Statistical Programming Practical

Name: Dominic Lee
Student ID: 201691989

All R code can be found in the appendix at the end.
All propositions/theorems/lemmas mentioned refer to the book which the course is based on.

1 Task 1

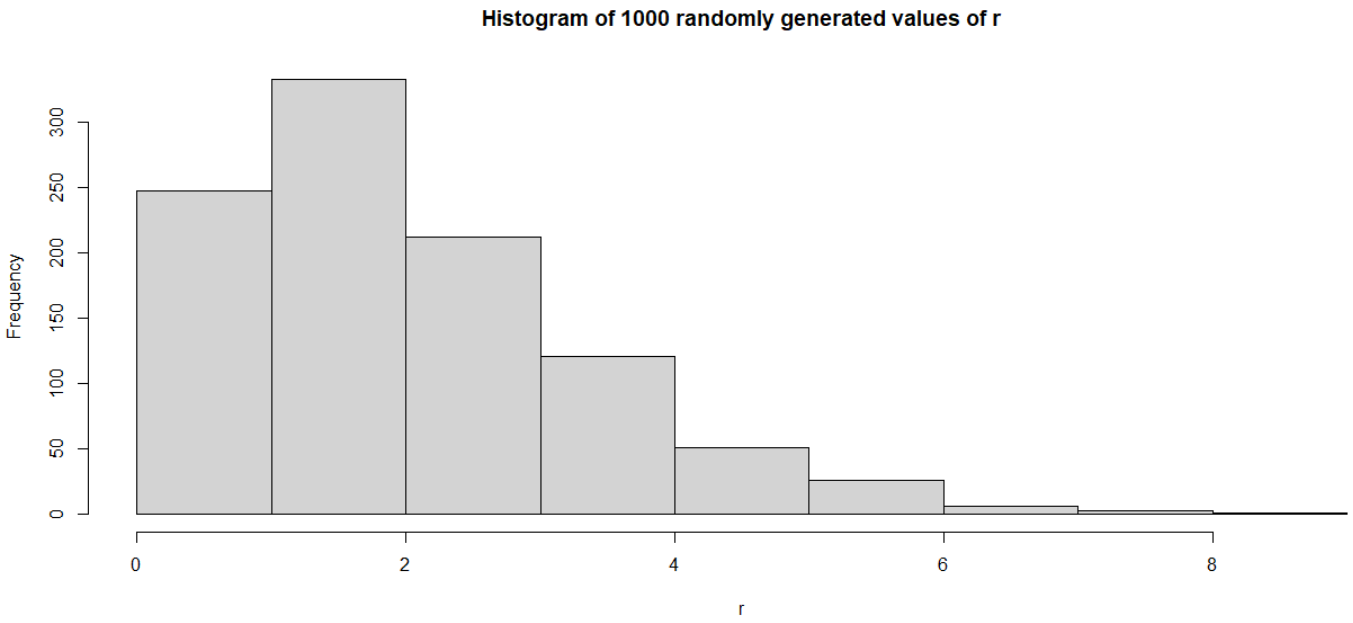

Figure 1: A histogram of 1000 randomly generated values of r from the $\Gamma(2,1)$ distribution

2 Task 2

We can model this situation with a binomial distribution, where the "trials" are the the random points, and a trial is a "success" if the point is in the circle. So the number of trials is n and the probability of success is $P(X_i^2+Y_i^2 \leq r^2) = 1-e^{-r^2/2}$.

So if $X \sim \text{Bin}(n, 1-e^{-r^2/2})$, we simply get $p(k|r) = P(X=k) = \binom{n}{k}(1-e^{-r^2/2})^k(e^{-r^2/2})^{n-k} = \binom{n}{k}(1-e^{-r^2/2})^k e^{-(n-k)r^2/2}$

By Bayes' formula, $p(r|k) = \frac{p(k|r)p(r)}{p(k)} = \frac{\binom{n}{k}}{p(k)} \cdot (1-e^{-r^2/2})^k e^{-(n-k)r^2/2} \cdot r^{\alpha-1}e^{-\beta r} = \text{const} \cdot (1-e^{-r^2/2})^k e^{-(n-k)r^2/2} \cdot r^{\alpha-1}e^{-\beta r}$ since, as said in the hint, $p(k)$ is constant in r.

