

Lab 5: Rejection Sampling

Hot Hand

The “hot hand” is the purported phenomenon that a person who experiences a successful outcome has a greater chance of success in further attempts. The concept originates from basketball whereas a shooter is allegedly more likely to score if their previous attempts were successful. While previous success at a task can indeed change the psychological attitude and subsequent success rate of a player, researchers for many years did not find evidence for a “hot hand” in practice, dismissing it as fallacious. However, later research questioned whether the belief is indeed a fallacy. In this lab, we will boil down this question as testing whether the results of shots are independent.

Let “1” denote a valid shot and “0” denote an invalid. Suppose we observe the following results of a player:

```
# observations #
set.seed(123)
y <- c(rep(1, 18), rep(0, 3), rep(1, 6), rep(0, 2),
       rbinom(67, 1, prob = 0.25), rep(1, 4))
```

Suppose $Y_i \sim \text{Bernoulli}(\theta)$ and $\theta \sim \text{Beta}(3, 7)$

Find the posterior:

```
# prior #
a <- 3
b <- 7
#posterior #
a_post <- a + sum(y)
b_post <- b + (length(y) - sum(y))
a_post; b_post
```

```
## [1] 50
```

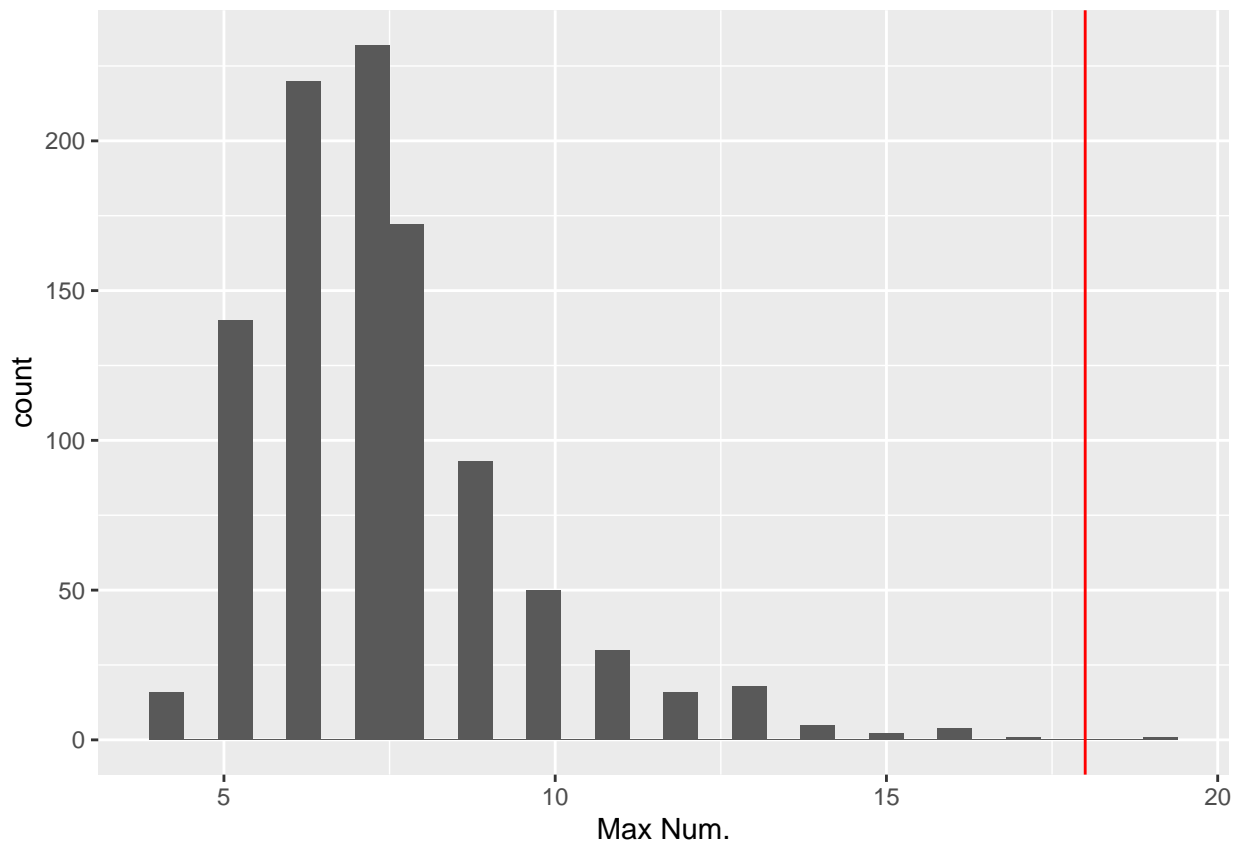
```
## [1] 60
```

Let the test stat. be the maximum number of the same consecutive results. (Other good test stat. might be the fraction of times you see (1, 1) vs (1, 0).)

```
# observed test stat. #
test_stat_obs <- max(rle(y)$lengths)

# test stat. based on simulation #
nsim <- 1000
test_stat_rep <- rep(NA, nsim)
for (i in 1:1000) {
  p_post <- rbeta(1, a_post, b_post)
  y_rep <- rbinom(100, size = 1, prob = p_post)
  test_stat <- max(rle(y_rep)$lengths)
