Project 5

Richard Affolter

03 April, 2021

Problem 15: Monte Carlo estimation of an expected value

Show that $\mathbb{E}[\hat{g}(X)] = \mathbb{E}[g(X)]$ **and** $\mathrm{Var}[\hat{g}(X)] = \frac{\mathrm{Var}[g(X)]}{N}$

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

We know that N is a constant and that the expected value operator is linear for the sum of random variables. This means we can rearrange as follows:

$$\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right] = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[g(X_i)]$$

Since our samples $X_1 ... X_N$ follow the same distribution of X we have a common expected value of $g(X_i)$. So

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

Therefore

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

For the variance we have

$$\mathrm{Var}[\hat{g}(X)] = \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right]$$

We know from the basic properties of the variance that

$$\mathrm{Var}[aX] = a^2 \mathrm{Var}[X]$$

and that

$$\mathrm{Var}[g(X_1) + g(X_2)] = \mathrm{Var}[g(X_1)] + \mathrm{Var}[g(X_2)] + 2\mathrm{Cov}[g(X_1), g(X_2)]$$

Since our $g(X_i)$ are independent of each other our covariance is zero.

Therefore

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

In a similar argument to above we find that since our samples $X_1 ... X_N$ follow the same distribution of X we have a common variance of $g(X_i)$ So

$$\mathrm{Var}[g(X_i)] = \mathrm{E}[g(X)])$$

Therefore

$$\begin{aligned}\frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}[g(X_i)] &= \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}[g(X)] \\ &= \frac{1}{N^2}N\cdot\mathrm{Var}[g(X)] \\ &= \frac{\mathrm{Var}[g(X)]}{N}\end{aligned}$$

Problem 16(data analysis): Sampling in the Rain Network

(a) Derive the expressions for $P(C = T \mid R = T, S = T, W = T)$, $P(C = T \mid R = F, S = T, W = T)$, $P(R = T \mid C = T, S = T, W = T)$ and $P(R = T \mid C = F, S = T, W = T)$ and compute their values.

In general we have $P(C, R, S, W) = P(C) \cdot P(R \mid C) \cdot P(S \mid C)P(W \mid S, R)$

$$\begin{aligned}P(C = T \mid R = T, S = T, W = T) &= \frac{P(C = T, R = T, S = T, W = T)}{\sum_C P(C \mid R = T, S = T, W = T)} \\ &= \frac{0.5\cdot 0.1\cdot 0.8\cdot 0.99}{0.5\cdot 0.1\cdot 0.8\cdot 0.99 + 0.5\cdot 0.5\cdot 0.2\cdot 0.99} \\ &= 0.\bar{4}\end{aligned}$$

$$\begin{aligned}P(C = T \mid R = F, S = T, W = T) &= \frac{P(C = T, R = F, S = T, W = T)}{\sum_C P(C \mid R = F, S = T, W = T)} \\ &= \frac{0.5\cdot 0.1\cdot 0.2\cdot 0.9}{0.5\cdot 0.1\cdot 0.2\cdot 0.9 + 0.5\cdot 0.5\cdot 0.8\cdot 0.9} \\ &= 0.048\end{aligned}$$

$$\begin{aligned}P(R = T \mid C = T, S = T, W = T) &= \frac{P(R = T, C = T, S = T, W = T)}{\sum_R P(R, C = T, S = T, W = T)} \\ &= \frac{0.5\cdot 0.1\cdot 0.8\cdot 0.99}{0.5\cdot 0.1\cdot 0.8\cdot 0.99 + 0.5\cdot 0.1\cdot 0.2\cdot 0.9} \\ &= 0.815\end{aligned}$$

$$\begin{aligned}P(R = T \mid C = F, S = T, W = T) &= \frac{P(R = T, C = F, S = T, W = T)}{\sum_R P(R, C = F, S = T, W = T)} \\ &= \frac{0.5\cdot 0.5\cdot 0.2\cdot 0.99}{0.5\cdot 0.5\cdot 0.2\cdot 0.99 + 0.5\cdot 0.5\cdot 0.8\cdot 0.9} \\ &= 0.216\end{aligned}$$

