STA 531 HW2 (Sampling)

Daniel Truver

2/8/2018

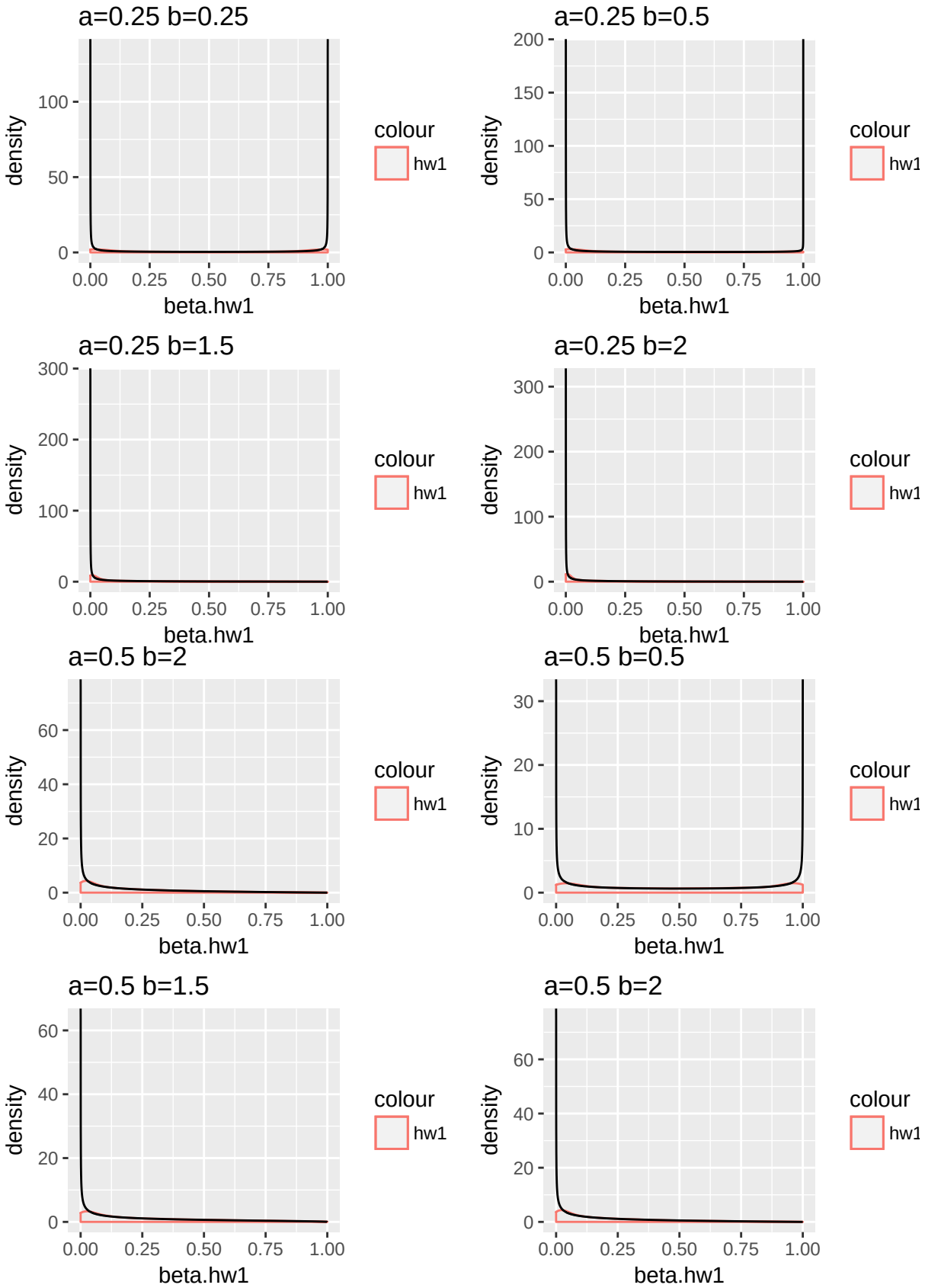
(1) Comparison of Beta R.V. Generators

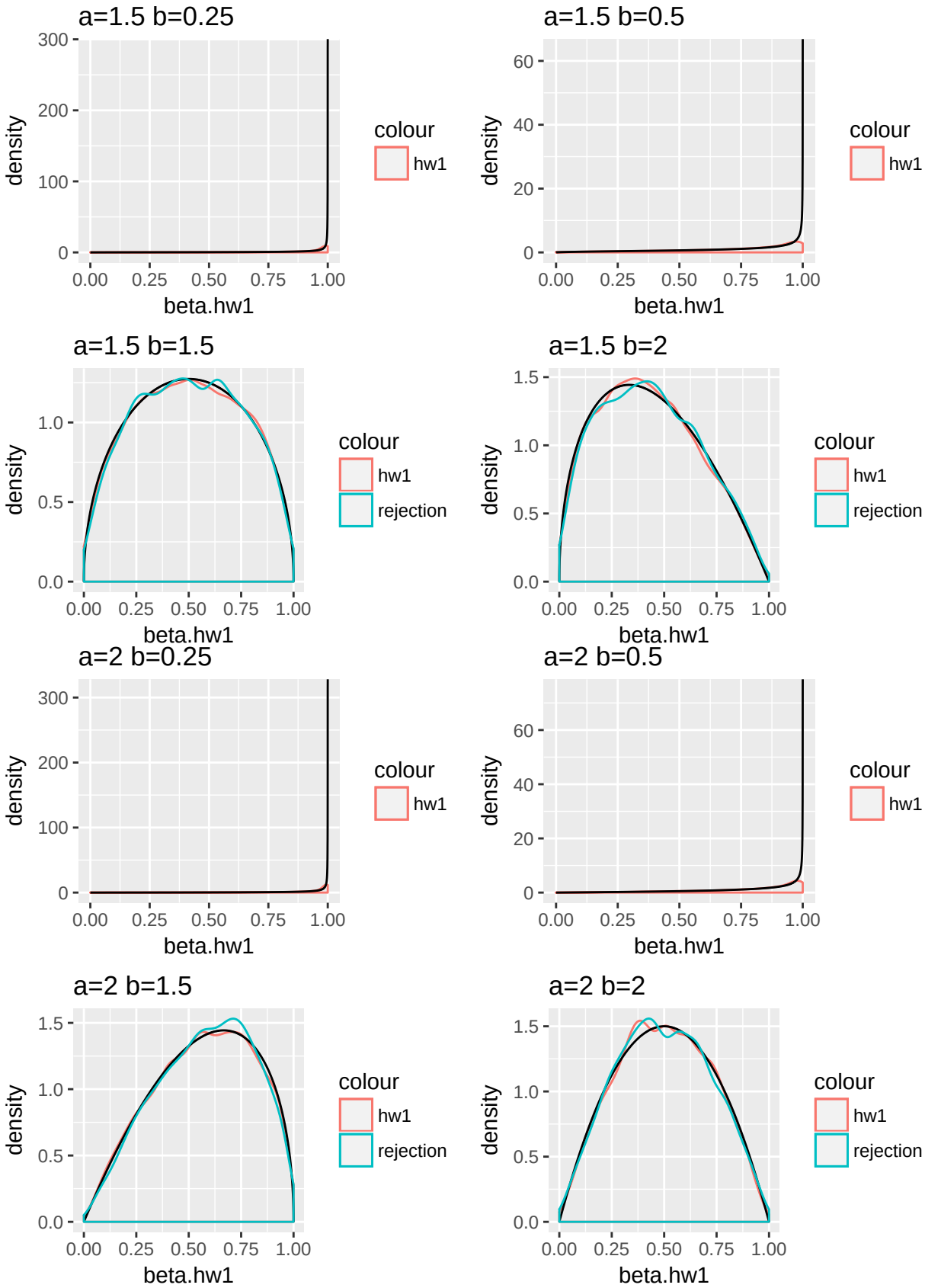
```
hw1.beta = function(alpha, beta, nsim = 10000){
  g_1 = rgamma(nsim, alpha, 1)
  g_2 = rgamma(nsim, beta, 1)
  B = g_1/(g_1 + g_2)
  return(B)
}
library(ggplot2)
```