  test_stat_rep[i] <- test_stat
}

ggplot(tibble(test_stat_rep), aes(test_stat_rep)) +
  geom_histogram() + xlab("Max Num.") +
  geom_vline(xintercept = test_stat_obs, colour = "red")
```



In this example, the data doesn't look independent. The “hot-hand” phenomenon might exist.

Next, we'll review rejection sampling.

Why We Need Different Sampling Strategies

- Remember that in Monte Carlo methods, to compute an integral numerically we need to sample from a distribution first. The pdf of this distribution can be of any form as long as it is a legal pdf.
- For some distributions, such as normal, beta, gamma and so on, we can easily do the sampling using built-in functions in R.
- If not, we need to be more clever about how we generate samples. There are two common approaches for sampling from a *univariate* distribution:
 - Inversion Sampling
 - Rejection sampling

Probability Integral Transform

- Suppose that a random variable, Y has a continuous distribution for which CDF is F_Y .
- Then the random variable $U = F_Y(Y)$ has a uniform distribution
 - This is known as the “probability integral transform PIT”
- By taking the inverse of F_Y we have $F_Y^{-1}(U) = Y$

Inversion Sampling

The inverse transform sampling method works as follows:

1. Generate a random number u from $\text{Unif}[0, 1]$
2. Find the inverse of the desired CDF, e.g. $F_Y^{-1}(u)$.
3. Compute $y = F_Y^{-1}(u)$. y is now a sample from the desired distribution.

But for most cases we only know the pdf $f(x)$ instead of cdf $F(x) = \int f(x)dx$. To find cdf, we need to do another integral and we come back to the original problem in Monte Carlo Method again: sampling to approximate the integral.