(b) Implement the Gibbs sampler for the Bayesian network in Figure 1 and draw 100 samples fromthe joint posterior probability distribution $P(R, C \mid S = T, W = T)$.

```r
# determine if we enter/stay in the TRUE state with the given probability
get_boolean = function(prob){
  return(prob >= runif(1))
}

# get the appropriate transition probabilities from (a)
get_transition_prob = function(update_Cloudy_else_Rain, Cloudy, Rain){
  # if update_Cloudy_else_Rain == TRUE we update Cloudy, else we update Rain
  transition_prob =
  if(update_Cloudy_else_Rain){
    if(Rain) 0.444    # this corresponds to P(C=T| R=T,S=T,W=T)
    else 0.048        # this corresponds to P(C=T| R=F,S=T,W=T)
  } else{
    if(Cloudy) 0.815  # this corresponds to P(R=T| C=T,S=T,W=T)
    else 0.216        # this corresponds to P(R=T| C=F,S=T,W=T)
  }
  return(transition_prob)
}

# initialize a vector with TRUE at the beginning
# (first pos. does not matter since we discard the burn-in)
initialize_vec = function(n){
  vec = rep(NA,n)
  vec[1] = TRUE
  return(vec)
}

Gibbs_sampler = function(n){
  # initialize
  Cloudy_vec = initialize_vec(n)
  Rain_vec = initialize_vec(n)

  # Gibbs sampling
  for(i in 2:n){
    # randomly decide what to update
    update_Cloudy_else_Rain = runif(1)>= 0.5

    transition_prob = get_transition_prob(update_Cloudy_else_Rain,
                                          Cloudy = Cloudy_vec[i-1],
                                          Rain = Rain_vec[i-1])

    if(update_Cloudy_else_Rain){
      # if we update Cloudy than Rain stays unchanged
      Rain_vec[i] = Rain_vec[i-1]
      Cloudy_vec[i] = get_boolean(transition_prob)
    } else{
      # if we update Rain than Cloudy stays unchanged
      Cloudy_vec[i] = Cloudy_vec[i-1]
      Rain_vec[i] = get_boolean(transition_prob)
    }
  }
  return(list(Cloudy = Cloudy_vec, Rain = Rain_vec))
}
```

```r
set.seed(123)
result = Gibbs_sampler(100)
Rain = result$Rain
Cloudy = result$Cloudy
```

(c) Estimate the marginal probability of rain, given that the sprinkler is on and the grass is wet $P(R = T \mid S = T, W = T)$ from the 100 samples.

```r
cat(" the estimated marginal probability is : ", mean(Rain))
```

