

HW4

Kwan Ho Lee

5/19/2021

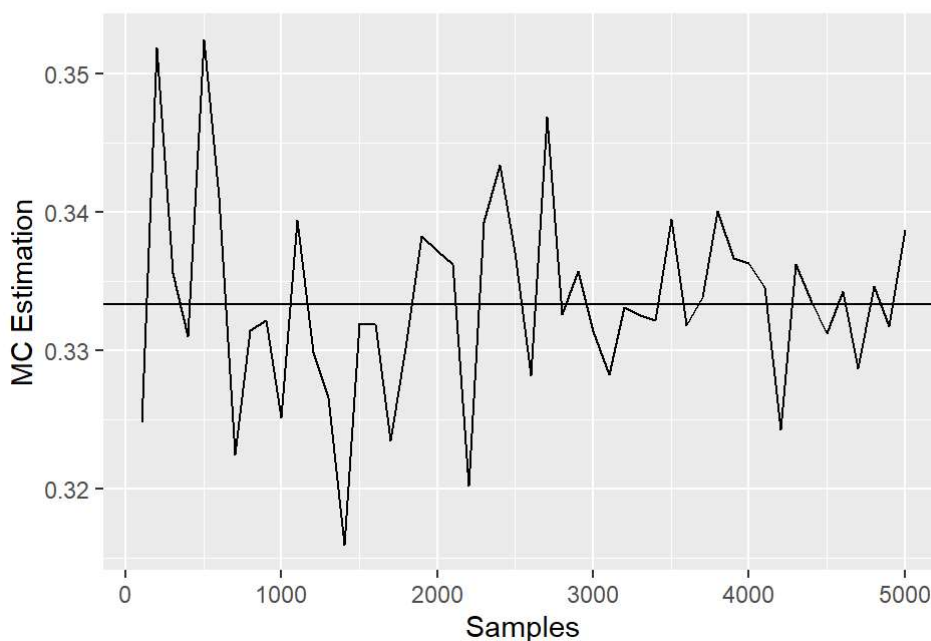
Problem 1

1a)

$$\int_0^1 x^2 dx$$

We use $f(x) = x^2$ and $p(x) = 1$ uniform over the $[0, 1]$ interval to approximate the integral using the formulation.

MC Estimation of $\int_0^1 x^2 dx$



The Monte Carlo estimation of $\int_0^1 x^2 dx$ is $\hat{I} = 0.3386958$. The exact evaluation is $I = \left. \frac{x^3}{3} \right|_0^1 = \frac{1}{3}$. Difference is $|I - \hat{I}| = 0.0053624$ which is only a factor of $\frac{|I - \hat{I}|}{1/3} = 0.0160873$.

1c)

$$\int_0^\infty \frac{3}{4} x^4 e^{-x^3/4} dx$$

Notice that $\int_0^\infty e^{-x^3/4} dx = 2^{2/3} \Gamma(\frac{4}{3}) = k$ so we use

$$p(x) = \frac{1}{k} e^{-x^3/4}$$

and

$$f(x) = k \frac{3}{4} x^4$$

Then we substituted into the original integral, we see

$$\int_0^\infty \frac{3}{4} x^4 e^{-x^3/4} dx = \int_0^\infty f(x) dp(x)$$

Hence, we sample from $p(x)$ using inverse CDF sampling and evaluate the samples at $f(x)$ for a large number of samples following the MC integration method. However, $P(x) = \int_0^x p(y) dy$ must first be calculated:

$$P(x) = \int_0^x p(y) dy = \frac{1}{k} \left(k - \frac{x}{3} E_{2/3} \left(\frac{y^3}{4} \right) \right)$$

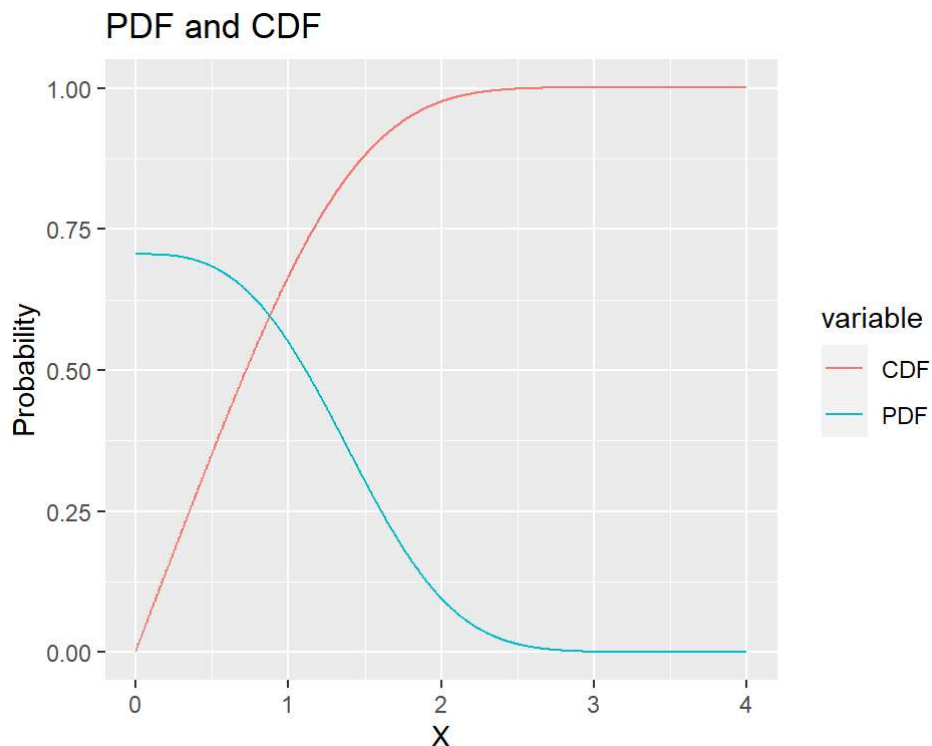
where

$$E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n} dt = x^{n-1} \Gamma(1-n, x)$$

