot
lines(x, rel_freq_R_run2, col = "orange")

# Add a legend to the plot
legend("topright",
       legend = c("C_run1", "C_run2", "R_run1", "R_run2"),
       col = c("red", "blue", "green", "orange"),
       lty = 1)

# Add vertical lines at x = 7000 and x = 15000
abline(v = c(7000, 15000), lty = 2, col = "black")
```



It seems that the relative frequencies start to plateau at around 7000 iterations. Hence it's sufficient to do about 7000 iterations.

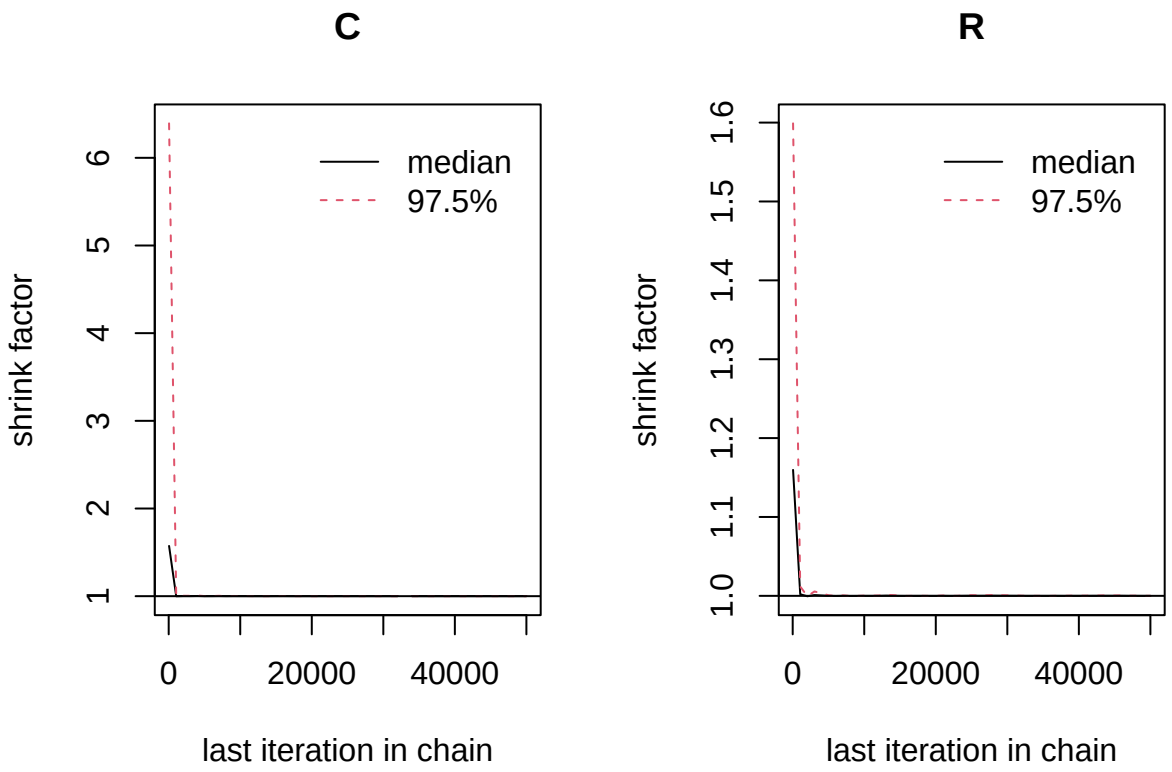
(And if you really want to be confident, maybe 15,000 iterations)

(g) Apply the *Gelman and Rubin* test and plot the potential scale reduction factor changes over the iterations using `gelman.plot()`.

```
mcmcs<- mcmc.list(mcmc(data = samples_50k_run1, start = 1, end = 5e4),
                  mcmc(data = samples_50k_run2, start = 1, end = 5e4))
gelman.diag(mcmcs)
```

```
## Potential scale reduction factors:
##
##   Point est. Upper C.I.
## C          1          1
## R          1          1
##
## Multivariate psrf
##
## 1
```

```
gelman.plot(mcmcs)
```



It's clear that after about 7000 iterations there are almost no fluctuations in the shrink factors and they go very close to 1.0. It would be safe to set a brun-in time of about 7000, or to be safer, 10000.

(h) Re-estimate $P(R = T|S = T, W = T)$ based on samples obtained after the burn-in time.