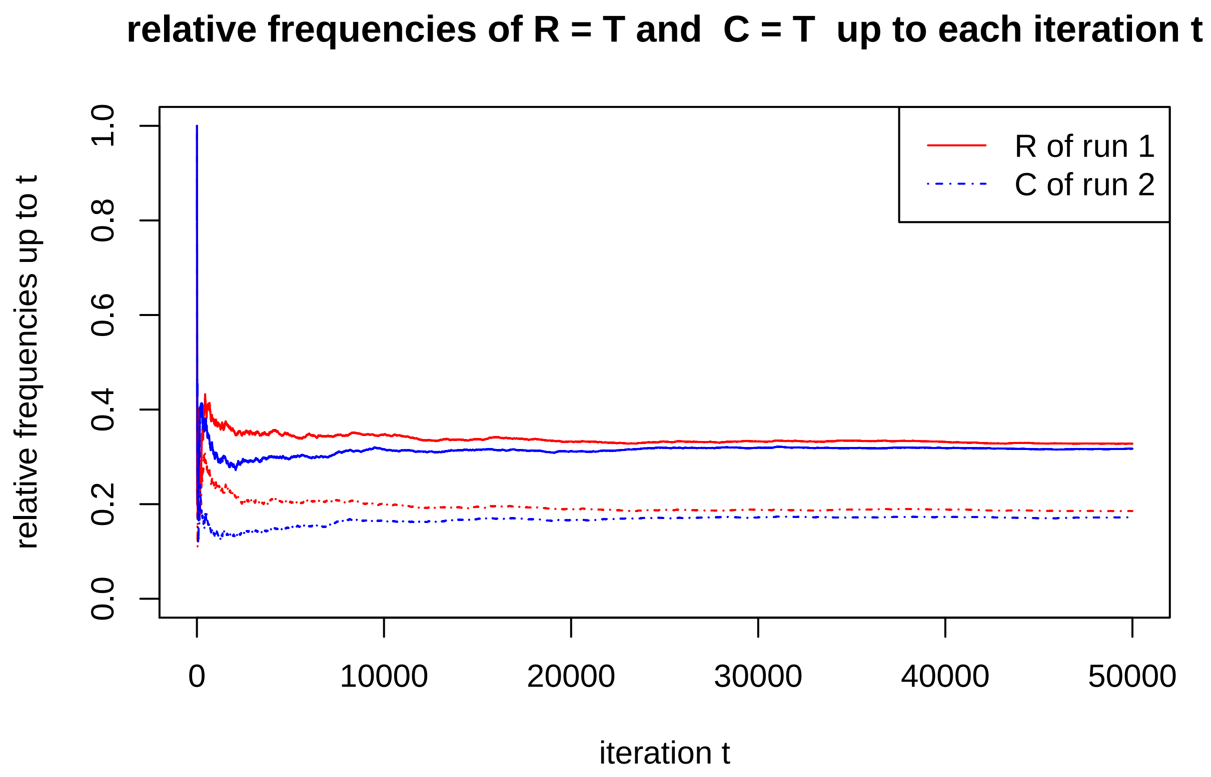
```
##  the estimated marginal probability is :  0.29
```

(d) Draw 50,000 samples instead of 100 using the Gibbs sampler.

```r
set.seed(123)
result = Gibbs_sampler(50000)
Rain = result$Rain
Cloudy = result$Cloudy
```

(e) Plot the relative frequencies of $R = T$ and $C = T$ up to each iteration t against t, for two independent runs of the sampler. Suggest a burn-in time based on this plot.

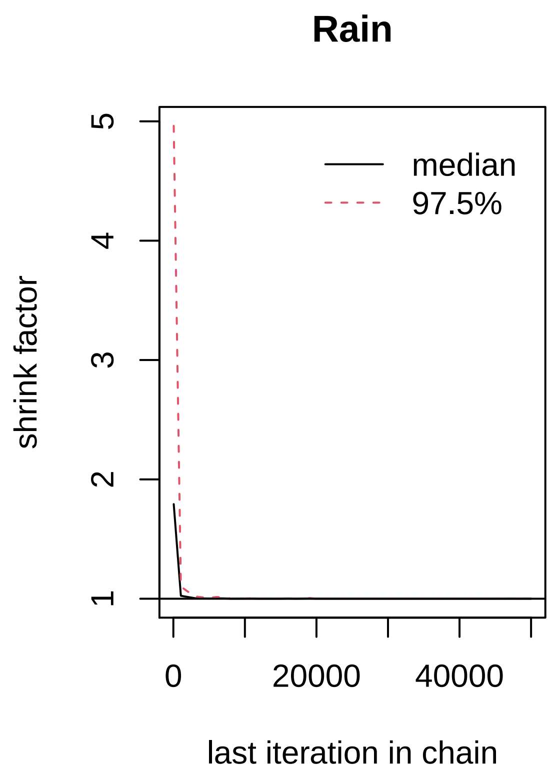
```r
# get the second independent run
result2 = Gibbs_sampler(50000)

# function for calculating the relative frequencies
get_rel_freq = function(x){
  return(cumsum(x)/1:length(x))
}

R1 = get_rel_freq(result$Rain)
C1 = get_rel_freq(result$Cloudy)

R2 = get_rel_freq(result2$Rain)
C2 = get_rel_freq(result2$Cloudy)

plot(R1, type = "l", lty = "solid",col = "red", ylim = c(0,1),
     main = "relative frequencies of R = T and  C = T  up to each iteration t",
     xlab = "iteration t", ylab = "relative frequencies up to t")
lines(C1, lty = "dotdash", col = "red")
lines(R2, lty = "solid", col = "blue")
lines(C2, lty = "dotdash", col = "blue")
legend("topright", legend = c("R of run 1", "C of run 2"), lty = c("solid", "dotdash"),
       col = c("red", "blue"))
```