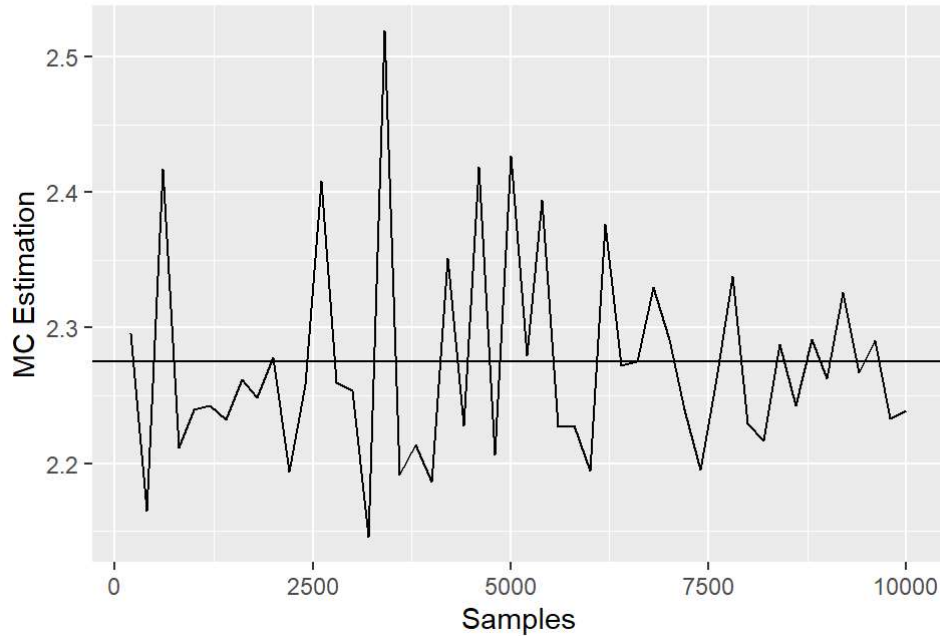
and

$$\Gamma(a, x) = \int_x^\infty t^a e^{-t} dt$$

Finally, we calculate the inverse CDF and the CDF numerically.



MC Estimation of $\int_0^{\infty} \frac{3}{4} x^4 e^{-x^3/4}$



Problem 2

Here we use importance sampling to approximate

$$I = \frac{1}{\sqrt{2\pi}} \int_1^2 e^{-x^2/2} dx$$

using $f(x) = e^{-x^2/2}/\sqrt{2\pi}$ and $p(x) = \mathbf{1}_{x \in [1,2]}$. Importance sampling is the approximation of

$$\mu = E_p(f(x)) \approx \hat{\mu}_g = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)p(x_i)}{g(x_i)}$$

where $X_i \sim g(X)$ iid. We rationalize this approximation using the law of large numbers and the fact that

$$\mu = E_p(f(x)) = \int_D f(x) dp(x) = \int_D \frac{f(x)g(x)}{g(x)} dp(x) = \int_D \frac{f(x)p(x)}{g(x)} dg(x) = E_g\left(\frac{f(x)p(x)}{g(x)}\right)$$

where $\frac{p(x)}{g(x)}$ is the *likelihood ratio*, $g(x)$ is the *importance distribution* given to be $g(x) = N(1.5, \nu)$ for $\nu \in \{0.1, 1, 10\}$, and $p(x)$ is the *nominal distribution*. In this problem, we use $p(x) \sim \mathbf{1}_{[1,2]}$ and $f(x) = e^{-x^2/2}/\sqrt{2\pi}$.