(a) Rejection Sampling (g = unif(0,1))

```
alpha_seq = c(.25,.5,1.5,2)
beta_seq = c(.25,.5,1.5,2)
n.sim = 10000
x = seq(0,1,length.out = n.sim)
plot.list = list()
i = 1
set.seed(2018)
for (a in alpha_seq){
  for (b in beta_seq){
    target.beta = dbeta(x,a,b)
    beta.hw1 =hw1.beta(alpha = a, beta = b, nsim = n.sim)
    c = max(target.beta) + 1
    x.star = runif(n.sim,0,1)
    U = runif(n.sim, 0, 1)
    x.star[!(U < dbeta(x.star,a,b)/c)] = NA
    g = ggplot(data = data.frame(x, target.beta, x.star, beta.hw1)) +
      geom_density(aes(beta.hw1, color = "hw1")) +
      geom_line(aes(x, target.beta)) +
      geom_density(aes(x.star, color = "rejection")) +
      ggtitle(paste0("a=", a, " b=",b))
    plot.list[[i]] = g
    i = i + 1
  }
}
```

Let's see visually how these perform in comparison to each other and the true distribution function.


a=0.25 b=0.25
density
beta.hw1
colour
hw1
a=0.25 b=0.5
density
beta.hw1
colour
hw1
a=0.25 b=1.5
density
beta.hw1
colour
hw1
a=0.25 b=2
density
beta.hw1
colour
hw1
a=0.5 b=2
density
beta.hw1
colour
hw1
a=0.5 b=0.5
density
beta.hw1
colour
hw1
a=0.5 b=1.5
density
beta.hw1
colour
hw1
a=0.5 b=2
density
beta.hw1
colour
hw1