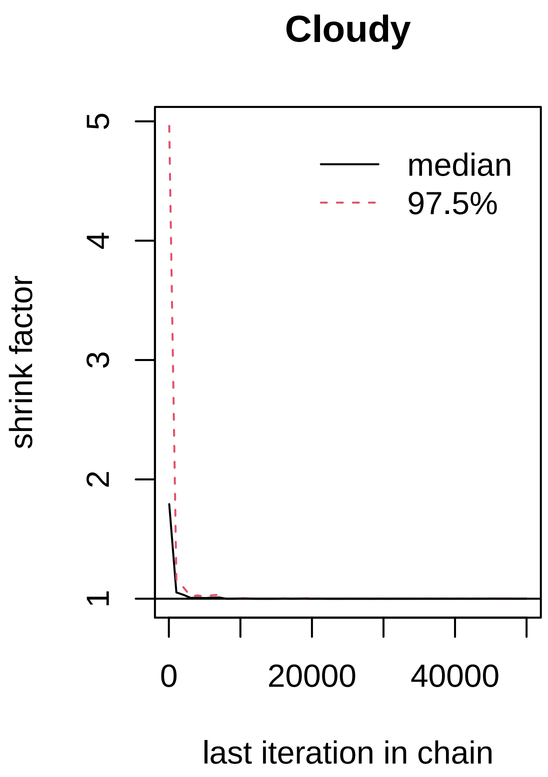


Based on this plot I would suggest a burn-in time of ~10000 since after this time the curves stay relatively flat, there is no drastic change in the relative frequencies any more and the gap between the two runs stays approximately constant.

(f) Apply the Gelman and Rubin test and plot the potential scale reduction factor changes over the iterations using `gelman.plot()` from the `coda` package. This factor measures the ratio of the variances within and between independent runs of the sampler and should be close to 1.0 for stationary distributions. Suggest a burn-in time based on this plot.

```
library(coda)
x1 = mcmc(cbind(Rain = result$Rain, Cloudy = result$Cloudy))
x2 = mcmc(cbind(Rain = result2$Rain, Cloudy = result2$Cloudy))
x = mcmc.list(x1,x2)
gelman.plot(x)
```

