Exercise 1

TMA4300 Computer Intensive Statistical Models

Mads Adrian Simonsen, William Scott Grundeland Olsen

08 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 $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix}^\top$ and the observed data is $\mathbf{y} = \begin{bmatrix} 125 & 18 & 20 & 34 \end{bmatrix}^\top$. The multinomial mass function is given as 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 $\theta \in (0,1)$. To do a rejection sampling (not weighted rejection sampling), we need to know the normalizing constant of $f(\theta \mid \mathbf{y})$. That is, the constant K such that $f(\theta \mid \mathbf{y}) = K f^*(\theta \mid \mathbf{y})$. This can be found as

$$\frac{1}{K} = \int_{\mathbb{R}} f^*(\theta \mid \mathbf{y})\,\mathrm{d}\theta = \int_0^1 f^*(\theta \mid \mathbf{y})\,\mathrm{d}\theta \approx 2.3577 \cdot 10^{28},$$

and we find it using the `integrate()`-function in `R` below. To use the rejection sampling we also need that

$$\frac{f(\theta \mid \mathbf{y})}{g(\theta \mid \mathbf{y})} = f(\theta \mid \mathbf{y}) \leq k,$$

and a value for k is found in the code block below. This is done by setting $k = \max_\theta f^*(\theta \mid \mathbf{y})$, and for the observed $\mathbf{y}$, we have that

$$\frac{\mathrm{d}f^*(\theta \mid \mathbf{y})}{\mathrm{d}\theta} = (\theta-1)^{37}\theta^{33}(\theta+2)^{124}(197\theta^2 - 15\theta - 68) = 0.$$

Because $\theta \in (0,1)$, the only factor we need to consider is $197\theta^2 - 15\theta - 68 = 0$, giving

$$\theta_{\max} = \frac{15 + \sqrt{53809}}{394},$$

such that $k = f^*(\theta_{\max} \mid \mathbf{y})$. This however is found numerically with the `optimize()`-function in `R` below. We then simulate $\Theta \sim$ Uniform$(0,1)$ and $U \sim$ Uniform$(0,1)$ and calculate $\alpha = f(\theta \mid \mathbf{y})/k$. Then, if $U \leq \alpha$, Θ is returned, and if not, the procedure is run again. We then sample from the posterior distribution in the code block below.

```r
y <- c(125, 18, 20, 34)   # Observed data

# Define the un-normalized posterior distribution f*(theta | y)
posterior_star <- function(theta, y) {
  return((2 + theta)^(y[1]) * (1 - theta)^(y[2] + y[3]) * theta^(y[4]))
}

# Find the normalizing constant 1 / K
norm_const <- integrate(function(theta)(posterior_star(theta, y)),
                        lower = 0,
                        upper = 1)$value

# Defining the normalized posterior distribution f(theta | y)
posterior <- function(theta, y) {
  return(posterior_star(theta, y) / norm_const)
}

# Finding the maximum
posterior_star_max <- optimize(function(theta)(posterior_star(theta, y)),
                               interval = c(0, 1),
                               maximum = TRUE)$objective

# k such that f(theta | y) <= k
k <- posterior_star_max / norm_const
```

```r
# Rejection sampling algorithm
rejection_sampling <- function(M, y) {
  count <- 0      # Count how many times the algorithm runs
  accept <- c()   # List of the accepted samples

  while(length(accept) < M) {   # Running until we have M accepted samples
    U <- runif(1)   # One Uniform(0, 1) sample
    Theta <- runif(1)   # One Uniform(0, 1) sample
    alpha <- posterior(Theta, y) / k
    if(U <= alpha) {
      accept <- rbind(accept, Theta)    # Appending to the list of accepted samples
    }
    count <- count + 1    # while loop has run one more time
  }
  return(list("accept" = accept, "co" = count))
}
```

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`.

```r
set.seed(69)

M <- 10000    # Number of samples from f(theta | y)

Theta_samp <- rejection_sampling(M, y)   # M samples from f(theta | y)
mu_est <- mean(Theta_samp$accept)    # = 1/M * sum(Theta_samp)

mu_num <- integrate(function(theta)(theta * posterior(theta, y)),
                    lower = 0,
                    upper = 1)$value    # Value of mu by numerical integration

# Plot
ggplot() +
  geom_histogram(
    data = as.data.frame(Theta_samp$accept),
    mapping = aes(x = Theta_samp$accept, y = ..density..),
    binwidth = 0.01,
    boundary = 0
  ) +
  stat_function(
    fun = posterior,
    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())
```

