Homework 6 - STAT 5362 Statistical Computing

*Sen Yang**

20 October 2018

Abstract

This is homework 6 for STAT 5362 - Statistical Computing.

Contents

*sen.2.yang@uconn.edu; M.S. student at Department of Statistics, University of Connecticut.

1 Rejection sampling

1.1 Show that g is a mixture of Gamma distributions

g is a probability density on $(0, \infty)$, such that

$$g(x) \propto (2x^{\theta-1} + x^{\theta-1/2})e^{-x}.$$

With a normalizing constant C, we have

$$C \int_0^\infty (2x^{\theta-1} + x^{\theta-1/2})e^{-x}dx = 1.$$

Therefore,

$$\begin{aligned}
&C \int_0^\infty (2x^{\theta-1} + x^{\theta-1/2})e^{-x}dx \\
&= 2C \int_0^\infty 2x^{\theta-1}e^{-x}dx + C \int_0^\infty x^{\theta+1/2-1}e^{-x}dx \\
&= 2C\Gamma(\theta) + C\Gamma(\theta + \frac{1}{2}) \\
&\implies C = \frac{1}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})}
\end{aligned}$$

Let us assume the mixture of the Gamma distribution would like

$$g(x) = c_1\Gamma(\alpha_1, \beta_1) + c_2\Gamma(\alpha_2, \beta_2)$$

Obviously, $\beta_1 = \beta_2 = 1$, and we have

$$\begin{aligned}
g(x) &= c_1\Gamma(\alpha_1, 1) + c_2\Gamma(\alpha_2, 1) \\
&= \frac{c_1}{\Gamma(\alpha_1)}x^{\alpha_1-1}e^{-x} + \frac{c_2}{\Gamma(\alpha_2)}x^{\alpha_2-1}e^{-x} \\
&= C(2x^{\theta-1} + x^{\theta-1/2})e^{-x} \\
&\implies \alpha_1 = \theta \qquad \alpha_2 = \theta + \frac{1}{2} \\
&\qquad \frac{c_1}{\Gamma(\alpha_1)} = 2C \qquad \frac{c_2}{\Gamma(\alpha_2)} = C \\
&\implies c_1 = \frac{2\Gamma(\theta)}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})} \\
&\qquad c_2 = \frac{\Gamma(\theta + \frac{1}{2})}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})}
\end{aligned}$$

It means g is a mixture of $\Gamma(\theta, 1)$ and $\Gamma(\theta + \frac{1}{2}, 1)$ with corresponding weight $c_1 = \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+\frac{1}{2})}$, $c_2 = \frac{\Gamma(\theta+\frac{1}{2})}{2\Gamma(\theta)+\Gamma(\theta+\frac{1}{2})}$, respectively.

1.2 Sample from g

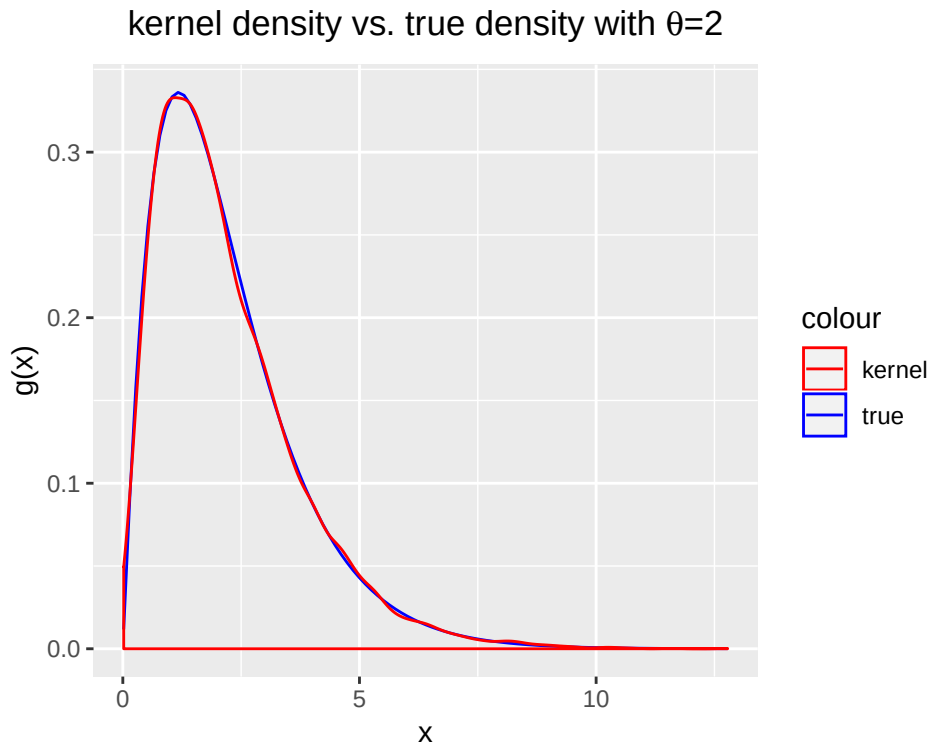
The algorithm to generate a sample from g:

Algorithm 1 Generate a sample from g

Generate a sample $U \sim U(0,1)$
if $U < c_1$ **then**
  $X = \Gamma(\theta, 1)$
else
  $X = \Gamma(\theta + \frac{1}{2}, 1)$
end if
return X.

Then, we realize this algorithm in R.

```
# Sample from g
## function to generate a sample from g
g_sample <- function(n,theta) {
  c1 <- 2*gamma(theta)/(2*gamma(theta)+gamma(theta+0.5))
  uni <- runif(n)
  samples <- data.frame(spl=matrix(0,n,1))
  samples[which(uni<c1),1] <- rgamma(length(which(uni<c1)),shape = theta,scale = 1)
  samples[which(uni>=c1),1] <- rgamma(length(which(uni>=c1)),shape = theta+0.5,scale = 1)
  samples
}
## generate a sample with theta = 2
g_sample1 <- g_sample(10000,2)

## ture density
trued_g <- function (x,theta) {
  c1 <- 2*gamma(theta)/(2*gamma(theta)+gamma(theta+0.5))
  trued_g <- c1*dgamma(x,shape=theta,scale=1)+(1-c1)*dgamma(x,shape=theta+0.5,scale=1)
  trued_g
}
## plot
library(ggplot2)
ggplot(g_sample1) + stat_function(aes(col="true"), fun=trued_g, args = list(theta=2)) +
  geom_density(aes(spl, col='kernel')) + scale_color_manual(values = c("red", "blue")) +
  labs(title = expression(paste("kernel density vs. true density with ", theta, "=2")),
       x = "x", y = "g(x)") + theme(plot.title = element_text(hjust = 0.5))
```



1.3 Sample from f

To reduce the expected number of iterations, α should be small.

$$\begin{aligned}\alpha &= \sup_{x>0} \frac{q(x)}{g(x)} \\ &= \sup_{x>0} \frac{\sqrt{4+x}x^{\theta-1}e^{-x}}{C(2x^{\theta-1}+x^{\theta-1/2})e^{-x}} \\ &= \frac{1}{C}\sup_{x>0}\frac{\sqrt{4+x}}{(2+\sqrt{x})}\end{aligned}$$

Since $(\frac{\sqrt{4+x}}{2+\sqrt{x}})' = \frac{2\sqrt{x}-4}{2\sqrt{4x+x^2}(2+\sqrt{x})^2}$, this is a convex function that is decreasing over $(0,4)$ and increasing over $(4,\infty)$. Therefore,

$$sup_{x>0}(\frac{\sqrt{4+x}}{2+\sqrt{x}}) = \max\{\lim_{x\to 0}\frac{\sqrt{4+x}}{2+\sqrt{x}}, \lim_{x\to\infty}\frac{\sqrt{4+x}}{2+\sqrt{x}}\} = 1$$

$$\Longrightarrow \alpha = \sup_{x>0}\frac{q(x)}{g(x)} = \frac{1}{C}$$

The algorithm to generate a sample from f:

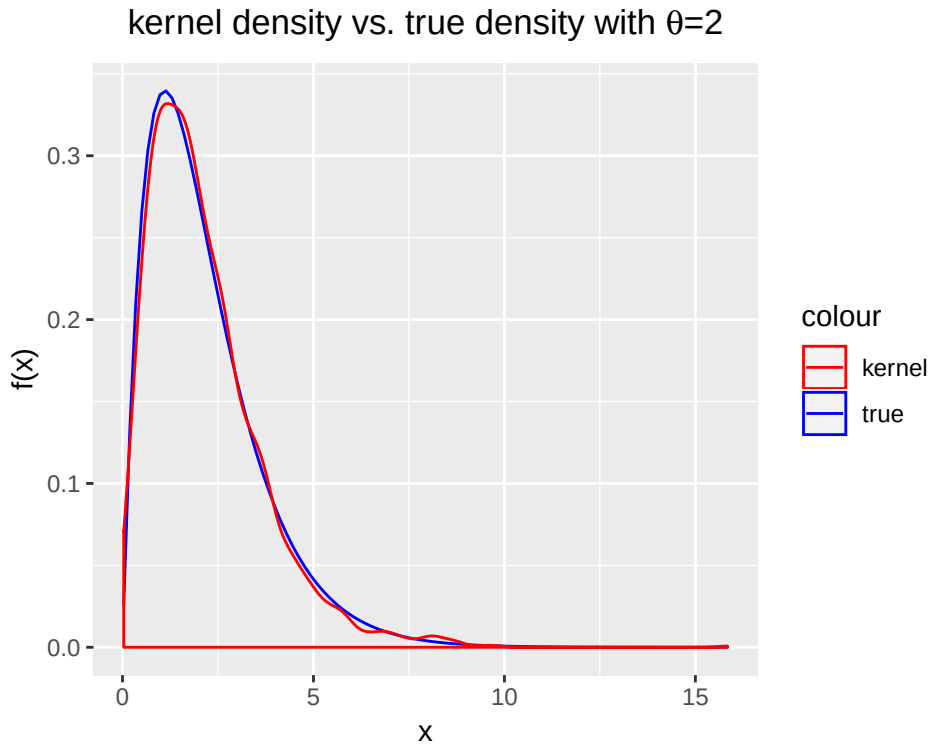
Algorithm 2 Generate a sample from f

Step 1: Generate a sample from $X \sim g$ and $U \sim U(0,1)$
Step 2:
if $U > \frac{q(x)}{\alpha g(x)}$ **then**
  Go to step 1.
else
  Return X.
end if

Then, we realize this algorithm in R.

```
# sample from f
## generate a sample from f
f_sample <- function(n,theta) {
  g_sample <- g_sample(n,theta)
  u_sample <- runif(n)
  c <- 1/(2*gamma(theta)+gamma(theta+0.5))
  f_sample <- matrix(0,n,1)
  for (i in 1:n) {
    qx <- sqrt(4 + g_sample[i,1])*g_sample[i,1]^(theta-1)*exp(-g_sample[i,1])
    gx <- (2*g_sample[i,1]^(theta-1)+g_sample[i,1]^(theta-0.5))*exp(-g_sample[i,1])
    if (u_sample[i] > qx*c/gx) next
      else {f_sample[i,1]<-g_sample[i,1]}
  }
  f_sample<-as.data.frame(f_sample[-which(f_sample==0),1])
  colnames(f_sample)<-"f"
  f_sample
}

f_sample1<-f_sample(10000,2)

qx<- function(x, theta) {
  qx<-sqrt(4 + x)*x^(theta-1)*exp(-x)
  qx
}
trued_f <- function (x,theta) {
  intgrl<-integrate(function(x) qx(x,theta),0,Inf)
  c_prime <-1/intgrl[["value"]]
  trued_f <- c_prime*sqrt(4 + x)*x^(theta-1)*exp(-x)
  trued_f
}

## plot
library(ggplot2)
ggplot() + stat_function(aes(col="true"), fun=trued_f, args = list(theta=2)) +
  geom_density(data=f_sample1,aes(f,col="kernel")) +
  scale_color_manual(values = c("red","blue")) +
```

```
  labs(title = expression(paste("kernel density vs. true density with ", theta, "=2")),
       x = "x", y = "f(x)") + theme(plot.title = element_text(hjust = 0.5))
```



2 Mixture Proposal

2.1 Generate a sample using a mixture of beta distribution as the instrumental density

Since $0 < x < 1$, $0 < x^2 < 1$. Then, we have

$$q(x) = \frac{x^{\theta-1}}{1+x^2} + \sqrt{2+x^2}(1-x)^{\beta-1} < x^{\theta-1} + \sqrt{3}(1-x)^{\beta-1} = h(x).$$

Therefore, we choose

$$h(x) = x^{\theta-1} + \sqrt{3}(1-x)^{\beta-1} \qquad with \quad \alpha = 1.$$

where $g(x) = Ch(x) = C(x^{\theta-1} + \sqrt{3}(1-x)^{\beta-1}) = p_1\theta x^{\theta-1} + p_2\beta(1-x)^{\beta-1} \sim p_1 Beta(\theta, 1) + p_2 Beta(1, \beta)$.