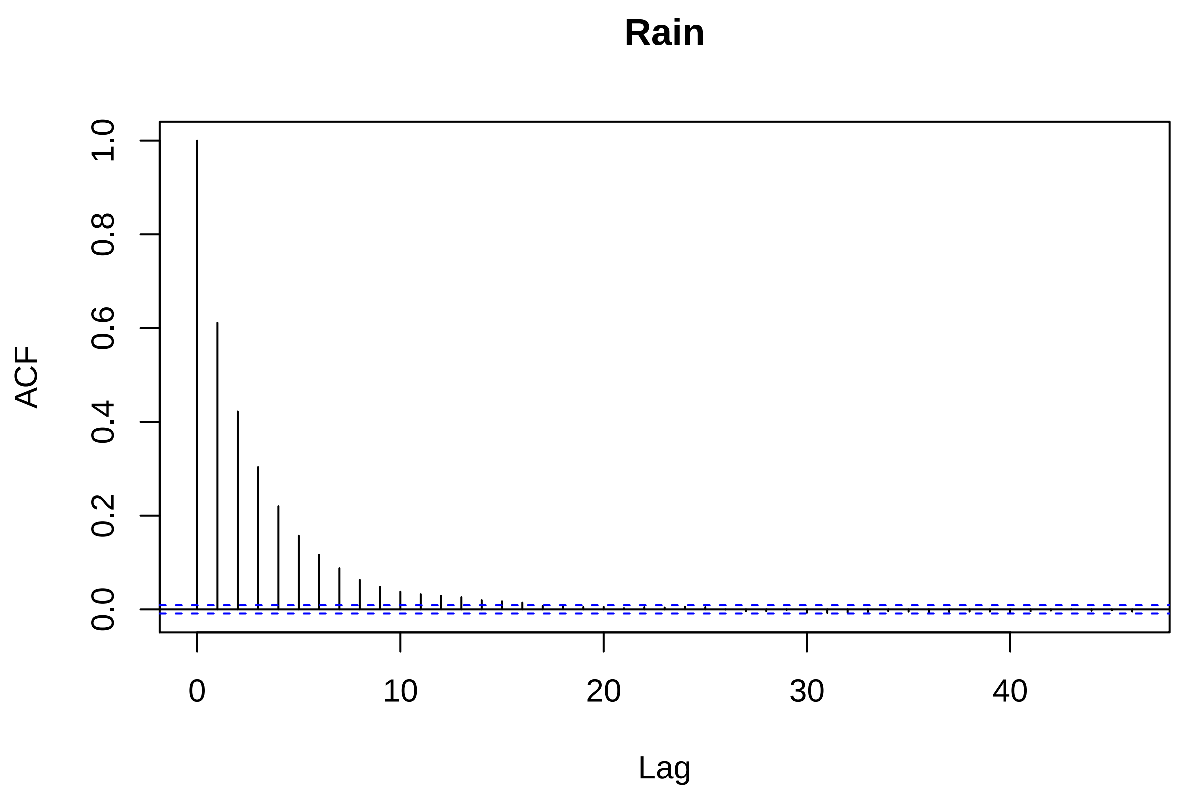




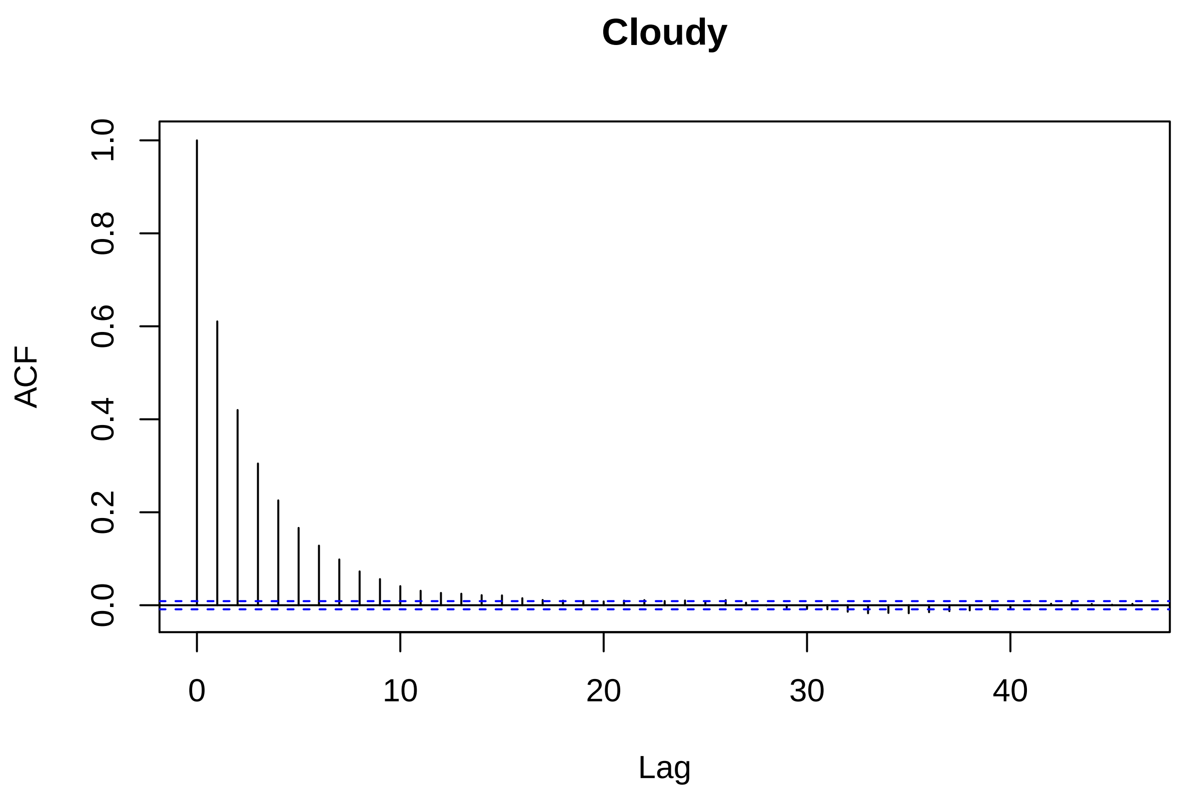
Based on the plots I would suggest a burn-in time of 10000 since after this iteration the shrink factor is very close to 1 for both variables *Rain* and *Cloudy*.