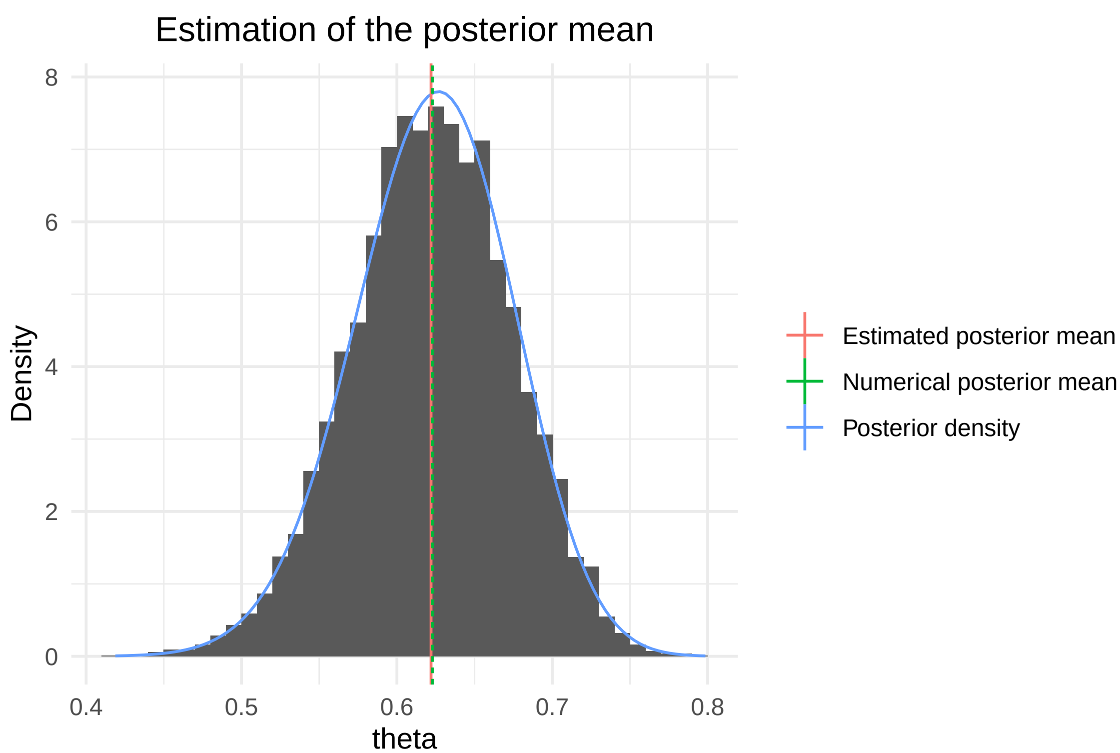

Figure 1: Estimation of the posterior mean $\mathrm{E}(\theta \mid \mathbf{y})$ using Monte Carlo integration and numerical integration. A histogram of the samples is also shown together with the theoretical posterior density distribution.

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

```
mu_est
```

```
## [1] 0.6235803
```

```
mu_num
```

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

From this it is clear that the estimated posterior mean is $\hat{\mu} \approx 0.624$ 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.

We are now interested in the number of random numbers the sampling algorithm needs to obtain one sample from $f(\theta \mid \mathbf{y})$. The expected number of trials up to the first sample from $f(\theta \mid \mathbf{y})$ is c given by the condition

$$\frac{f(\theta \mid \mathbf{y})}{g(\theta \mid \mathbf{y})} = f(\theta \mid \mathbf{y}) \leq k.$$

We may then choose

$$k \geq \max_{\theta \in [0,1]} f(\theta \mid \mathbf{y}),$$

and we chose the equality earlier. This was found earlier and stored in the variable `k`, which we can see the value of in the code block below.

```
k
```

```
## [1] 7.799308
```

Using the sampler, the expected number of random numbers that has to be generated in order to obtain one sample of $f(\theta \mid \mathbf{y})$ is the amount of times the ´while´ loop runs divided by the length amount of samples of $f(\theta \mid \mathbf{y})$. This value is given in the following code block.

```
Theta_samp$co / M
```

```
## [1] 7.8237
```

These corresponds well, and we see that we need to generate approximately 7.8 random numbers in order to obtain one sample of $f(\theta \mid \mathbf{y})$.

Subsection 4.

We now assume a $\text{Beta}(1, 5)$ prior

$$\tilde{f}(\theta) = \frac{1}{\mathrm{B}(1,5)}(1-\theta)^4 \propto (1-\theta)^4,$$

where $\mathrm{B}(\cdot, \cdot)$ is the beta function. This gives the posterior

$$\tilde{f}(\theta \mid \mathbf{y}) \propto f(\mathbf{y} \mid \theta)\tilde{f}(\theta) \propto \tilde{f}^*(\theta \mid \mathbf{y}) := (2+\theta)^{y_1}(1-\theta)^{y_2+y_3+4}\theta^{y_4}.$$