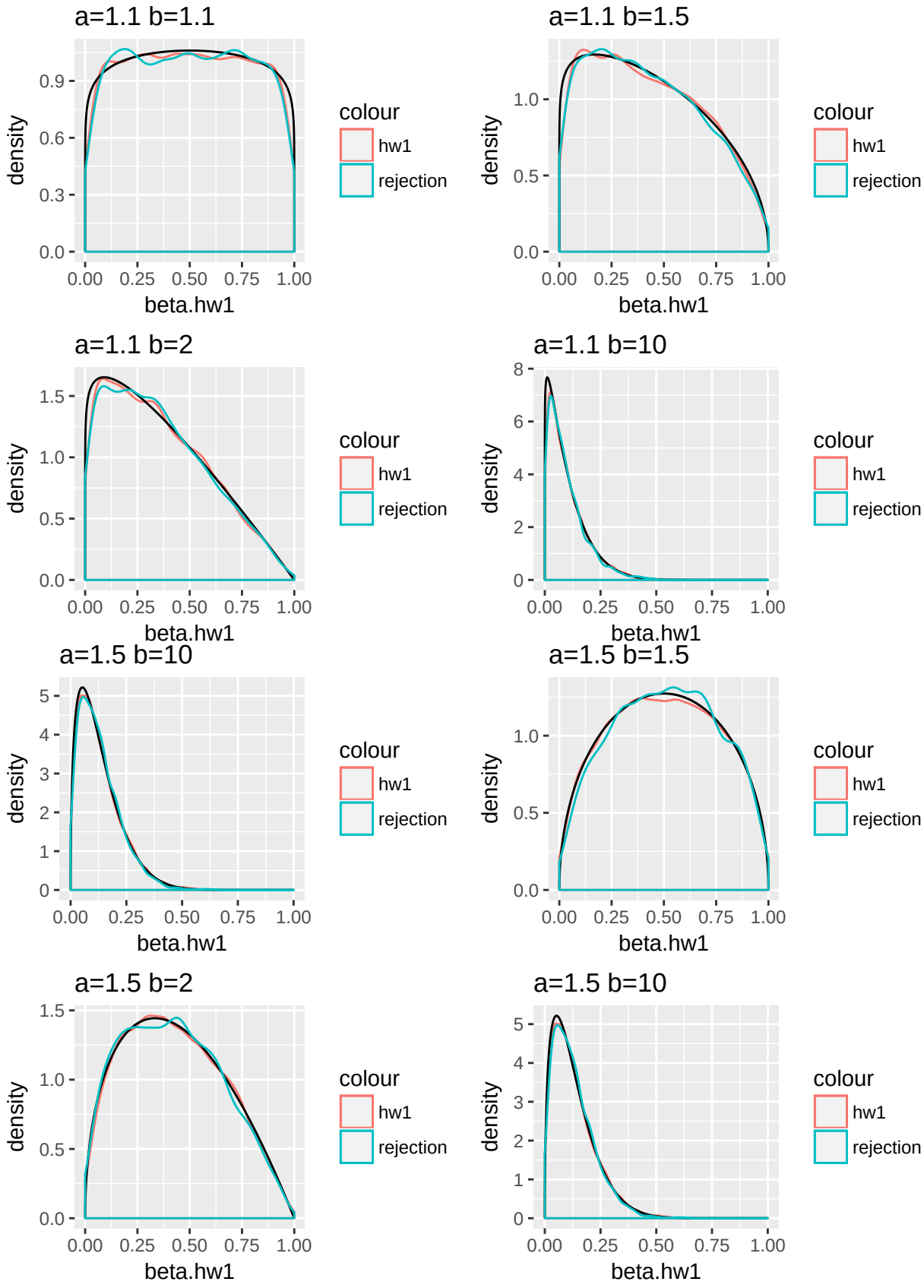


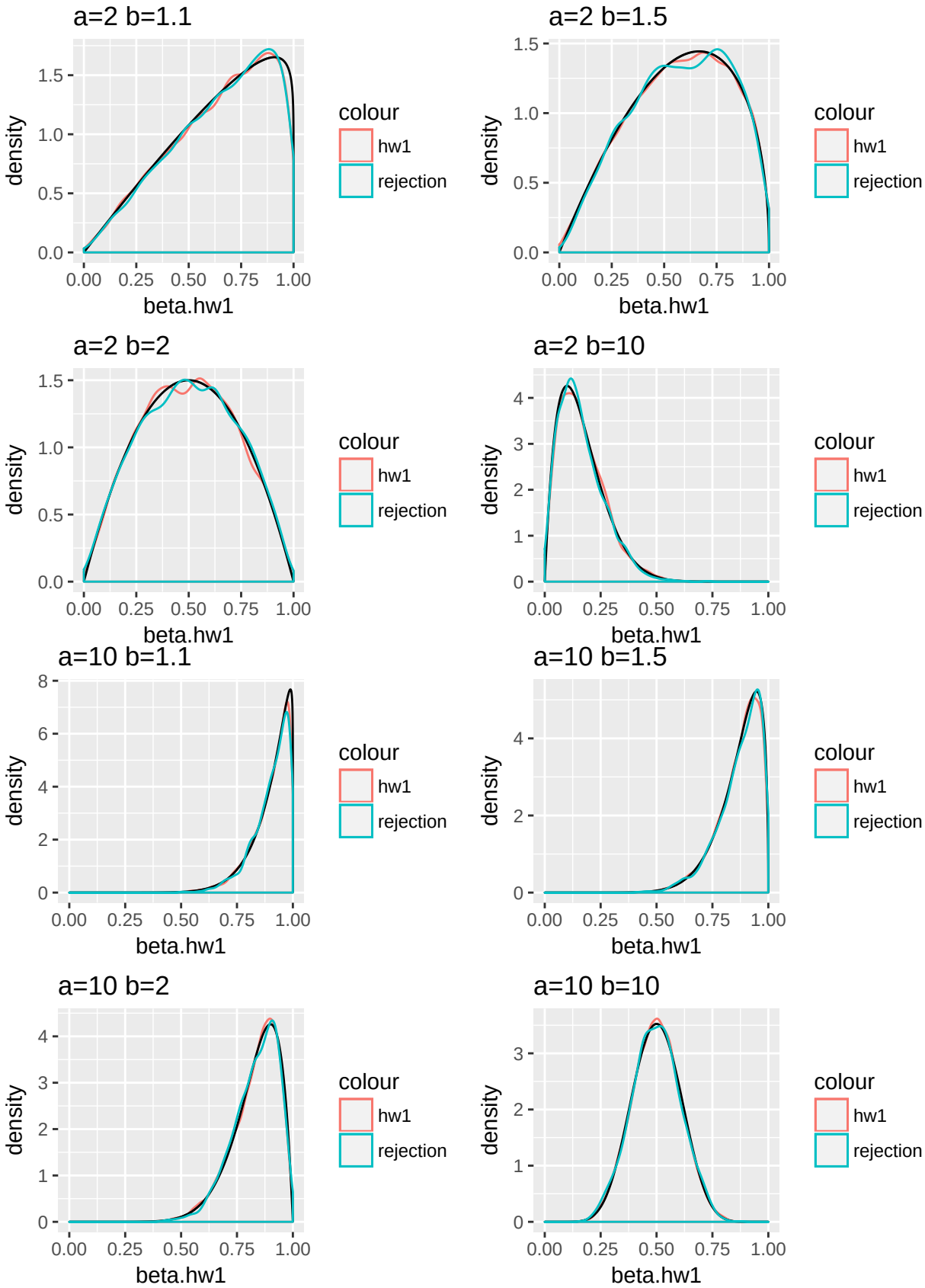
We see from these plots that rejection sampling the method of gamma random variables from homework 1