```
samples_cut <- samples_50k_run1[1e4 + 1:5e4, ]
```

```
res_cut <- table(samples_cut) / 4e4
res_cut
```

```
##    R
## C          0        1
##   0 0.642425 0.182200
##   1 0.032800 0.142575
```

```
P_R_T_given_S_T_and_W_T_reestimated <- colSums(res_cut)[[2]]
P_R_T_given_S_T_and_W_T_reestimated
```

```
## [1] 0.324775
```

(i) Compute the probability analytically and compare with (c) and (h)

Estimation from (c)

The value computed in (c) is: 0.37.

The value computed in (h) is: 0.324775.

Analytical compuation

$$
\begin{aligned}
P(R_{=T}|S_{=T}, W_{=T}) &= \frac{P(R_{=T}, S_{=T}, W_{=T})}{P(S_{=T}, W_{=T})} \\
&= \frac{\sum_C P(R_{=T}, S_{=T}, W_{=T}, C)}{\sum_C P(S_{=T}, W_{=T}, C)}
\end{aligned}
$$

The numerator part:

$$
\begin{aligned}
\sum_C P(R_{=T}, S_{=T}, W_{=T}, C) &= \sum_C P(R_{=T}, S_{=T}, W_{=T}|C)P(C) \\
&= \sum_C P(W_{=T}|R_{=T}, S_{=T}, \not{\varnothing})P(R_{=T}, S_{=T}|C)P(C) \\
&= \sum_C P(W_{=T}|R_{=T}, S_{=T})P(R_{=T}|C)P(S_{=T}|C)P(C) \\
&= P(W_{=T}|R_{=T}, S_{=T})P(R_{=T}|C_{=T})P(S_{=T}|C_{=T})P(C_{=T}) \\
&\quad + P(W_{=T}|R_{=T}, S_{=T})P(R_{=T}|C_{=F})P(S_{=T}|C_{=F})P(C_{=F}) \\
&= 0.99 \cdot 0.8 \cdot 0.1 \cdot 0.5 + 0.99 \cdot 0.2 \cdot 0.5 \cdot 0.5 \\
&= 0.0891
\end{aligned}
$$

```
numerator <- P_W_T_given_R_T_and_S_T * P_R_T_given_C_T * P_S_T_given_C_T * P_C_T +
  P_W_T_given_R_T_and_S_T * P_R_T_given_C_F * P_S_T_given_C_F * P_C_F
numerator
```

```
## [1] 0.0891
```

The denominator:

$$
\begin{aligned}
\sum_C P(S_{=T}, W_{=T}, C) &= \sum_C \sum_R P(W_{=T}, S_{=T}, R, C) \\
&= \sum_C \sum_R P(W_{=T}|R, S_{=T}, \not{\varnothing})P(R, S_{=T}, C) \\
&= \sum_C \sum_R P(W_{=T}|R, S_{=T})P(R|C)P(S_{=T}|C)P(C) \\
&= P(W_{=T}|R_{=T}, S_{=T})P(R_{=T}|C_{=T})P(S_{=T}|C_{=T})P(C_{=T}) \\
&\quad + P(W_{=T}|R_{=F}, S_{=T})P(R_{=F}|C_{=T})P(S_{=T}|C_{=T})P(C_{=T}) \\
&\quad + P(W_{=T}|R_{=T}, S_{=T})P(R_{=T}|C_{=F})P(S_{=T}|C_{=F})P(C_{=F}) \\
&\quad + P(W_{=T}|R_{=F}, S_{=T})P(R_{=F}|C_{=F})P(S_{=T}|C_{=F})P(C_{=F}) \\
&= 0.99 \cdot 0.8 \cdot 0.1 \cdot 0.5 + 0.9 \cdot (1 - 0.8) \cdot 0.1 \cdot 0.5 \\
&\quad + 0.99 \cdot 0.2 \cdot 0.5 \cdot 0.5 + 0.9 \cdot (1 - 0.2) \cdot 0.5 \cdot 0.5 \\
&= 0.2781
\end{aligned}
$$

```
denominator <- P_W_T_given_R_T_and_S_T * P_R_T_given_C_T * P_S_T_given_C_T * P_C_T +
  P_W_T_given_R_F_and_S_T * P_R_F_given_C_T * P_S_T_given_C_T * P_C_T +
  P_W_T_given_R_T_and_S_T * P_R_T_given_C_F * P_S_T_given_C_F * P_C_F +
  P_W_T_given_R_F_and_S_T * P_R_F_given_C_F * P_S_T_given_C_F * P_C_F
denominator
```

```
## [1] 0.2781
```

Plug back the values of the denominator and numerator, we have:

$$P(R_{=T}|S_{=T}, W_{=T}) = \frac{0.0891}{0.2781} \approx 0.3204$$

```
P_R_T_given_S_T_and_W_T_analytic <- numerator / denominator
P_R_T_given_S_T_and_W_T_analytic
```

```
## [1] 0.3203883
```

```
# Create a data frame with the three probabilities
probs <- data.frame(
  Probability = c("Estimation in (c)", "Re-estimation in (h)", "Analytic solution"),
  Value = c(P_R_T_given_S_T_and_W_T,
            P_R_T_given_S_T_and_W_T_reestimated,
            P_R_T_given_S_T_and_W_T_analytic)
)

# Use knitr::kable() to format the data frame as a table
knitr::kable(probs, align = "l", caption = "Probabilities", row.names = FALSE)
```

Table 1: Probabilities

Probability	Value
Estimation in (c)	0.3700000
Re-estimation in (h)	0.3247750
Analytic solution	0.3203883

The value obtained in (h) seems to be very close to the analytical solution. The value obtained in (c) is more different from the analytical solution but is expected since we only sampled 100 times.