We can use envelope rejection sampling to generate samples from the posterior density $p(r|k)$ by using the prior density $g(r) := \frac{1}{Z}r^{\alpha-1}e^{-\beta r}$ for the proposals, $f(r) := (1-e^{-r^2/2})^k e^{-(n-k)r^2/2} \cdot r^{\alpha-1}e^{-\beta r}$ as the target function and $c = Z$. Indeed, since $0 \leq (1-e^{-r^2/2})^k \leq 1$ and $0 \leq e^{-(n-k)r^2/2} \leq 1$ for all r, $(1-e^{-r^2/2})^k e^{-(n-k)r^2/2} \leq 1$ therefore $f(r) \leq r^{\alpha-1}e^{-\beta r} = cg(r)$ for all r. Also notice that the density we sample from is proportional to $f(r)$, and since $p(r|k)$ is also proportional to $f(r)$, we must be sampling from $p(r|k)$ (otherwise one of the two densities wouldn't integrate to 1) which is what we want.

3 Task 3

Note that for $\alpha = 2, \beta = 1$, $Z = \int_0^\infty xe^{-x}dx = 1$. Code to implement the envelope rejection sampling method can be found in the appendix. Below is a histogram of the results.

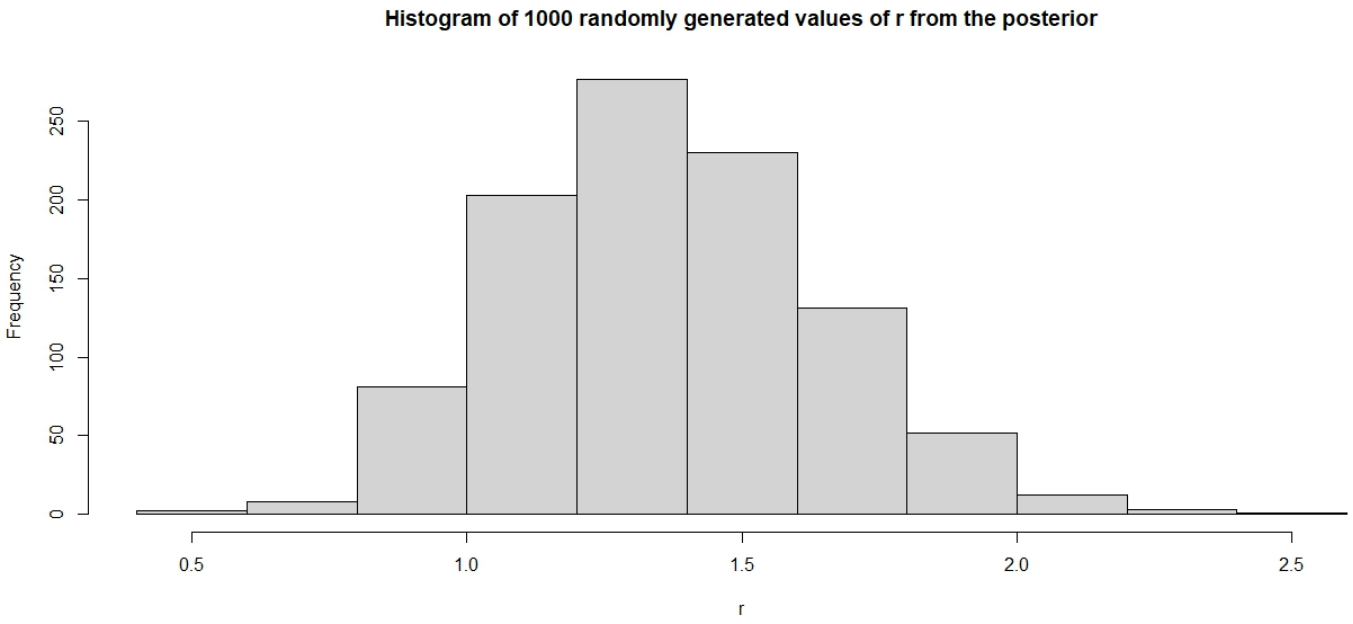

Figure 2: A histogram of 1000 randomly generated values of r from the posterior distribution with $n = 10, k = 6$