have similar performance when $\alpha \geq 0, \beta \geq 0$. For $alpha \leq 0, \beta \leq 0$, the c we need to pick for the uniform to dominate the beta distribution is simply too large and there are not enough accepted values of x^*, just constant rejection. In these cases, the gamma method has a better approximation to the distribution, but it still fails to capture behavior near 0 and 1.

(b) Rejection Sampling (g = truncated normal)

```
library(truncnorm)
alpha_seq = c(1.1,1.5,2,10)
beta_seq = c(1.1,1.5,2,10)
n.sim = 10000
x = seq(0,1,length.out = n.sim)
plot.list = list()
i = 1
set.seed(2018)
for (a in alpha_seq){
  for (b in beta_seq){
    target.beta = dbeta(x,a,b)
    beta.hw1 =hw1.beta(alpha = a, beta = b, nsim = n.sim)
    c = max(target.beta/dtruncnorm(x,0,1,.5,1)) + 1
    x.star = rtruncnorm(n.sim, a = 0, b = 1, mean = a/(a+b), sd = 1)
    U = runif(n.sim, 0, 1)
    x.star[!(U < dbeta(x.star,a,b)/(c*dtruncnorm(x.star,a=0,b=1,mean = .5, sd = 1)))] = NA
    g = ggplot(data = data.frame(x, target.beta, x.star, beta.hw1)) +
      geom_density(aes(beta.hw1, color = "hw1")) +
      geom_line(aes(x, target.beta)) +
      geom_density(aes(x.star, color = "rejection")) +
      ggtitle(paste0("a=", a, " b=",b))
    plot.list[[i]] = g
    i = i + 1
  }
}
```


a=1.1 b=1.1
a=1.1 b=1.5
a=1.1 b=2
a=1.1 b=10
a=1.5 b=10
a=1.5 b=1.5
a=1.5 b=2
a=1.5 b=10
density
beta.hw1
colour
hw1
rejection
0.00
0.25
0.50
0.75
1.00



Once more we see that both the proven method from the last homework and the rejection sampling approximate

the beta distribution fairly well when $\alpha \geq 0, \beta \geq 0$. In both cases, 10^{4} draws takes less than a second with decent results.

(2) Proofs

(a) Acceptance Probability

$$\begin{aligned} Pr\left(U \leq \frac{\pi(x)}{cg(x)}\right) &= \int_X Pr\left(U \leq \frac{\pi(x)}{cg(x)} \mid X = x\right) Pr(X = x)dx \\ &= \int_{-\infty}^{\infty} \frac{\pi(x)}{cg(x)} g(x)dx \\ &= \frac{1}{c}\int_{-\infty}^{\infty} \pi(x)dx \\ &= \frac{1}{c} \end{aligned}$$

(b)

From the equality show above, we have

$$c = \frac{1}{Pr[U \leq \pi(x)/(cg(x)]}.$$

A bonafide probability is in the denominator. We know that it is between 0 and 1. c, therefore, satisfies the inequality $c \geq 1$.

(c)

Because the probability of acceptance is $1/c$, we can model the number of trials until k variates are accepted as a negative binomial random variable. The expected number of trials is then $n = ck$.

(d)

I guess an example will suffice to show that this is possible. Suppose π is the beta(2,2) distribution and we want to perform rejection sampling with a \$g = \$ unif(0,1); $c = 3$ and $c' = 1.5 < c$ both satisfy $\pi(x) \leq c'g(x) < cg(x)$. So, here we are, an example that it is possible.

