Problem 15: Monte Carlo estimation of an expected value

Show that $\mathbb{E}(\hat{g}(\mathbf{X})) = \mathbb{E}(g(X))$:

$\mathbb{E}(\hat{g}(\mathbf{X})) = \mathbb{E}(\frac{1}{N}\sum_{i=1}^{N} g(X_i))$

$= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}(g(X_i))$, by linearity of expectation. Since all X_i are drawn from the distribution X,

$\mathbb{E}(g(X_i)) = \mathbb{E}(g(X))$

So,

$= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}(g(X)) = \frac{1}{N} \cdot N\, \mathbb{E}(g(X))$

$= \mathbb{E}(g(X))$

Show that $\mathrm{Var}(\hat{g}(\mathbf{X})) = \frac{\mathrm{Var}(g(X))}{n}$

$\mathrm{Var}(\hat{g}(\mathbf{X})) = \mathrm{Var}(\frac{1}{N}\sum_{i=1}^{N} g(X_i))$

$= \frac{1}{N^2}\mathrm{Var}(\sum_{i=1}^{N} g(X_i))$, since $1/N$ is constant. By Bienaymé's identity:

$= \frac{1}{N^2}\mathrm{Var}(\sum_{i=1}^{N} g(X_i)) = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}(g(X_i))$

$= \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}(g(X)) = \frac{1}{N^2}\cdot N \cdot \mathrm{Var}(g(X))$

$= \mathrm{Var}(g(X))/N$

(a)

$$P(C=T \mid R=T, S=T, W=T)$$

$$= P(C=T \mid S=T, R=T), \text{ by Markov blanket}$$

$$= \frac{P(C=T)\,P(S=T, R=T \mid C=T)}{P(S=T, R=T)}$$

$$= \frac{P(C=T)\,P(S=T \mid C=T)\,P(R=T \mid C=T)}{P(C=T)\,P(S=T \mid C=T)\,P(R=T \mid C=T) + P(C=F)\,P(S=T \mid C=F)\,P(R=T \mid C=F)} \quad (*)$$

Plug in values.

$$= \frac{0.5 \cdot 0.1 \cdot 0.8}{0.5 \cdot 0.1 \cdot 0.8 + 0.5 \cdot 0.5 \cdot 0.2}$$

$$= 0.4444\ldots$$

For $P(C=T \mid R=F, S=T, W=T)$ is completly analogus. Take $(*)$ and replace $R=T$ with $R=F$:

$$P(C=T \mid R=F, S=T, W=T)$$

$$= \frac{P(C=T)\, P(S=T \mid C=T)\, P(R=F \mid C=T)}{P(C=T)\, P(S=T \mid C=T)\, P(R=F) + P(C=F)\, P(S=T \mid C=F)\, P(R=F \mid C=F)}$$

$$= \frac{0.5 \cdot 0.1 \cdot 0.2}{0.5 \cdot 0.1 \cdot 0.2 + 0.5 \cdot 0.5 \cdot 0.8}$$

$$\approx 0.04762$$

We now compute

$$P(R=T \mid C=T, S=T = W=T)$$

$$= \frac{P(R=T \mid C=T, S=T)\, P(W=T \mid R=T, S=T)}{P(W=T \mid C=T, S=T)}$$

$$= \frac{P(R=T \mid C=T)\, P(W=T \mid R=T, S=T)}{P(R=T \mid C=T)\, P(W=T \mid R=T, S=T) + P(R=F \mid C=T)\, P(W=T \mid R=F, S=T)} \quad (**)$$

Plug in values:

$$= \frac{0.8 \cdot 0.99}{0.8 \cdot 0.99 + 0.2 \cdot 0.9} \approx 0.8148$$

Analogously, we can find

$P(R=T \mid C=F, S=T, W=T)$ by replacing

$C=T$ with $C=F$ in $(**)$

$$= \frac{P(R=T \mid C=F)\, P(W=T \mid R=T, S=T)}{P(R=T \mid C=F)\, P(W=T \mid R=T, S=T) + P(R=F \mid C=F)\, P(W=T \mid R=F, S=T)}$$

$$= \frac{0.2 \cdot 0.99}{0.2 \cdot 0.99 + 0.8 \cdot 0.9} \approx 0.2157$$

Project 6 - Problem 16 (b) - (i)

Group J

2023-04-05

Subtask (b)