$$\hat{I}_{\nu=0.1} = 0.0881578$$

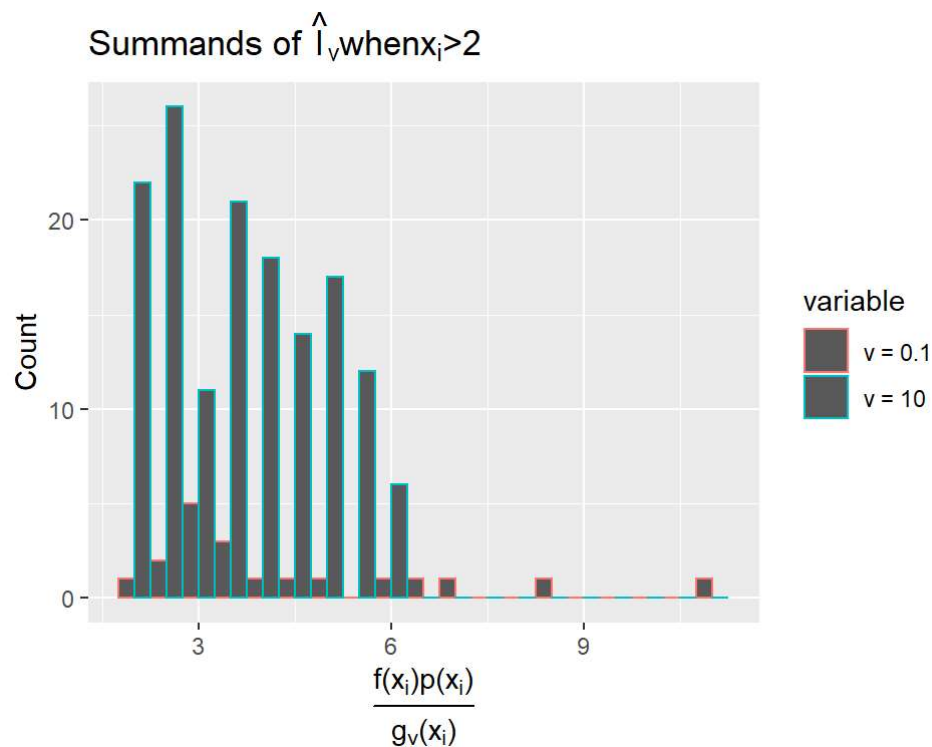
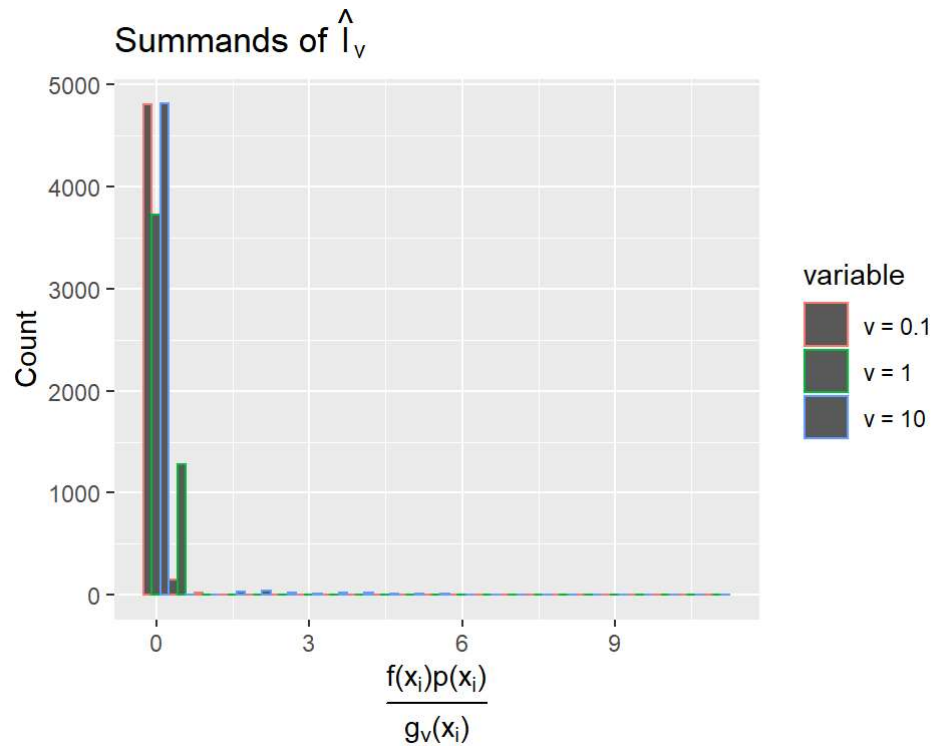
$$\hat{I}_{\nu=1} = 0.1346239$$

$$\hat{I}_{\nu=10} = 0.1213762$$

The true value of the integral is

$$\int_1^2 \frac{e^{-x^2/2}}{\sqrt{2\pi}} dx \approx 0.135905$$

Following we observe the histogram of the summands:



We observe that when the importance distribution is too wide or too narrow, there are many aberrant values in the summand for the approximation $\hat{I} = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)p(x_i)}{g(x_i)}$.

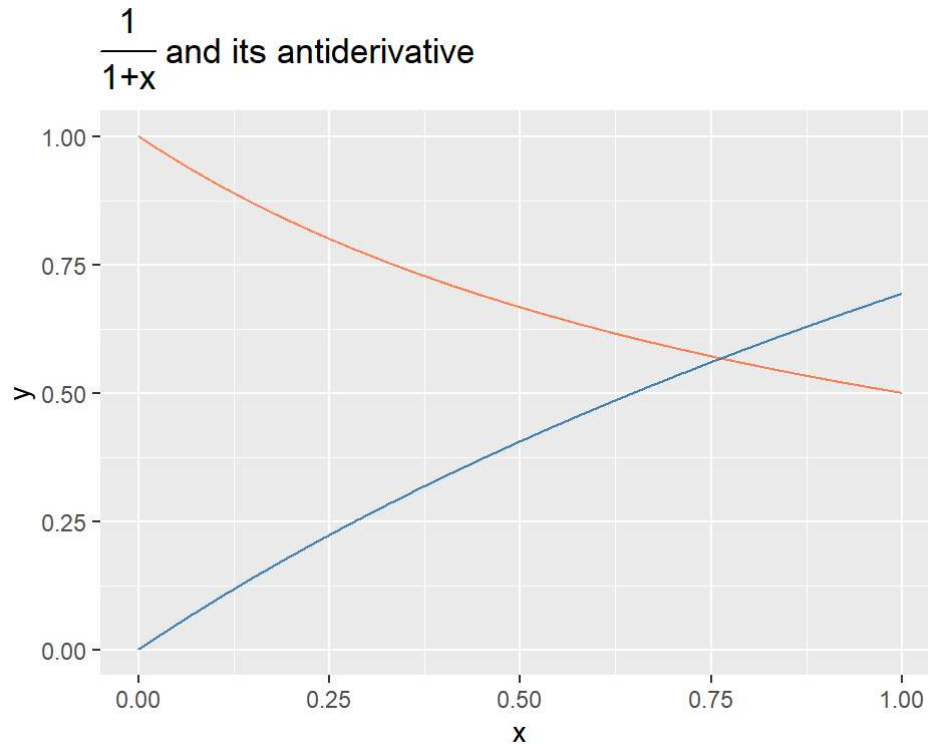
Problem 3

$$I = \int_0^1 \frac{1}{1+x} dx$$

3a)

We simulate the integral using $h(x) = \frac{1}{1+x}$ and $U_i \sim U[0, 1]$ for $i \in \{1, \dots, n\}$ in the model

$I \approx \hat{I}_{MC} = \frac{1}{N} \sum_{i=1}^N h(u_i)$ where u_i are the observations of the random variables U_i .



The pink line above is the function $h(x) = \frac{1}{1+x}$ and the blue line is $H(x) = \int_0^x h(t)dt$.