Now for speculation. We may choose c over c' if we suspect that π has flat regions, like a step function, to reduce the acceptance probability in those regions. Perhaps this will allow the algorithm to better explore the space.

(e)

Consider the conditional

$$P\left(X \leq x \mid U \leq \frac{\pi(x)}{cg(x)}\right) = P\left(X \leq x, U \leq \frac{\pi(x)}{cg(x)}\right) / P\left(U \leq \frac{\pi(x)}{cg(x)}\right)$$

Suppose that $\pi(x) > cg(x)$ on some non-neglibigle set $A \subset \text{support}(\pi(x))$.

Then, for $x \in A$,

$$P\left(U \leq \frac{\pi(x)}{cg(x)}\right) = 1.$$

This leaves us with

$$\begin{aligned} P\left(X \leq x \mid U \leq \frac{\pi(x)}{cg(x)}\right) &= P\left(X \leq x, U \leq \frac{\pi(x)}{cg(x)}\right) \\ &= \int_{-\infty}^{x} \int_{0}^{1} g(X) dU dX \\ &= \int_{-\infty}^{x} g(X) dX \\ &= F_g(x) \end{aligned}$$

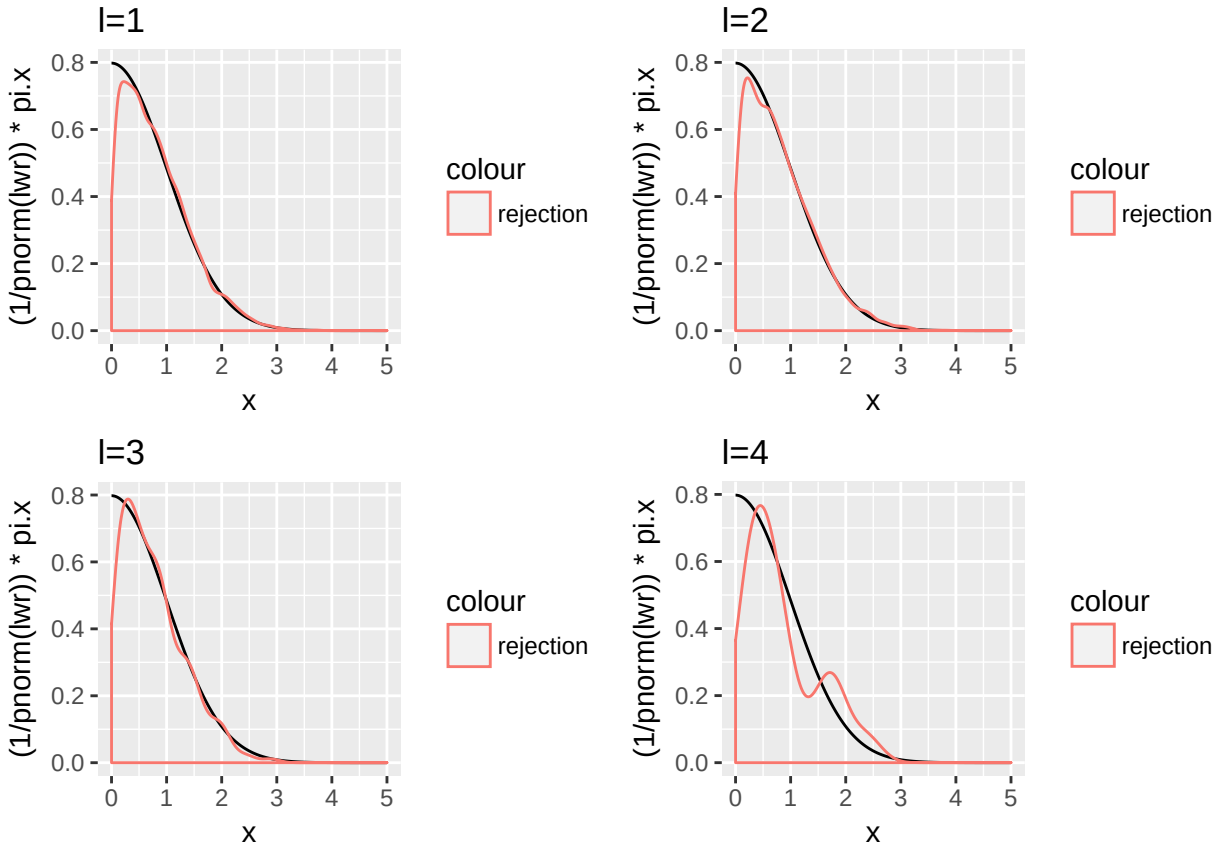
on the set A.