The probabilities derived in (a) are used to implement a Gibbs sampler for the network:

```
n <- 100
retMat <- matrix(ncol = 2, nrow = n)

#We define a matrix, where sampling a 0 or 1 corresponds to C or R respectively.
#The matrix contains the needed conditional probabilities to perform Gibbs sampling

probs <- matrix(data = c((.5*.1*.2/(.5*.1*.2 + .5*.5*.8)),
                         (.2*.99)/(.2*.99+.8*.9),
                         (.5*.1*.8)/(.5*.1*.8 + .5*.5*.2),
                         (.8*.99)/(.8*.99+.2*.9)), nrow = 2, ncol = 2)

#Define x_0 as the initial state of the Markov chain
x_0 <- c(1,1)
x <- x_0

#Uniformly sample whether to update C or R

#Get initial index
choice <- rbinom(1, 1, .5)
idx <- choice + 1

for (i in 1:n) {

  sampling_prob <- probs[idx, (!choice) + 1]
  sample <- rbinom(1, 1, sampling_prob)

  retMat[i, ] <- x
  x[idx] <- sample

  idx <- ifelse(idx == 2, yes = 1, no = 2)

}

colnames(retMat) <- c("C", "R")
head(retMat)
```

```
##      C R
```

```
## [1,] 1 1
## [2,] 1 0
## [3,] 0 0
## [4,] 0 0
## [5,] 0 0
## [6,] 0 0
```

Subtask (c)

This allows us to calculate the marginal probability $P(R = T|S = T, W = T)$:

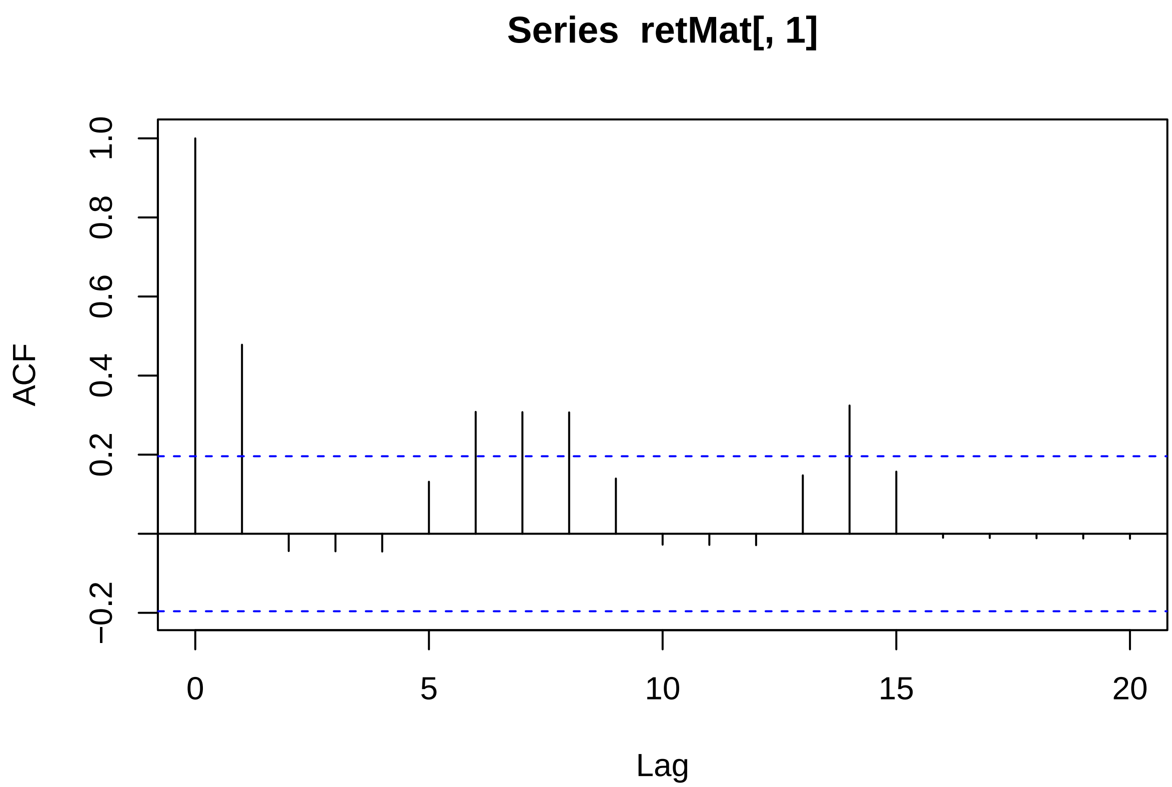
```
counts <- matrix(data = c(0, 0, 0, 0),
                 nrow = 2, ncol = 2)

for (i in 1:n) {
  temp <- retMat[i, ]
  temp <- temp + 1
  counts[temp[1], temp[2]] <- counts[temp[1], temp[2]] + 1
}

# Calculate probability:
(task_c <- rowSums(counts/n)[2])
```

```
## [1] 0.06
```

