Exercise 1

TMA4300 Computer Intensive Statistical Models

Mads Adrian Simonsen, William Scott Grundeland Olsen

01 februar, 2021

Problem A: Stochastic simulation by the probability integral transform and bivariate techniques

1.

2.

(a)

(b)

3.

(a)

(b)

(c)

4.

5

Problem B: The gamma distribution

1.

(a)

(b)

2.

(a)

(b)

3.

(a)

(b)

4.

5.

(a)

(b)

Problem C: Monte Carlo integration and variance reduction

1.

2.

3.

(a)

(b)

Problem D: Rejection sampling and importance sampling

Subproblem 1.

We consider a vector of multinomially distributed counts

$$\begin{bmatrix} y_1 \end{bmatrix} \quad \begin{bmatrix} \frac{1}{2}+\frac{\theta}{4} \end{bmatrix}$$

and the observed data is $\mathbf{y} = \begin{bmatrix}125 & 18 & 20 & 34\end{bmatrix}^\top$. The multinomial mass function is given as

$$f(\mathbf{y} \mid \theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_3},$$

and assuming a prior that is Uniform$(0, 1)$ the posterior will be

$$f(\theta \mid \mathbf{y}) \propto f^*(\theta) := (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_3},$$

for $\theta \in (0, 1)$. We wish to sample from this using a Uniform$(0, 1)$ proposal density, that is, $g(\theta \mid \mathbf{y}) = 1$, for $\theta \in (0, 1)$. Because we do not know the normalizing constant, we may use weighted resampling, which is an approximate algorithm. We do this by first generating $\Theta_1, \dots, \Theta_n \sim g(\theta) \sim$ Uniform$(0, 1)$ and then calculate the weights

$$w(\Theta_i) = \frac{f(\Theta_i)/g(\Theta_i)}{\sum_{j=1}^n f(\Theta_j)/g(\Theta_j)} = \frac{f(\Theta_i)}{\sum_{j=1}^n f(\Theta_j)},$$

because $g(\theta) = 1$ for all $\theta \in (0, 1)$. We then generate a second sample of size m from the discrete distribution $\{\Theta_1, \dots, \Theta_n\}$ with probabilities $w(\Theta_1), \dots, w(\Theta_n)$. This is done in the code block below, where we choose m such that $n/m = 20$.

```r
posterior_f_star <- function(theta, y) {
  return((2 + theta)^(y[1]) * (1 - theta)^(y[2] + y[3]) * theta^(y[4]))
}

weighted_resampling_f <- function(n, y) {
  Theta <- runif(n)   # Generate n Uniform(0, 1) variables
  f_star <- posterior_f_star(Theta, y)    # Calculating the vector f_star
  W <- f_star / sum(f_star)   # Calculating the weights
  m <- n / 20   # Using m such that n / m = 20
  X <- sample(Theta, size = m, prob = W)  # Sample m values from Theta with probability W
  return(X)
}
```

Subproblem 2.

Drawing $\Theta_1, \dots, \Theta_M \sim f(\theta \mid \mathbf{y})$, the Monte Carlo estimate of $\mu = \mathrm{E}(\theta \mid \mathbf{y})$ is

$$\hat{\mu} = \frac{1}{M}\sum_{i=1}^{M}\Theta_i.$$

We do this for $M = 10000$ in the code block below. Figure 1 shows the result of this. We see the estimation of the posterior mean $\mathrm{E}(\theta \mid \mathbf{y})$ using Monte Carlo integration and numerical integration together with the theoretical posterior density distribution and a generated histogram of the samples. In the figure the posterior density is plotted using a normalizing constant we find by numerical integration in `R` below, giving the normalizing constant `norm_const`.

```r
set.seed(69)

M <- 10000    # Number of samples from f(theta | y)
n <- 20 * M   # Number of samples needed to give M samples from f(theta | y)
y <- c(125, 18, 20, 34)   # Observed data
Theta_samp <- weighted_resampling_f(n, y)   # M samples from f(theta | y)
mu_est <- mean(Theta_samp)    # = 1/M * sum(Theta_samp)

# Integrating numerically
norm_const <- integrate(function(theta)(posterior_f_star(theta, y)),
                        lower = 0,
```

```r
                        upper = 1)$value  # Integrate to find normalizing constant
posterior_f <- function(theta, y) {   # Creating the true posterior f
  return(posterior_f_star(theta, y) / norm_const)
}
mu_num <- integrate(function(theta)(theta * posterior_f(theta, y)),
                    lower = 0,
                    upper = 1)$value    # Value of mu by numerical integration

# Plot
ggplot() +
  geom_histogram(
    data = as.data.frame(Theta_samp),
    mapping = aes(x = Theta_samp, y = ..density..),
    binwidth = 0.01,
    boundary = 0
  ) +
  stat_function(
    fun = posterior_f,
    args = list(y = y),
    aes(col = "Posterior density")
  ) +
  geom_vline(
    aes(xintercept = c(mu_est, mu_num),
        col = c("Estimated posterior mean", "Numerical posterior mean"),
        linetype = c("dashed", "dotted"))
  ) +
  guides(linetype = FALSE) +    # Remove linetype from label
  ggtitle("Estimation of the posterior mean") +
  xlab("theta") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.title = element_blank())
```