(3) Accept-Reject for N(0,1) Based on Exponential

```
n.sim = 10000
x = seq(0,5,length.out = 1000)
plot.list = list()
i = 1
set.seed(2018)
lwr = 0
lambda = c(1,2,3,4)
for (l in lambda){
  pi.x = dnorm(x,mean = 0,sd = 1)
  g.x = dexp(x, l)
  c = max(pi.x / g.x)
  x.star = rexp(n.sim, l)
  U = runif(n.sim, 0, 1)
  x.star[!( U < dnorm(x.star,0,1)/(c*dexp(x.star,l)) )] = NA
  g = ggplot(data = data.frame(x, pi.x, x.star)) +
    geom_line(aes(x, (1/pnorm(lwr))*pi.x)) +
    geom_density(aes(x.star, color = "rejection")) +
    ggtitle(paste0("l=",l)) +
    xlim(0,5)
  plot.list[[i]] = g
  i = i + 1
}
```

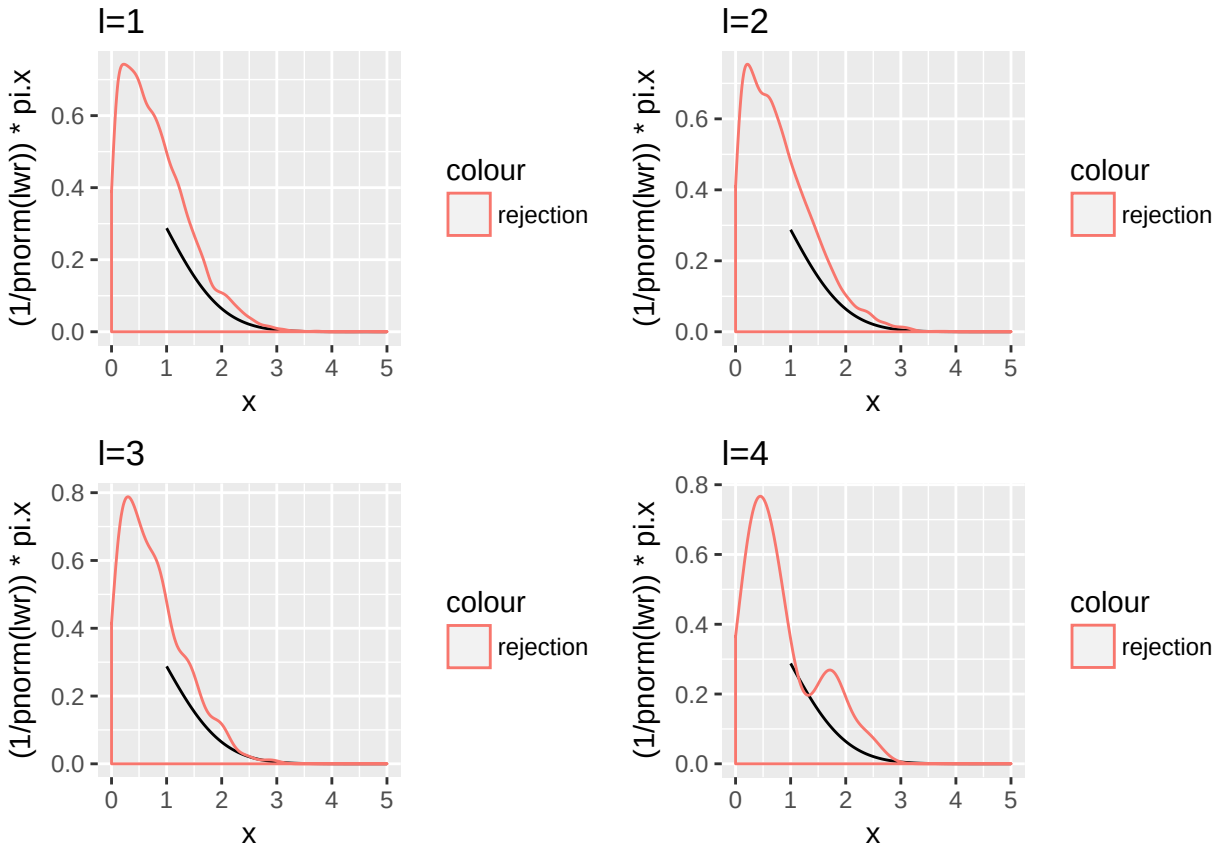
Results on [0, inf)

`ggplot` always plots densities so that they interate to 1, but we only have half the normal distribution, so we multiply it by 2 so as to better evaluate the fit. It seems $\lambda = 1$ has the best performance.