Subtask (d)

Here we use the effective sample size (ESS) to estimate the autocorrelation in the data. Further, the **acf**-function plots the autocorrelation of each sample with its neighboring samples.

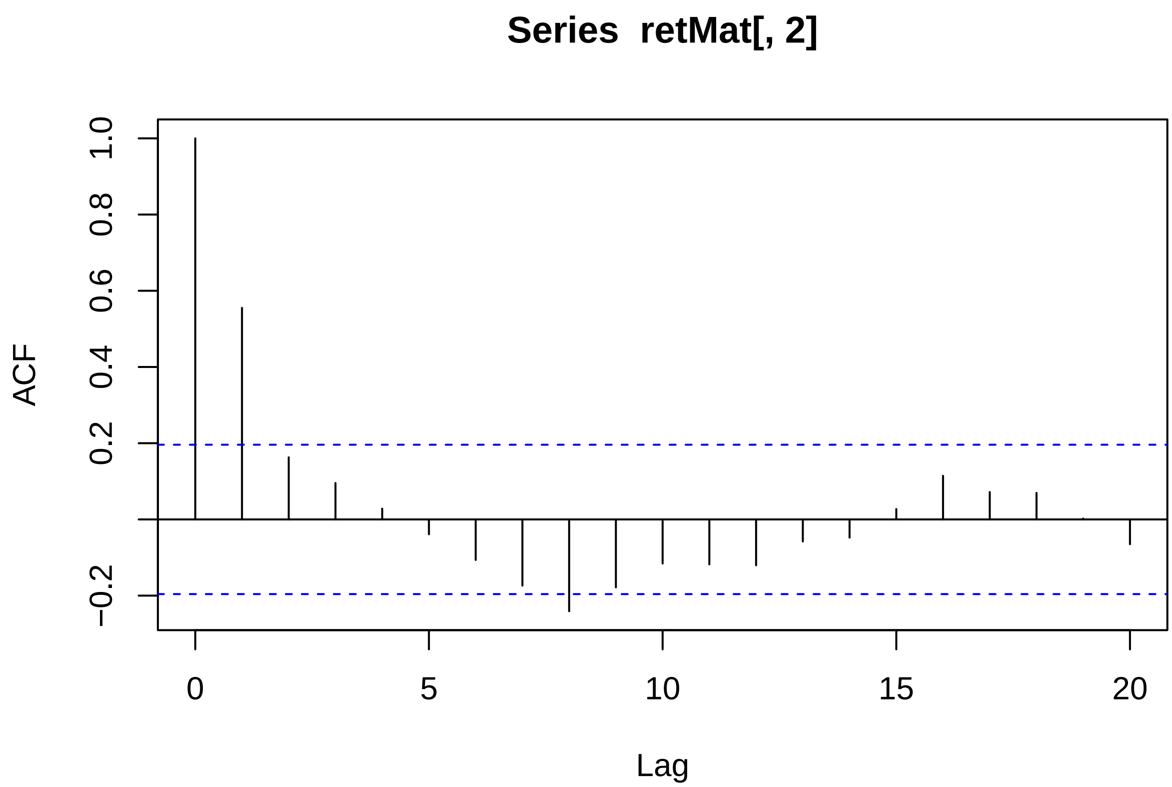
```
# We use the upper limit for k that is selected by the acf-function
acf1 <- acf(retMat[, 1])
```




```
n/(1 + 2 *sum(acf1$acf))
```

```
## [1] 14.18226
```

```
acf2 <- acf(retMat[, 2])
```




```
n/(1 + 2 * sum(acf2$acf))
```

```
## [1] 36.65333
```