The MCMC estimate of the integral is $\hat{I}_{MC} = 0.6885054$ compared with the true integral $I \approx 0.6931472$.

3b)

Using the control variate $c(x) = 1 + x$, we calculate

$\hat{I}_{CV} = \frac{1}{N} \sum_{i=1}^N h(U_i) + b(\frac{1}{N} \sum_{i=1}^N c(U_i) - E(c(U)))$ where the variance of \hat{I}_{CV} is minimized at $b = \frac{-cov(h(X), c(X))}{var(c(X))} \approx 0.476696$. Upon calculation, \hat{I}_{CV} is realized as 0.6931881.

3c)

The variance of \hat{I}_{CV} is

$$Var(\hat{I}_{CV}) = \frac{1}{N^2} Var(h(X)) - \left(\frac{b}{N}\right)^2 Var(c(X)) = Var(\hat{I}_{MC}) - \frac{1}{N^2} \frac{Cov(h(X), c(X))^2}{Var(c(X))}$$

Running the MC and CV estimators 300 times with 1500 samples, $Var(\hat{I}_{MC}) \approx 1.3687825 \times 10^{-5}$ and $Var(\hat{I}_{CV}) \approx 4.1128021 \times 10^{-7}$. Notice that this is an improvement of 1.5221966 orders of magnitude in the variance of the estimator.

3d)

A new estimator for I , $\hat{I}_0 = \log 2$ has 0 variance and 0 bias. Alternatively, we could add another control variate, for example $\hat{I}_{CV2} = \hat{I}_{CV} + b_2(\frac{1}{N} \sum_{i=1}^N c_2(U_i) - E(c_2(U)))$ where $c_2(x) = 1 - \frac{x}{2}$.

Problem 4

4a)

Using the model $y_{ij} = \mu + \alpha_i + e_{ij}$ where e_{ij} are sampled iid from a double exponential distribution, our null hypothesis H_0 is that $\alpha_i = 0$ for all $i \in \{1, \dots, 3\}$. The alternative hypothesis H_A is that, for at least one $i^* \in \{1, \dots, 3\}$, $\alpha_{i^*} \neq 0$.

To test the null hypothesis H_0 using Monte Carlo methods, we would execute the following algorithm:

```

Algorithm: Monte Carlo test
  input
    Y11...Y1N = Y.1,
    Y21...Y2N = Y.2,
    Y31...Y3N = Y.3,
      Observations for each of three treatments
    M,
      Number of samples for simulated empirical distribution
    K,
      Number of samples for each simulated statistic
    alpha
      Testing significance level
  output
    {0,1}
      0 if fail to reject H_0
      1 if reject H_0

m.hat = mean({Y1...YN})
b.hat = 1/2*sqrt(var(Y1...YN))
d.h0 = doubleExponential.pdf(mean=m.hat, sd=b.hat)

e.cdf = []
for i from 1 to M do:
  x.samp = sample(from=d.h0, n=K)
  x.stat = mean(x.samp)
  e.cdf.append(x.stat)

for i from 1 to 3 do:
  y.i.stat = mean(Y.i)
  if y.i.stat not in quantiles(e.cdf, alpha/2, 1-alpha/2) then:
    return 1
return 0

```

Hence, we generate a null empirical distribution using parameters estimated from the observed data and predicated on the H_0 model and reject the null hypothesis if any of the observed means for any of the treatments fall outside a reasonable simulated empirical quantile.

4b)

We could use Fisher's 2-sample permutation method applied to each pair of treatments. Note that this scales poorly ($O(n!)$) with the number of treatments.

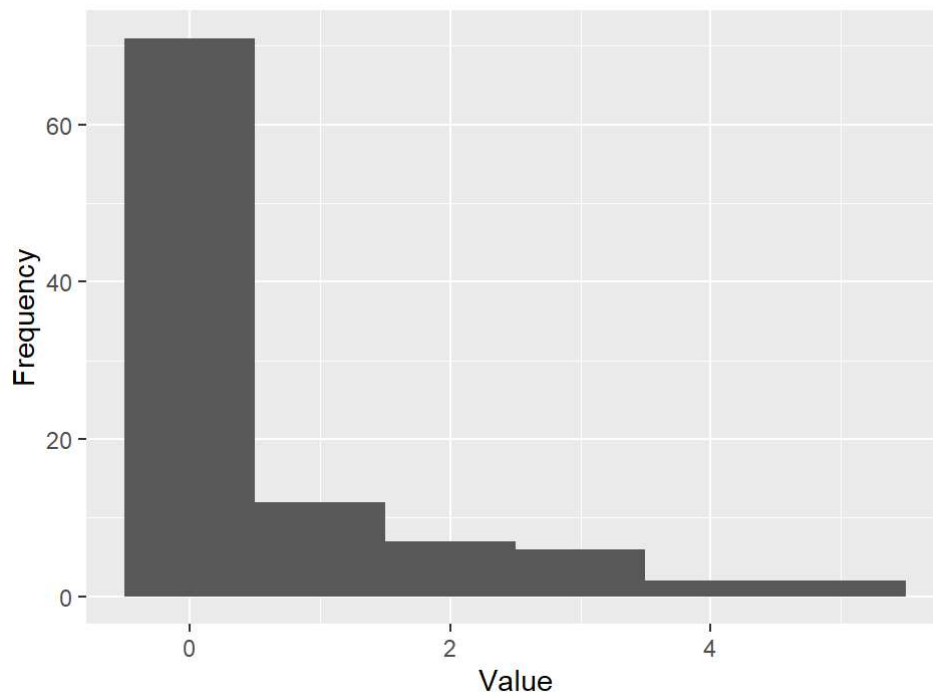
Problem 5

5a)

We seek to sample $n = 100$ times from $Poisson(\lambda r_i)$ with $r_i \sim Bernoulli(p)$ using $\lambda = 2$ and $p = 0.3$.