In the following code block we find the values of `mu_est` and `mu_num`.

```r
mu_est
```

```
## [1] 0.6225143
```

```r
mu_num
```

```
## [1] 0.6228061
```

From this it is clear that the estimated posterior mean is $\hat{\mu} \approx 0.623$ using Monte Carlo integration, and $\mu \approx 0.623$ using numerical integration with `integrate()`. Figure 1 also shows that these means corresponds well to the real posterior mean.

Subproblem 3.

Subproblem 4.

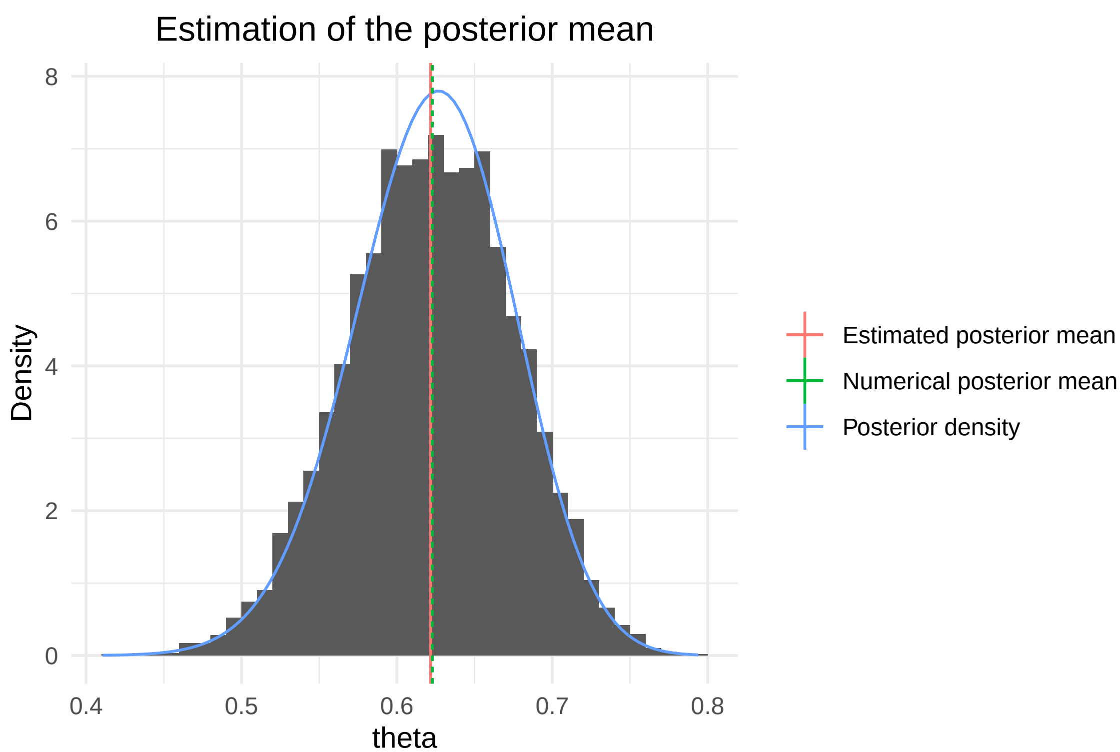

Figure 1: Estimation of the posterior mean $E(\theta \mid \mathbf{y})$ using Monte Carlo integration and numerical integration. A histogram of the samples is also shown together with the theoretical posterior density distribution.