We can see this histogram is less spread out than the one in task 1 (the largest value is between 2 and 2.5 as opposed to task 1 where the largest value is between 7 and 8), i.e. the data has a lower sample variance. This makes sense intuitively, since we now have more information about r, so we would expect to be able to estimate it better, i.e. our posterior distribution should have a lower variance.

It also seems like the mean of the new histogram is slightly smaller than the mean of that in task 1 (here it seems < 1.5 but in task 1 it seems to be > 1.5). Intuitively this should imply that when we know nothing about r other than its prior distribution, the probability of any random point being in the circle $P(k = 1|n = 1)$ is over 60%, since here observing 60% of points in the circle gives us a smaller mean.

4 Task 4

4.1 Estimating $E(R|K = 6)$

For the Monte Carlo method, we want to use the estimation $E(R|K = 6) \approx \frac{1}{N}\sum_{i=1}^{N} r_i$ where the r_i are i.i.d. with density $p(r|6)$. Since this is a difficult distribution to sample from, we'll use the envelope rejection sampling method from task 3. To simplify notation, let $\theta = E(R|K = 6)$ and $\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} r_i$.

Now we need to pick an N to control the error. By proposition 3.14, $\text{bias}(\hat{\theta}) = 0$ and $\text{mse}(\hat{\theta}) = \text{Var}(\hat{\theta}) = \frac{1}{N}\text{Var}(r_i)$, so to control the mse directly, we would need to know $\text{Var}(r_i)$. Unfortunately, this is difficult to work out because we would need to know the constant in the expression for $p(r|6)$. Instead we will run a Monte Carlo estimate for $\text{Var}(r_i)$ with $N = 1000$, then use the estimator $\hat{\sigma^2} = \frac{1}{999}\sum_{i=1}^{1000}(r_i - \bar{r_i})^2$ where $\bar{r_i} = \frac{1}{1000}\sum_{i=1}^{1000} r_i$. I picked 1000 because it takes my computer around 15 seconds - not too long. And when I run it multiple times it gives me fairly consistent answers. It gives an estimate for the variance of 0.07891096.

To be safe and to make calculations easier, we will take it as 0.1. So, if we want, say, RMSE ≤ 0.01, we need N such that $\sqrt{\frac{0.1}{N}} \leq 0.01$, i.e. $N \geq 1000$. Using $N = 1000$, we get a Monte Carlo estimate of $E(R|K = 6) \approx 1.368254$. Also, this program took around 15 seconds to run so the value of N isn't too large. The time was kept down by not having to store all sample values of r_i, we merely added them to the current sum.

4.2 Estimating $P(R < 1|K = 6)$

To compute $P(R < 1|K = 6)$ using the Monte Carlo estimate, we rewrite $P(R < 1|K = 6) = E(\mathbb{I}_{\{R<1\}}|K = 6)$. We now use the Monte Carlo estimate $P(R < 1|K = 6) \approx \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{\{r_i<1\}}$ where again the r_i are i.i.d. with density $p(r|6)$. Again we'll use the envelope rejection sampling method and to simplify notation, let $\theta = P(R < 1|K = 6)$ and $\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{\{r_i<1\}}$. We'll roughly follow the same steps as for the previous estimate.