The sample generated using $p = 0.3$ and $\lambda = 2$ is below:

$x_i | r, \lambda, p$ when $p=0.3$ and 2



5b)

5b).i

$$f(\lambda | p, r, x) = \frac{b^a \lambda^{a-1} e^{-b\lambda}}{\Gamma(a)} p^{\sum(r_i)} (1-p)^{n-\sum(r_i)} \prod \frac{r_i^{x_i}}{x_i!} e^{-\lambda \sum r_i} \lambda^{\sum x_i} =$$

$$\lambda^{a+\sum x_i-1} e^{-(b+\sum r_i)\lambda} \frac{1}{N_{p,r,x}} = \mathbf{\Gamma}_{a+\sum x_i, b+\sum r_i}(\lambda)$$

5b).ii

$$f(\lambda | p, r, x) = \frac{b^a \lambda^{a-1} e^{-b\lambda}}{\Gamma(a)} p^{\sum(r_i)} (1-p)^{n-\sum(r_i)} \prod \frac{r_i^{x_i}}{x_i!} e^{-\lambda \sum r_i} \lambda^{\sum x_i} =$$

$$p^{\sum r_i} (1-p)^{n-\sum r_i} \frac{1}{N_{\lambda,r,x}} = \beta_{\sum r_i, n-\sum r_i}(p)$$

where

$$\frac{1}{N_{\lambda,r,x}} = \frac{b^a \lambda^{a-1} e^{-b\lambda}}{\Gamma(a)} \prod \frac{r_i^{x_i}}{x_i!} e^{-\lambda \sum r_i} \lambda^{\sum x_i}$$

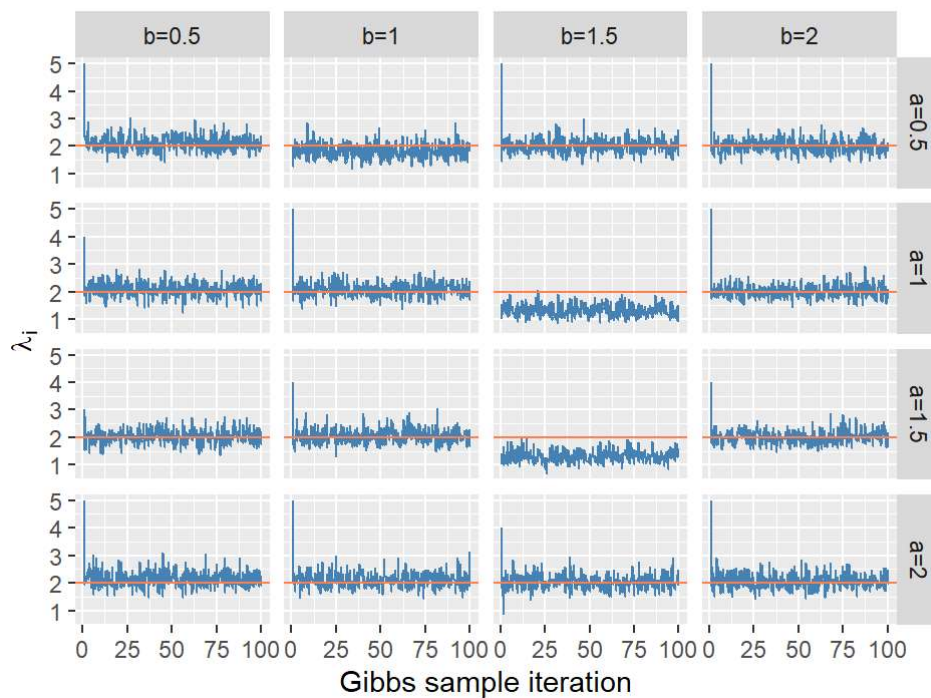
5b).iii

$$\begin{aligned} f(\lambda \mid p, r, x) &= \frac{b^a \lambda^{a-1} e^{-b\lambda}}{\Gamma(a)} p^{\sum(r_i)} (1-p)^{n-\sum(r_i)} \prod \frac{r_i^{x_i}}{x_i!} e^{-\lambda \sum r_i} \lambda^{\sum x_i} = \\ &= \left(\frac{b^a \lambda^{a-1} e^{-b\lambda}}{\Gamma(a)} \right) e^{-\lambda \sum r_i} (\lambda r_i)^{\sum x_i} \left(\prod \frac{1}{x_i!} \right) p^{r_i} (1-p)^{1-r_i} = \frac{1}{N_{\lambda,x,p}} \left(e^{-\lambda} \frac{p}{1-p} \right)^{r_i} r_i^{\sum x_i} = \\ &= \text{Bernoulli} \left(\frac{pe^{-\lambda}}{pe^{-\lambda} + (1-p)\mathbf{1}_{x_i=0}} \right) \end{aligned}$$

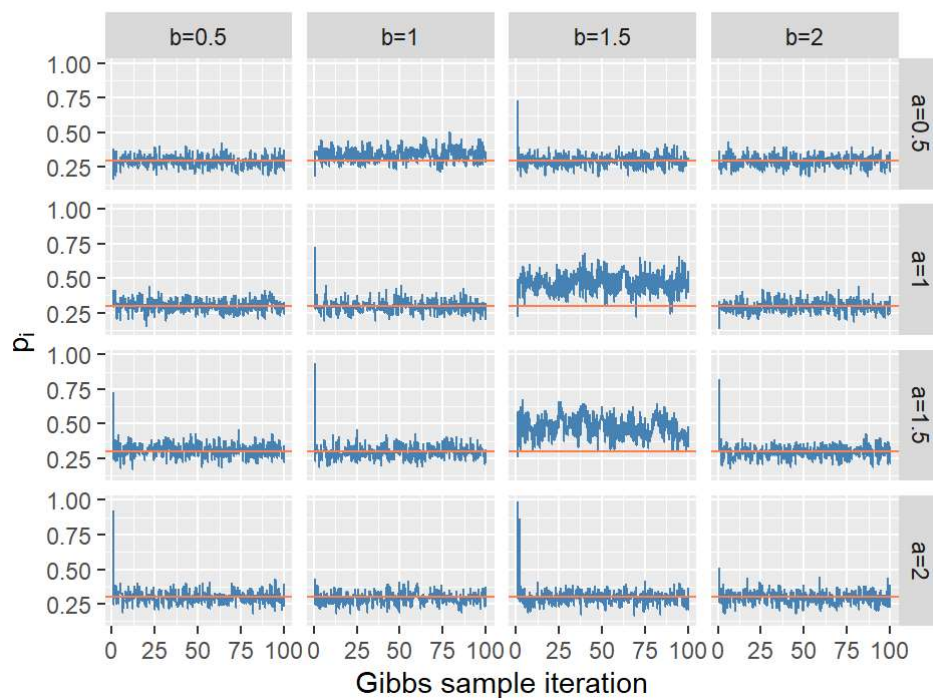
5c)