Then,

$$
\begin{aligned}
g(x) &= \int_0^1 C(x^{\theta-1} + \sqrt{3}(1-x)^{\beta-1})dx \\
&= \frac{C}{\theta}x^{\theta} - \frac{\sqrt{3}C}{\beta}(1-x)^{\beta}|_{x=0}^{1} \\
&= \frac{C}{\theta} + \frac{\sqrt{3}C}{\beta} \\
&= 1 \\
\Longrightarrow C &= \frac{\theta\beta}{\beta + \sqrt{3}\theta} \\
\Longrightarrow p_1 &= \frac{\beta}{\beta + \sqrt{3}\theta} \qquad p_2 = \frac{\sqrt{3}\theta}{\beta + \sqrt{3}\theta}
\end{aligned}
$$

The algorithm to generate a sample from f:

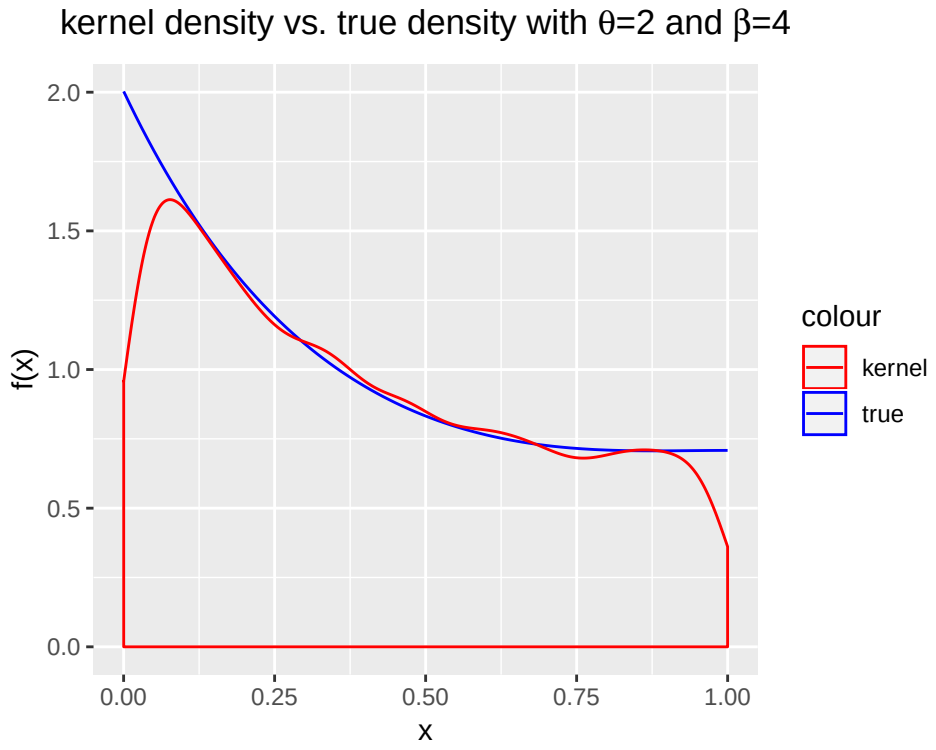
Algorithm 3 Generate a sample from f

Step 1: Generate a sample from g(x)
Generate a sample $U \sim U(0,1)$
if $U < p_1$ **then**
  $X = Beta(\theta, 1)$
else
  $X = Beta(1, \beta)$
end if
return X.
Step 2: Generate a sample from $X \sim g$ and $U \sim U(0,1)$
Step 3:
if $U > \frac{q(x)}{\alpha h(x)} = \frac{q(x)}{h(x)}$ **then**
  Go to step 2.
else
  Return X.
end if