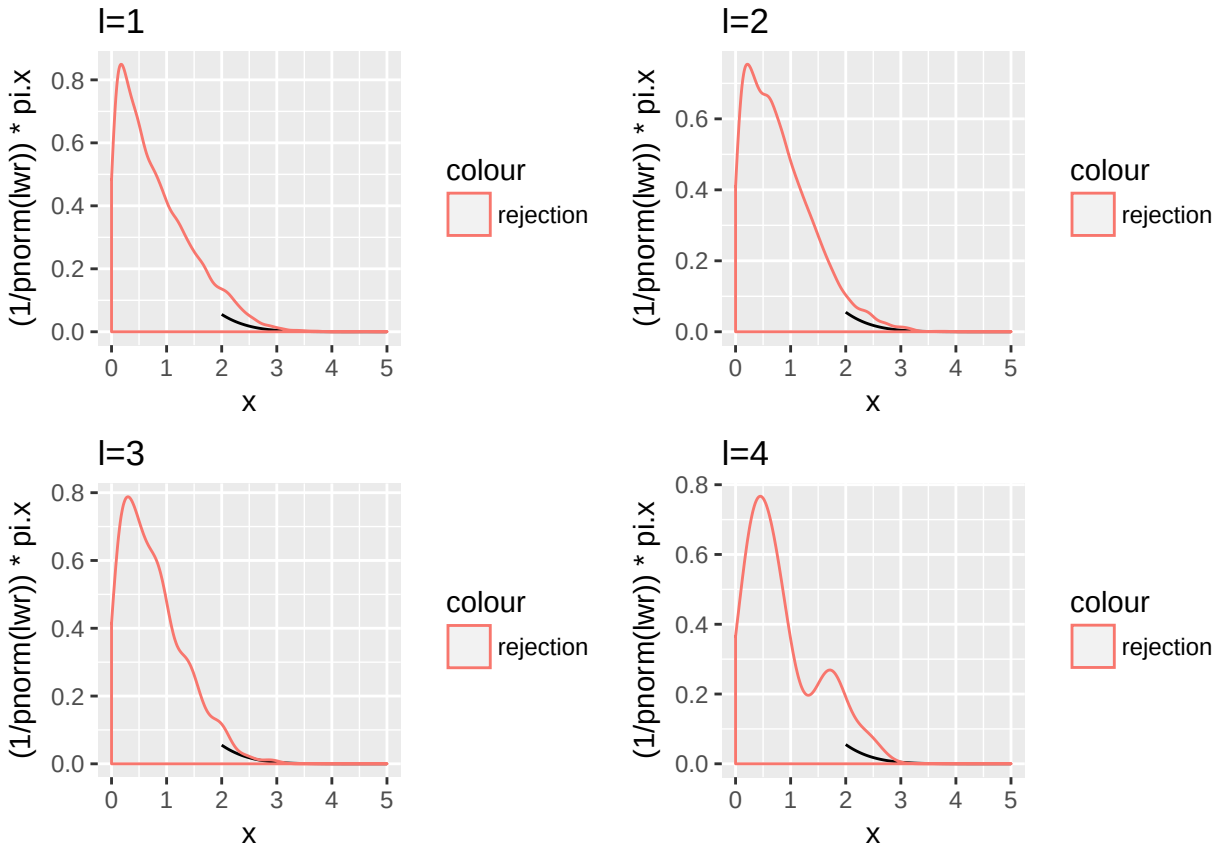
Results on [1, inf)

It seems $\lambda = 3$ has the best performance.



Results on [2, inf)

It seems $\lambda = 2$ has the best performance.



Results on [4, inf)

Honestly, we can't tell anything about performance this far into the tails.