To sample from the posterior $f(\lambda, \mathbf{r}, p \mid \mathbf{x})$, we condition on the sample generated in part (5a). Then we expect to see high density around the values $p = 0.3$ and $\lambda = 2$ as were used to generate the sample.

Gibbs sampler convergence for λ using prior $f(p) \sim \beta(a,b)(p)$



Gibbs sampler convergence for p using prior $f(p) \sim \beta(a, b)$



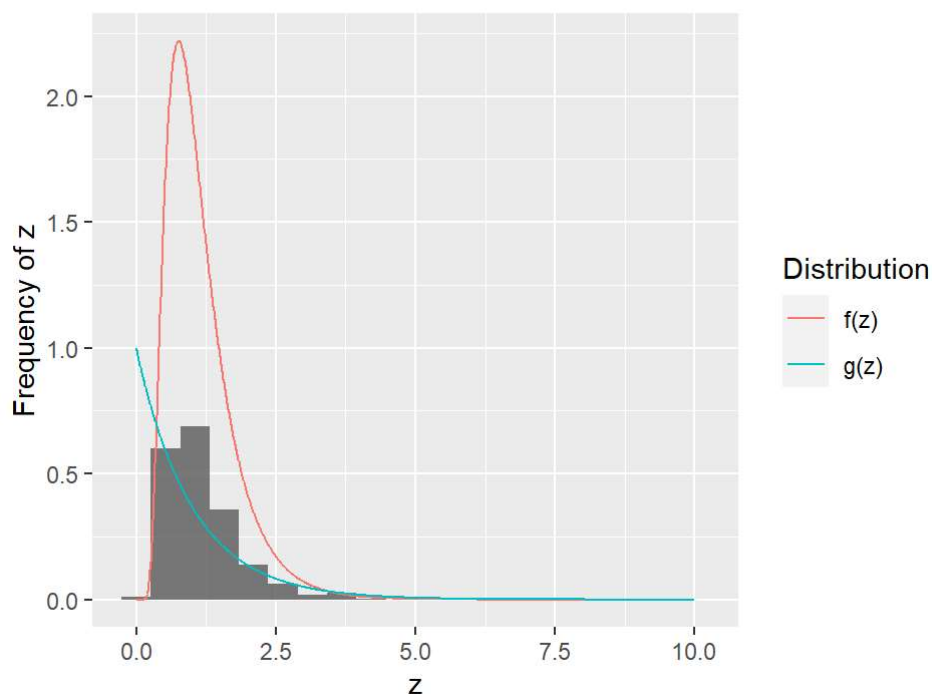
The Gibbs sampler stabilizes very quickly, typically within 15 iterations, though this is not true in general. The choice of prior heavily biases the estimates, as can be concluded by the distance from the stabilized region of the blue chain to the red true value.

Problem 6

$$f_{\theta_1, \theta_2}(z) \propto z^{-3/2} e^{-\theta_1 z - \frac{\theta_2}{z} + 2\sqrt{\theta_1 \theta_2} + \log \sqrt{2\theta_2}}$$

We draw 1000 samples by executing the following:

Target $f(z)$, envelope $g(z)$, and Metropolis-Hastings samp



Notice that the sample in grey follows the target distribution in red rather than the sampling envelope in blue even though the target and envelope have a small inner product. Furthermore, we can test the accuracy of this simulation using the following the target distribution:

$$E_f(Z) = \sqrt{\frac{\theta_2}{\theta_1}} = 1.1547005 \quad \text{and} \quad E_f\left(\frac{1}{Z}\right) = \sqrt{\frac{\theta_1}{\theta_2}} + \frac{1}{2\theta_2} = 1.1160254$$

In the sample produced by the M-H algorithm above:

$$\frac{1}{N} \sum_{i=1}^N z_i = 1.15011 \quad \text{and} \quad \frac{1}{N} \sum_{i=1}^N \frac{1}{z_i} = 1.1424996$$

Notice that the test statistics and their true values are pretty close, indicating by another means that the sample is accurately drawn from $f(z)$

Appendix

The following code is used in Question 1 part (a)

```
library(ggplot2)
library(latex2exp)
set.seed(100)

# Define functions

f = function(x){
  return(x^2)
}
p = function(x){
  return(1)
}
f = Vectorize(f)
p = Vectorize(p)

## Parameterize

N = 5000
m = 50
K = 10000
v = 1

## Simulate

pxs = seq(0, 1, length.out=K)
i1s = rep(NA, m)
ns = (1:m)*N/m
for(i in 1:m){
  xs = sample(pxs, ns[i], prob=p(pxs), replace=T)
  i1 = v*mean(f(xs))
  i1s[i] = i1
}

## Plot

df_plt1 = data.frame(x = ns, y = i1s)
plt1 = ggplot(data=df_plt1, aes(x=x, y=y)) +
  geom_line() +
  geom_hline(yintercept = 1/3) +
  labs(
    x="Samples",
    y="MC Estimation",
    title=TeX("MC Estimation of  $\int_0^1 x^2 dx$ ")
  )
plt1
```