Then, we realize this algorithm in R.

```r
# Sample from g
## function to generate a sample from g
g_sample <- function(n,theta,beta) {
  p1 <- beta/(beta+sqrt(3)*theta)
  uni <- runif(n)
  samples <- data.frame(spl=matrix(0,n,1))
  samples[which(uni<p1),1] <- rbeta(length(which(uni<p1)),shape1 = theta, shape2 = 1)
  samples[which(uni>=p1),1] <- rbeta(length(which(uni>=p1)),shape1 = 1, shape2 = beta)
  samples
}
```

```r
# sample from f
## generate a sample from f
f_sample <- function(n,theta,beta) {
  g_sample <- g_sample(n,theta,beta)
  u_sample <- runif(n)
  f_sample <- matrix(0,n,1)
  for (i in 1:n) {
    qx <- g_sample[i,1]^(theta-1)/(1+g_sample[i,1]^2)+
      sqrt(2+g_sample[i,1]^2)*(1-g_sample[i,1])^(beta-1)
    hx <- g_sample[i,1]^(theta-1)+3^0.5*(1-g_sample[i,1])^(beta-1)
    if (u_sample[i] > qx/hx) next
    else {f_sample[i,1]<-g_sample[i,1]}
  }
  f_sample<-as.data.frame(f_sample[-which(f_sample==0),1])
  colnames(f_sample)<-"f"
  f_sample
}

f_sample1<-f_sample(10000,2,4)

qx<- function(x, theta, beta) {
  qx<-x^(theta-1)/(1+x^2)+sqrt(2+x^2)*(1-x)^(beta-1)
  qx
}
trued_f <- function (x,theta,beta) {
  intgrl<-integrate(function(x) qx(x,theta,beta),0,1)
  c_prime <-1/intgrl[["value"]]
  trued_f <- c_prime*(x^(theta-1)/(1+x^2)+sqrt(2+x^2)*(1-x)^(beta-1))
  trued_f
}

## plot
library(ggplot2)
ggplot() + stat_function(aes(col="true"), fun=trued_f, args = list(theta=2, beta=4)) +
  geom_density(data=f_sample1,aes(f,col="kernel")) +
  scale_color_manual(values = c("red","blue")) +
  labs(title = expression(paste("kernel density vs. true density with ", theta,
                                "=2 and ", beta, "=4")),
       x = "x", y = "f(x)") + theme(plot.title = element_text(hjust = 0.5))
```



2.2 Generate a sample using individual Beta distributions

Let

$$q_1(x) = \frac{x^{\theta-1}}{1+x^2} \qquad q_2(x) = \sqrt{2+x^2}(1-x)^{\beta-1}$$

$$h_1(x) = x^{\theta-1} \qquad h_2(x) = \sqrt{3}(1-x)^{\beta-1} \qquad with\ \alpha_1 = \alpha_2 = 1.$$

Therefore, we can get corresponding $g(x)$.

$$g_1(x) = \theta x^{\theta-1} \qquad g_2(x) = \beta(1-x)^{\beta-1}$$

The algorithm to generate a sample from f:

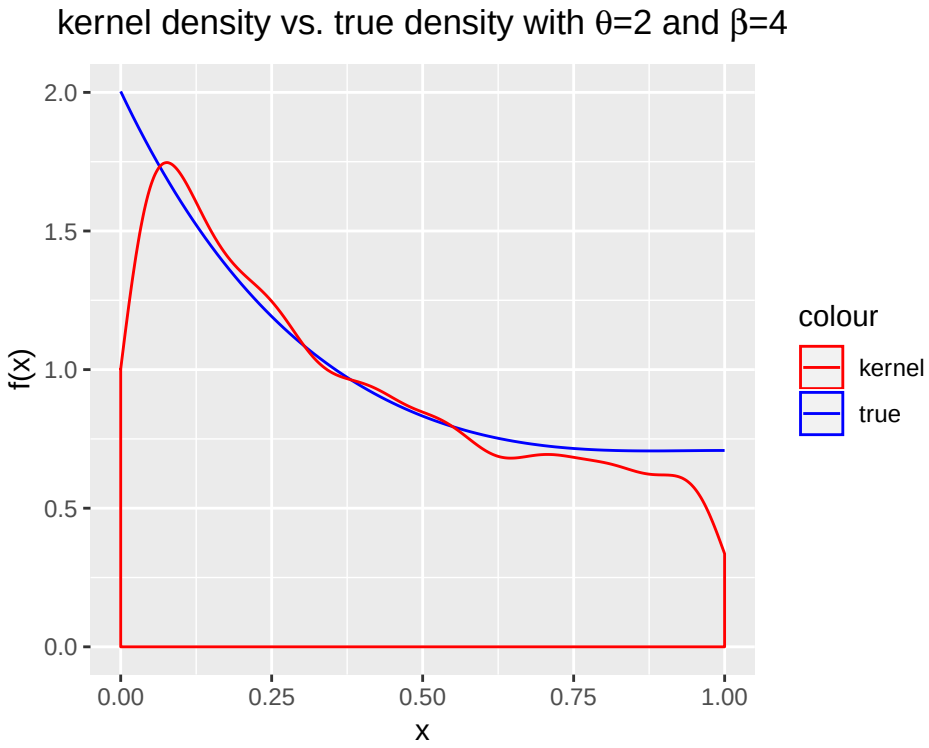
Algorithm 4 Generate a sample from f

Step 1: Generate a sample $U \sim U(0,1)$
Step 2:
if $U < \frac{\alpha_1}{\alpha_1+\alpha_2} = 0.5$ **then**
  Sample from $X \sim g_1$ and $U_1 \sim U(0,1)$
  Step 3:
  if $U > \frac{q_1(x)}{\alpha_1 h_1(x)} = \frac{q_1(x)}{h_1(x)}$ **then**
    Go to Step 3.
  else
    Return X.
  end if
else
  Sample from $X \sim g_2$ and $U_2 \sim U(0,1)$
  Step 4:
  if $U > \frac{q_2(x)}{\alpha_2 h_2(x)} = \frac{q_2(x)}{h_2(x)}$ **then**
    Go to Step 4.
  else
    Return X.
  end if
end if

Then, we realize this algorithm in R.

```
# Separate Beta

# Sample from g
## function to generate a sample from g
g_sample1 <- function(n,theta,beta) {
  g_sample1 <- data.frame(rbeta(n,shape1=theta,shape2=1))
  g_sample1
}

g_sample2 <- function(n,theta,beta) {
  g_sample2 <- data.frame(rbeta(n,shape1=1,shape2=beta))
  g_sample2
}

# sample from f
## generate a sample from f
f_sample_sep <- function(n,theta,beta) {
  u_sample <- runif(n)
  n1 <- length(which(u_sample<0.5))
  g_sample1 <- g_sample1(n1,theta,beta)
  u_sample1 <- runif(n1)
  f_sample1 <- matrix(0,n1,1)
  for (j in 1:n1) {
    qx1 <- g_sample1[j,1]^(theta-1)/(1+g_sample1[j,1]^2)
```

```
    hx1 <- g_sample1[j,1]^(theta-1)
    if (u_sample1[j] > qx1/hx1) next
     else {f_sample1[j,1]<-g_sample1[j,1]}
   }
  n2 <- length(which(u_sample>=0.5))
  g_sample2 <- g_sample2(n2,theta,beta)
  u_sample2 <- runif(n2)
  f_sample2 <- matrix(0,n2,1)
  for (k in 1:n2) {
    hx2 <- 3^0.5*(1-g_sample2[k,1])^(beta-1)
    qx2 <- sqrt(2+g_sample2[k,1]^2)*(1-g_sample2[k,1])^(beta-1)
    if (u_sample2[k] > qx2/hx2) next
    else {f_sample2[k,1]<-g_sample2[k,1]}
    }
  f_sample<-rbind(f_sample1,f_sample2)
  f_sample<-as.data.frame(f_sample[-which(f_sample==0),1])
  colnames(f_sample)<-"f"
  f_sample
}

f_sample_sep<-f_sample_sep(10000,2,4)

## plot
library(ggplot2)
ggplot() + stat_function(aes(col="true"), fun=trued_f, args = list(theta=2, beta=4)) +
  geom_density(data=f_sample_sep,aes(f,col="kernel")) +
  scale_color_manual(values = c("red","blue")) +
  labs(title = expression(paste("kernel density vs. true density with ", theta,
                                "=2 and ", beta, "=4")),
       x = "x", y = "f(x)") + theme(plot.title = element_text(hjust = 0.5))
```


kernel density vs. true density with θ=2 and β=4
2.0
1.5
1.0
0.5
0.0
f(x)
0.00
0.25
0.50
0.75
1.00
x
colour
kernel
true