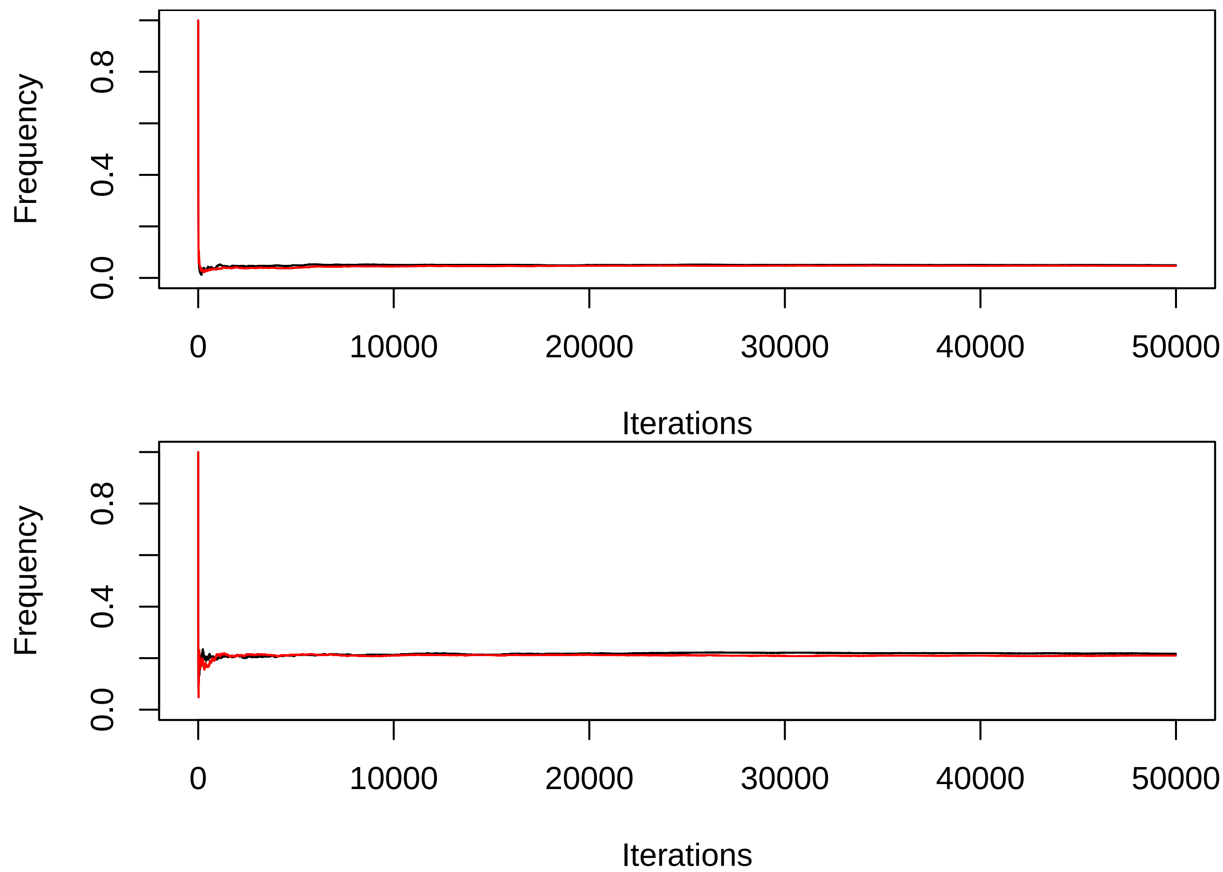
Subtask (e)

```
sim_1 <- matrix(ncol = 2, nrow = 50000)
sim_2 <- matrix(ncol = 2, nrow = 50000)

#We define a matrix, where sampling a 0 or 1 corresponds to C or R respectively.
#The matrix contains the needed conditional probabilities to perform Gibbs sampling

probs <- matrix(data = c((.5*.1*.2/(.5*.1*.2 + .5*.5*.8)),
                         (.2*.99)/(.2*.99+.8*.9),
                         (.5*.1*.8)/(.5*.1*.8 + .5*.5*.2),
                         (.8*.99)/(.8*.99+.2*.9)), nrow = 2, ncol = 2)

#Define x_0 as the initial state of the Markov chain
x_0 <- c(1,1)
x <- x_0

#Uniformly sample whether to update C or R
idx <- choice + 1
```

```r
for (i in 1:50000) {

  sampling_prob <- probs[idx, (!choice) + 1]
  sample <- rbinom(1, 1, sampling_prob)

  sim_1[i, ] <- x
  x[idx] <- sample

  idx <- ifelse(idx == 2, yes = 1, no = 2)
}

x_0 <- c(1,1)
x <- x_0

#Uniformly sample whether to update C or R
idx <- choice + 1

for (i in 1:50000) {

  sampling_prob <- probs[idx, (!choice) + 1]
  sample <- rbinom(1, 1, sampling_prob)

  sim_2[i, ] <- x
  x[idx] <- sample

  idx <- ifelse(idx == 2, yes = 1, no = 2)
}
```