The following code is used in Question 1 part (c)

```
library(reshape)

## Parameterize
N = 10000
m = 50
K = 1000
v = 1
mx = 4

## Define functions

k = 2^(2/3)*gamma(4/3)
f = function(x){
  return(k*3/4*x^4)
}
p = function(x){
  return(1/k*exp(-x^3/4))
}
f = Vectorize(f)
p = Vectorize(p)

## compute the CDF for inverse sampling

d = mx/K
pc_xs = seq(0, mx, by=d)
pdf = p(pc_xs)
cdf = cumsum(pdf)*d

# must calculate inverse cdf

us = seq(0, 1, length.out=K)
invcdf = rep(NA, K)
for(i in 1:length(us)){
  ix = sum(cdf<us[i]) + 1
  invcdf[i] = cdf[ix]
}

## Simulate
i3s = rep(NA, m)
ns = (1:m)*N/m

for(i in 1:m){
  xs = sample(pc_xs, ns[i], prob=pdf, replace=T)
  i3 = v*mean(f(xs))
  i3s[i] = i3
}

## Plot

pc_df = data.frame(x=pc_xs, CDF=cdf, PDF=pdf)
pc_df_m = melt(pc_df, id="x")
```

```
plt3 = ggplot(pc_df_m, aes(x=x, y=value, color=variable)) +  
  geom_line() +  
  labs(x="X", y="Probability", title="PDF and CDF")  
plt3  
  
# simulation results  
  
tru = 2^(4/3)*gamma(5/3)  
df_plt4 = data.frame(x = ns, y = i3s)  
plt4 = ggplot(data=df_plt4, aes(x=x, y=y)) +  
  geom_line() +  
  geom_hline(yintercept = tru) +  
  labs(  
    x="Samples",  
    y="MC Estimation",  
    title=TeX(paste(  
      "MC Estimation of",  
      "$\\int_0^{\\infty} \\frac{3}{4} x^4 e^{-x^3/4}$"  
    ))  
  )  
plt4
```

The following code is used in Question 2

```
## Define functions

g = function(n, u, sd){

  return(dnorm(n, mean=u, sd=sd))
}
p = function(x){

  if(x<1){return(0)}
  else if(x>2){return(0)}
  else{return(1)}
}
p = Vectorize(p)
f = function(x){

  r = exp(-x^2/2)/sqrt(2*pi)
  return(r)
}

## Parameterize

m = 5000
k = 10000
u = 1.5
vs = c(0.1, 1, 10) # standard deviations

## Simulate

samples = list()
summands = list()
i = 1
for(v in vs){
  d = c(-1, 1)*(4*v) + u
  domain = seq(d[1], d[2], length.out=k)
  ps = g(domain, u=u, sd=v)
  xs = sample(domain, m, prob=ps, replace=T)
  samples[[i]] = xs
  summands[[i]] = (f(xs)*p(xs))/(g(xs, u=u, sd=v))
  i = i + 1
}
```

```

df_plt = data.frame(
  sum1 = summands[[1]],
  sum2 = summands[[2]],
  sum3 = summands[[3]]
)
names(df_plt) = c(
  paste("v =", vs[1]),
  paste("v =", vs[2]),
  paste("v =", vs[3])
)
df_plt_m = melt(df_plt) # format data

# plot data

plt1 = ggplot(
  data=df_plt_m,
  aes(x=value, color=variable)
) +
  geom_histogram(binwidth=0.5, position="dodge") +
  labs(
    x = TeX("$\\frac{f(x_i)p(x_i)}{g_v(x_i)}$"),
    y = "Count",
    title = TeX("Summands of $\\hat{I}_v$")
  )
plt1

# plot only observations larger than 2

plt2 = ggplot(
  data=df_plt_m[df_plt_m$value > 2,],
  aes(x=value, color=variable)
) +
  geom_histogram(binwidth=0.5, position="dodge") +
  labs(
    x = TeX("$\\frac{f(x_i)p(x_i)}{g_v(x_i)}$"),
    y = "Count",
    title = TeX("Summands of $\\hat{I}_v$ when $x_i > 2$")
  )
plt2

```

The following code is used in Question 3 part (a)

```
## Parameterize
N = 1500
us = runif(N, 0, 1)
itru = log(2)
xs = seq(0,1,length.out=N)

## Define functions

h = function(x){
  if(x<0){return(0)}
  else if(x>1){return(0)}
  else{
    return(1/(1+x))
  }
}
h = Vectorize(h)

## Simulate

i1 = mean(h(us))

## Plot h and int(h)

df_3.a = data.frame(x=xs, y=h(xs))
df_3.a.2 = data.frame(x=xs, y=cumsum((1/N)*h(xs)))
plt_3.a = ggplot() +
  geom_line(data=df_3.a, aes(x=x, y=y), color="coral") +
  geom_line(data=df_3.a.2, aes(x=x, y=y), color="steelblue") +
  labs(title=TeX("$\\frac{1}{1+x}$ and its antiderivative"))

plt_3.a
```

The following code is used in Question 3 part (b)

```
cc = function(x){
  if(x<0){return(0)}
  else if(x>1){return(0)}
  else{
    return(1+x)
  }
}
cc = Vectorize(cc)

b = -cov(h(xs), cc(xs)) / var(cc(xs))
i2 = mean(h(us)) + b*(mean(cc(us)) - 1.5)
```

The following code is used in Question 3 part (c)


```
## Parameterize

M = 300

## Simulate

i1s = rep(NA, M)
i2s = rep(NA, M)

for(i in 1:M){
  us = runif(N)
  i1 = mean(h(us))
  i2 = mean(h(us)) + b*(mean(cc(us)) - 1.5)
  i1s[i] = i1
  i2s[i] = i2
}

v1 = var(i1s)
v2 = var(i2s)
```