We need to pick a value of N to control the error. Again, $\text{bias}(\hat{\theta}) = 0$ and $\text{mse}(\hat{\theta}) = \text{Var}(\hat{\theta}) = \frac{1}{N}\text{Var}(\mathbb{I}_{\{r_i<1\}})$. Luckily this time $\text{Var}(\mathbb{I}_{\{r_i<1\}})$ is easy to bound:

$$\begin{aligned}
\text{Var}(\mathbb{I}_{\{r_i<1\}}) &= E(\mathbb{I}^2_{\{r_i<1\}}) - E(\mathbb{I}_{\{r_i<1\}})^2 && (1)\\
&= \theta - \theta^2 && (2)\\
&= \theta(1-\theta) && (3)\\
&\leq 1/4 && (4)
\end{aligned}$$

Where line (4) comes from the fact that θ is a probability so $0 \leq \theta \leq 1$ and $x(1-x) \leq 1/4$ for all $x \in [0, 1]$.

So $\text{mse}(\hat{\theta}) \leq \frac{0.25}{N}$, so this time to guarantee that $\text{RMSE}(\hat{\theta}) \leq 0.01$, we should take $N \geq 2500$. However, I ran the code a few times, and the value seems to be around 0.08, so a RMSE of 0.01 is not that good a relative error. Therefore I ran it with $N = 10000$, which took a long time, but should give a bit better RMSE of $0.01 \times \sqrt{\frac{1}{4}} = 0.005$.

Using $N = 10000$, we get a Monte Carlo estimate of $P(R < 1|K = 6) \approx 0.0821$. This program took a long time to run, so it would be nicer to get a better error by using a larger N but this isn't really practical.

5 Appendix

5.1 Task 1

```
x=rgamma(1000,2,1)
hist(x)
```

5.2 Task 3

```
g <- function(r) {r*exp(-r)} #defined my own function instead of using dgamma to make the code neater
f <- function(r) {(1-exp((-r^2)/2))^6*exp(-2*r^2)*g(r)}


samples=integer(1000) #initiates the vector samples, which will contain our 1000 samples.

for(i in 1:1000) {
        x=1
        u=2*f(x)/g(x) #these values of u and x make sure to trigger the while loop
        while(f(x) < u*g(x)) {
                x= rgamma(1,2,1)
                u=runif(1,0,1)
                if(f(x) >= u*g(x)) {
                        samples[i]=x}}}

hist(samples)
```

5.3 Task 4

5.3.1 Estimating $E(R|K = 6)$

```
g <- function(r) {r*exp(-r)} #defined my own function instead of using dgamma to make the code neater
f <- function(r) {(1-exp((-r^2)/2))^6*exp(-2*r^2)*g(r)}
```

```
samples=integer(1000) #initiates the vector samples, which will contain our 10000 samples.

for(i in 1:1000) {
        x=1
        u=2*f(x)/g(x) #these values of u and x make sure to trigger the while loop
        while(f(x) < u*g(x)) {
                x= rgamma(1,2,1)
                u=runif(1,0,1)
                if(f(x) >= u*g(x)) {
                        samples[i]=x}}}
s=sum(samples)/1000
y=samples-s
sum(y^2)/999
```

```
g <- function(r) {r*exp(-r)} #defined my own function instead of using dgamma to make the code neater
f <- function(r) {(1-exp((-r^2)/2))^6*exp(-2*r^2)*g(r)}


s=0 #initiates the sum

for(i in 1:1000) {
        x=1
        u=2*f(x)/g(x) #these values of u and x make sure to trigger the while loop
        while(f(x) < u*g(x)) {
                x= rgamma(1,2,1)
                u=runif(1,0,1)
                if(f(x) >= u*g(x)) {
                        s=s+x}}}
s/1000
```

5.3.2 Estimating $P(R < 1|K = 6)$

```
g <- function(r) {r*exp(-r)} #defined my own function instead of using dgamma to make the code neater
f <- function(r) {(1-exp((-r^2)/2))^6*exp(-2*r^2)*g(r)}


s=0 #initiates the sum of the samples

for(i in 1:10000) {
        x=1
        u=2*f(x)/g(x) #these values of u and x make sure to trigger the while loop
        while(f(x) < u*g(x)) {
                x= rgamma(1,2,1)
                u=runif(1,0,1)
                if(f(x) >= u*g(x)) {
                        if(x < 1) {
                                s=s+1}}}}

s/10000
```