Subtask (f)

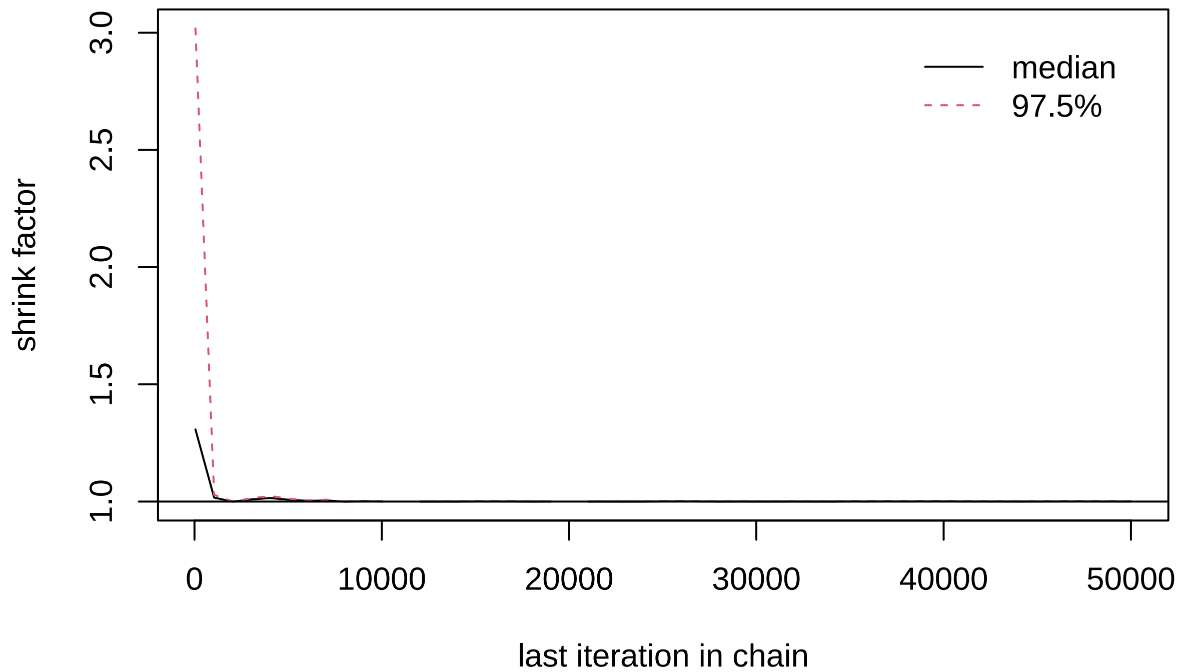
```r
par(mfrow=c(2,1), mar = c(4,4,0,1))

plot(cumsum(sim_1[, 1])/c(1:50000),
     type = 'l', ylim = c(0, 1),
     ylab = "Frequency",
     xlab = "Iterations")
lines(cumsum(sim_2[, 1])/c(1:50000), col = "red")

plot(cumsum(sim_1[, 2])/c(1:50000),
     type = 'l', ylim = c(0, 1),
     ylab = "Frequency",
     xlab = "Iterations")
lines(cumsum(sim_2[, 2])/c(1:50000), col = "red")
```