We wish to estimate μ by importance sampling, and to avoid needing to know the normalizing constant, we use the self-normalizing importance sampling estimator

$$\tilde{\mu}_{\mathrm{IS}} = \frac{\sum_{i=1}^n \Theta_i w(\Theta_i)}{\sum_{i=1}^n w(\Theta_i)},$$

where

$$w(\Theta_i) = \frac{\tilde{f}^*(\Theta_i \mid \mathbf{y})}{f^*(\Theta_i)} = (1-\Theta_i)^4,$$

and $\Theta_1, \ldots, \Theta_n \sim f(\theta \mid \mathbf{y})$. In the limit $n \to \infty$, the estimator $\tilde{\mu}_{\mathrm{IS}}$ is unbiased.

```
set.seed(69)

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

importance_sampling <- function(n, y) {
  Theta <- rejection_sampling(n, y)$accept    # Sample Theta from f(theta | y)
```

```
  w <- (1 - Theta)^4    # Calculate the weights w
  importance_mean <- sum(Theta * w) / sum(w)    # Self-normalizing importance sampling
  return(importance_mean)
}

# Test
n <- 10000
y <- c(125, 18, 20, 34)   # Observed data

mu_is <- importance_sampling(n, y)    # Estimated using importance sampling
mu_is
```

```
## [1] 0.5947878
```

```
# Finding mu using numerical integration
const <- integrate(function(theta)(posterior_tilde_star(theta, y)),
                   lower = 0,
                   upper = 1)$value
true_mu <- integrate(function(theta)(theta * posterior_tilde_star(theta, y) / const),
                     lower = 0,
                     upper = 1)$value
true_mu
```

```
## [1] 0.5959316
```

We then see that the estimated $\tilde{\mu}_{\text{IS}} \approx 0.5948$, while the posterior mean using numerical integration is $\mu \approx 0.5959$, both of which corresponds well. We notice that $\tilde{\mu}_{\text{IS}} < \hat{\mu}$, as found in Subproblem 2. This can be explained by looking at the priors. The previous prior $\text{Beta}(1, 1)$ and the new prior $\text{Beta}(1, 5)$ are shown in Figure 2. From this it is clear that the $\text{Beta}(1, 5)$ prior favors lower values for θ compared to the uniform prior, which do not favor any particular θ. It is therefore expected that $\tilde{\mu}_{\text{IS}} < \hat{\mu}$, which is shown to be true.

```
ggplot() +
  stat_function(
    fun = dbeta,
    args = list(shape1 = 1, shape2 = 1),
    aes(col = "Beta(1, 1)")
  ) +
  stat_function(
    fun = dbeta,
    args = list(shape1 = 1, shape2 = 5),
    aes(col = "Beta(1, 5)")
  ) +
  ggtitle("The different priors") +
  xlab("theta") +
  ylab("Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.title = element_blank())
```

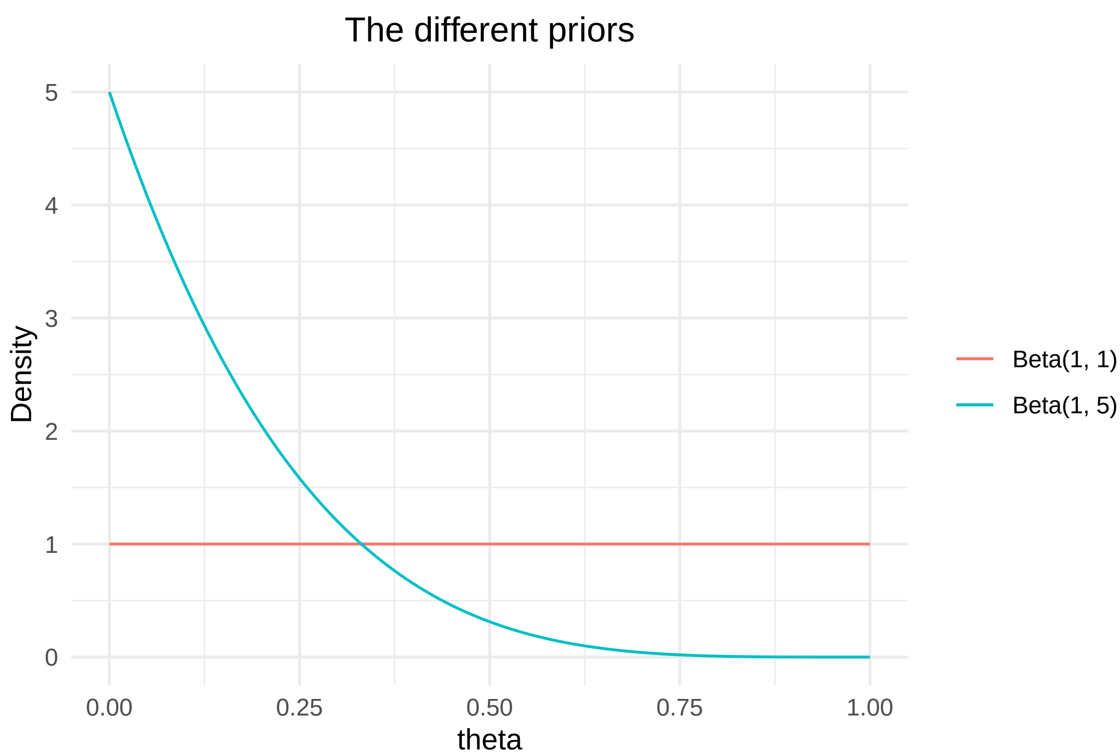

Figure 2: The two priors Beta$(1, 1)$ and Beta$(1, 5)$.