(g) We expect adjacent members from a Gibbs sampling sequence to be positively correlated,and we can quantify this with the auto-correlation function. The lag-k auto-correlation ρ_k is the correlation between every draw and its kth neighbouring sample. Use the R function `acf()` to provide plots for both variables *Rain* and *Cloudy*. Suggest an interval for drawing approximately independent samples.

```r
library(dplyr)
result2$Rain %>% as.numeric() %>% acf(main = "Rain")
```



```
result2$Cloudy %>% as.numeric() %>% acf(main = "Cloudy")
```



Based on the plots we can see that for both variables the autocorrelation ρ_k is for $k > 15$ is not significantly different from zero. Therefore we can use a thinning factor of 16.

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

```
thin = 16
interval_vec = c(rep(FALSE, thin-1),TRUE)
burn_in = 10000
result$Rain[-c(1:burn_in)][interval_vec] %>% as.numeric() %>% mean() -> estimate
cat("the estimated P(R = T | S = T, W = T) using a burn-in time of: ", burn_in,
    "\n and a thinning interval of: ", thin, " is: ", estimate)
```

```
## the estimated P(R = T | S = T, W = T) using a burn-in time of:  10000
##  and a thinning interval of:  16  is:  0.322
```

(i) Compute the probability $P(R = T \mid S = T, W = T)$ analytically. Compare with (c) and (h)and comment on your results.

$$\begin{aligned}
P(R = T \mid S = T, W = T) &= \frac{\sum\limits_{C} P(C, R = T, S = T, W = T)}{\sum\limits_{R}\sum\limits_{C} P(C, R, S = T, W = T)} \\
&= \frac{0.5 \cdot 0.1 \cdot 0.8 \cdot 0.99 + 0.5 \cdot 0.5 \cdot 0.2 \cdot 0.99}{0.5 \cdot 0.1 \cdot 0.8 \cdot 0.99 + 0.5 \cdot 0.5 \cdot 0.2 \cdot 0.99 + 0.5 \cdot 0.1 \cdot 0.2 \cdot 0.9 + 0.5 \cdot 0.5 \cdot 0.8 \cdot 0.9} \\
&= 0.320
\end{aligned}$$

If we compare the analytical result of 0.32 with the result of (c) being 0.29 we see that it is already quite close, even though the chain did not reach the stationary distribution yet. If we use a much higher sample size and use burn-in and thinning we get the result of (g) being 0.322 which is very close to the analytical one, even though the effective sample size was 2500 samples out of 50000.