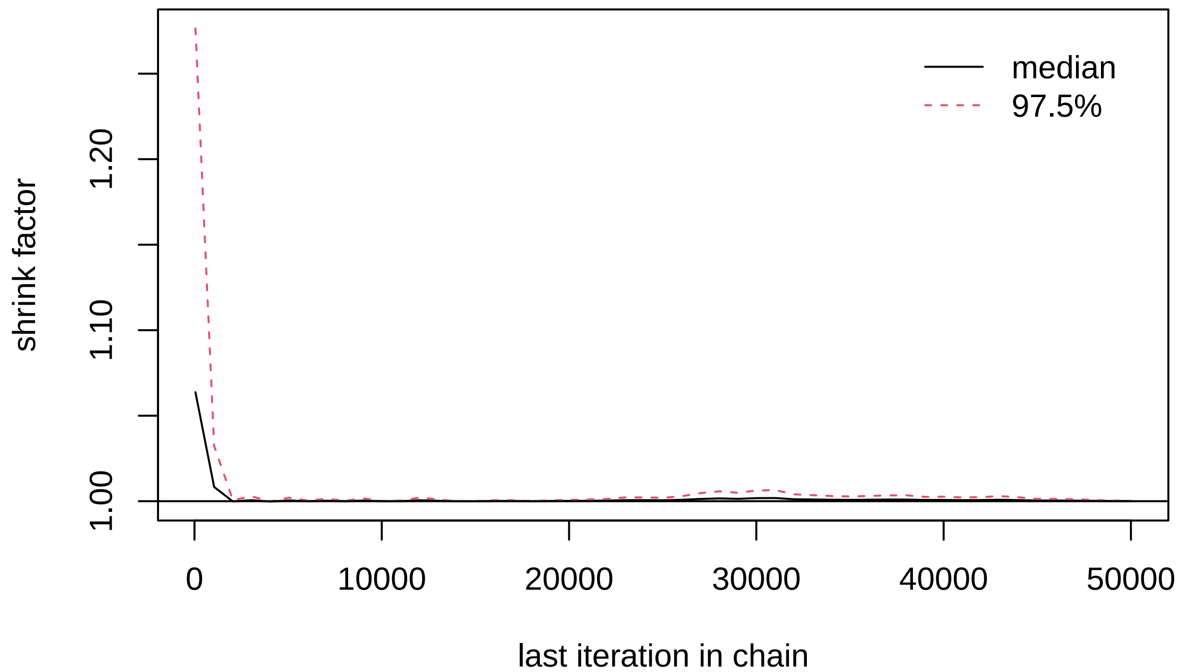
Based on the plots, we suggest a burn-in time of maximally a few thousand iterations. We pick 10000, as it is clear that this point is far past the burn-in phase, but implementing a threshold might allow for a more exact estimation.

Subtask (g)

```
gelman.plot(mcmc.list(mcmc(sim_1[, 1]), mcmc(sim_2[, 1])))
```




```
gelman.plot(mcmc.list(mcmc(sim_1[, 2]), mcmc(sim_2[, 2])))
```



We can see that the shrink factor converges to 1 fast after less than a few thousand iterations. This is in accordance with the result obtained in subtask (f).

Subtask (h)

This allows us to calculate the marginal probability $P(R = T|S = T, W = T)$:

```
# For simulation 1:
counts <- matrix(data = c(0, 0, 0, 0),
                 nrow = 2, ncol = 2)

for (i in 1:50000) {
  temp <- sim_1[i, ]
  temp <- temp + 1
  counts[temp[1], temp[2]] <- counts[temp[1], temp[2]] + 1
}

# Calculate probability:
(task_h_1 <- rowSums(counts/50000)[2])
```

```
## [1] 0.04936
```

```
# For simulation 2:
counts <- matrix(data = c(0, 0, 0, 0),
                 nrow = 2, ncol = 2)
```

```
for (i in 1:50000) {
  temp <- sim_2[i, ]
  temp <- temp + 1
  counts[temp[1], temp[2]] <- counts[temp[1], temp[2]] + 1
}

# Calculate probability:
(task_h_2 <- rowSums(counts/50000)[2])
```

```
## [1] 0.04628
```

Subtask (i)

Given the answers from subtask (a), the analytical solution returns:

$$
\begin{aligned}
P(R=T|S=T,W=T) &= \frac{P(R=T,S=T,W=T)}{P(S=T,W=T)} \\
&= \frac{\sum_C P(C,R=T,S=T,W=T)}{\sum_{C,R} P(C,R,S=T,W=T)} \\
&= \frac{\sum_C P(C)P(R=T|C)P(S=T|C)P(W=T|S,R)}{\sum_{C,R} P(C)P(R|C)P(S=T|C)P(W=T|S,R)}
\end{aligned}
$$

This returns:

```
# COmpute analytical solution
# Compare to:

task_c
```

```
## [1] 0.06
```

```
task_h_1
```

```
## [1] 0.04936
```

```
task_h_2
```

```
## [1] 0.04628
```