The following code is used in Question 5 part (a)

```
## Parameterize

n = 100
ltrue = 2
ptrue = 0.3

## Sample from prior

r = runif(n) < ptrue # r Bernoulli(p)

## Sample from posterior

xs = r * rpois(n, ltrue)

## Plot

df = data.frame(x=xs)
plt = ggplot(data=df, aes(x=x)) +
  geom_histogram(bins = max(xs)+1) +
  labs(
    x="Value",
    y="Frequency",
    title=TeX(
      "$x_i$ |  $\text{Pr}\{r\}$ ,  $\lambda$ ,  $p$  when  $p=0.3$  and  $\lambda=2$ "
    )
  )
```

The following code is used in Question 5 part (c)

```

## Parameterize

N = 400
A = seq(1/2, 2, by=1/2)
B = A

## Generate data frame

df.main = data.frame(matrix(NA,
                             length(A)*length(B)*N,
                             (5+n)
                             ))
names(df.main) = c("k", "p", "l", "a", "b", paste("r", 1:n, sep=""))

z = 0
for(a in A){for(b in B){

## Generate lambda, r, p randomly

r0 = runif(n) > rbeta(n, a, b)
l0 = ceiling(runif(1)*max(xs))
p0 = rbeta(1, a, b)
r1 = r0
l1 = l0
p1 = p0

Rs = matrix(NA, N, n)
ls = rep(NA, N)
ps = rep(NA, N)
Rs[1,] = r1
ls[1] = l1
ps[1] = p1

## Run the Gibbs sampler

for(i in 2:N){

  r1 = runif(n) < (p1*exp(-l0))/(p1*exp(-l0) + (1-p1)*(xs==0))
  l1 = rgamma(1, shape = a + sum(xs), rate = b + sum(r1))
  p1 = rbeta(1, 1 + sum(r1), n + 1 - sum(r1))

  Rs[i,] = r1
  ls[i] = l1
  ps[i] = p1
}

# Update the main dataframe (k, p, l, a, b, r1, ..., rn)

ixs = (1:N)+z*N
df.main[ixs,][1] = 1:n

```

```

df.main[ixs,][2] = ps    # estimate of p
df.main[ixs,][3] = ls    # estimate of lambda
df.main[ixs,][4] = a
df.main[ixs,][5] = b
df.main[ixs,][6:dim(df.main)[2]] = as.integer(Rs)
z = z + 1

}}

## Plot

plt_lambda = ggplot(data=df.main) +
  geom_line(aes(x=k, y=l), color="steelblue") +
  geom_hline(aes(yintercept=ltrue), color="coral") +
  facet_grid(a ~ b,
    labeller=labeller(
      a = (function (x) paste("a=", x, sep="")),
      b = (function (x) paste("b=", x, sep=""))
    )) +
  labs(x="Gibbs sample iteration",
    y=TeX("$\\lambda_i$"),
    title=TeX(paste(
      "Gibbs sampler convergence for $\\lambda$",
      "using prior $f(p)\\sim\\beta(a,b)(p)$"
    )))

plt_p = ggplot(data=df.main) +
  geom_line(aes(x=k, y=p), color="steelblue") +
  geom_hline(aes(yintercept=ptrue), color="coral") +
  facet_grid(a ~ b,
    labeller=labeller(
      a = (function (x) paste("a=", x, sep="")),
      b = (function (x) paste("b=", x, sep=""))
    )) +
  labs(x="Gibbs sample iteration",
    y=TeX("$p_i$"),
    title=TeX(paste(
      "Gibbs sampler convergence for $p$",
      "using prior $f(p)\\sim\\beta(a,b)(p)$"
    )))

plt_lambda
plt_p

```

The following code is used in Question 6

```
## Parameterize

N = 1000
ran = seq(0, 10, by=0.01)
t1 = 1.5
t2 = 2

## Define functions

f = function(z){

  if(z<=0){return(0)}
  else if(t1<=0){return(0)}
  else if(t2<=0){return(0)}

  # Density
  r = z^(-3/2)*exp(-t1*z-t2/z+2*sqrt(t1*t2)+log(sqrt(2*t2)))
  return(r)
}
f = Vectorize(f)

g = function(z){
  # Envelope density g(z)

  return(dgamma(z, 1, 1))
}
g = Vectorize(g)

r = function(x,y){
  return(min(f(y)/f(x)*g(x)/g(y),1))
}

## Metropolis-Hastings algorithm

samp = rep(NA, N)
envelope = g(ran)
z0 = sample(size=1, x=ran, prob=envelope)
samp[1] = z0

for(i in 2:N){
  z1 = sample(size=1, x=ran, prob=envelope)
  a = r(z0,z1)
  if(a == 1){

    z0 = z1
  }else{

    if(a > runif(1)){

      z0 = z1
    }
  }
}
```

```
}
  samp[i] = z0
}

## Plot

dists_df = data.frame(x=ran, f=f(ran), g=g(ran))
plt_dists = ggplot(data=melt(dists_df, id="x")) +
  geom_line(aes(x=x, y=value, color=variable))

plt_methast = ggplot() +
  geom_histogram(
    data=melt(samp),
    aes(x=value, y=..density..),
    alpha=0.8, bins=N/50
  ) +
  geom_line(
    data=melt(dists_df, id="x"),
    aes(x=x, y=value, color=variable)
  ) +
  scale_colour_discrete(
    labels=c("f(z)", "g(z)"),
    name="Distribution"
  ) +
  labs(
    x="z", y="Frequency of z",
    title=TeX(paste(
      "Target  $f(z)$ ",
      "envelope  $g(z)$ ",
      "and Metropolis-Hastings sample",
      sep=" "
    ))
  )

plt_methast
```