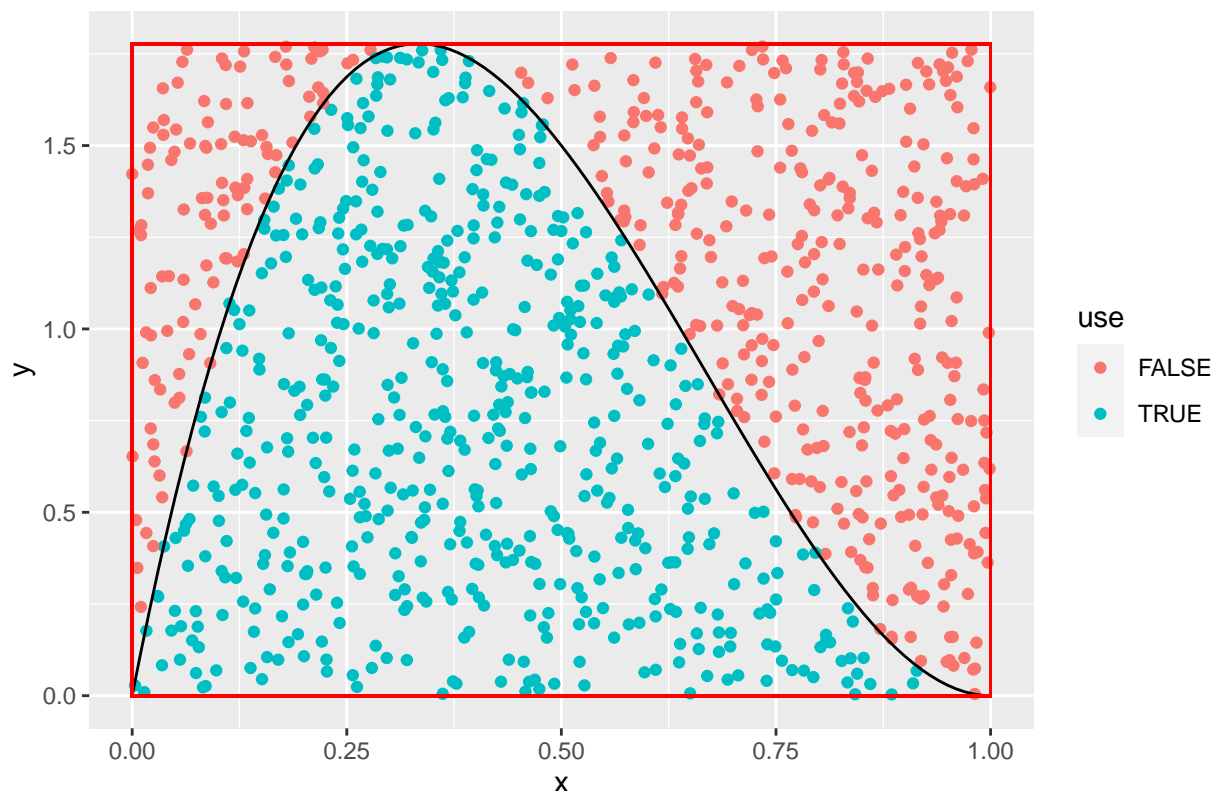
Rejection Sampling

Intuition

- A naive thought: If I fill a region that envelopes the pdf with points uniformly, then those points under the curve form a good representation of the target pdf and the corresponding x values constitute a good sample from the target distribution. An example:

```
## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.  
## Please use `as_tibble()` instead.  
## The signature and semantics have changed, see `?as_tibble`.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

Proposal distribution outlined in red, target in black



The points highlighted in green have the desired distribution. We throw away the reddish samples.

- But we can't control the exact sample size n (how much points are accepted). Also it is not efficient in the tails in the above example.

A Better Algorithm to Complete the Task: Rejection Sampling Algorithm

1. Choose a proposal density, $q(\theta)$ that we can easily sample from (e.g. uniform or normal) such that:
 2. Find $M = \max \frac{p(\theta|y)}{q(\theta)}$
 - If $M = \infty$ then q cannot be used as a proposal distribution
 - If M is finite, $Mq(\theta)$ “envelopes” $p(\theta|y)$
 3. Draw a sample, $\theta^{(s)}$ from $q(\theta)$
 4. Draw a $u^{(s)} \sim \text{Unif}(0, 1)$
 - If $u^{(s)} < \frac{p(\theta^{(s)}|y)}{Mq(\theta^{(s)})}$ then accept $\theta^{(s)}$ as a samples
 - Otherwise throw out $\theta^{(s)}$ and try again
- Discussion:
 - The crucial part is to find a good proposal density that can envelope our target density and make sampling efficient.
 - Uniform distribution won’t work when our target distribution has a domain $(-\infty, +\infty)$
 - In the case where target distribution has a domain $(-\infty, +\infty)$, normal distribution might be a good proposal but it is not always a good choice.

An Example on Rejection Sampling

Problem and Analysis

- Suppose our target density is $p(x) \propto \frac{1}{4}e^{-|x|/2}|\sin 2x|$. We can find the constant part using Laplace transformation but for now let’s consider it as C . So $p(x) = C\frac{1}{4}e^{-|x|/2}|\sin 2x|$
- If we look closely we can notice that the nonperiodic part $\frac{1}{4}e^{-|x|/2}$ is the pdf of laplace distribution.
- Let’s visualize the laplace density and the main part of $p(x)$: $\frac{1}{4}e^{-|x|/2}|\sin 2x|$

Two methods are introduced to draw the same plot, we recommend you using ggplot

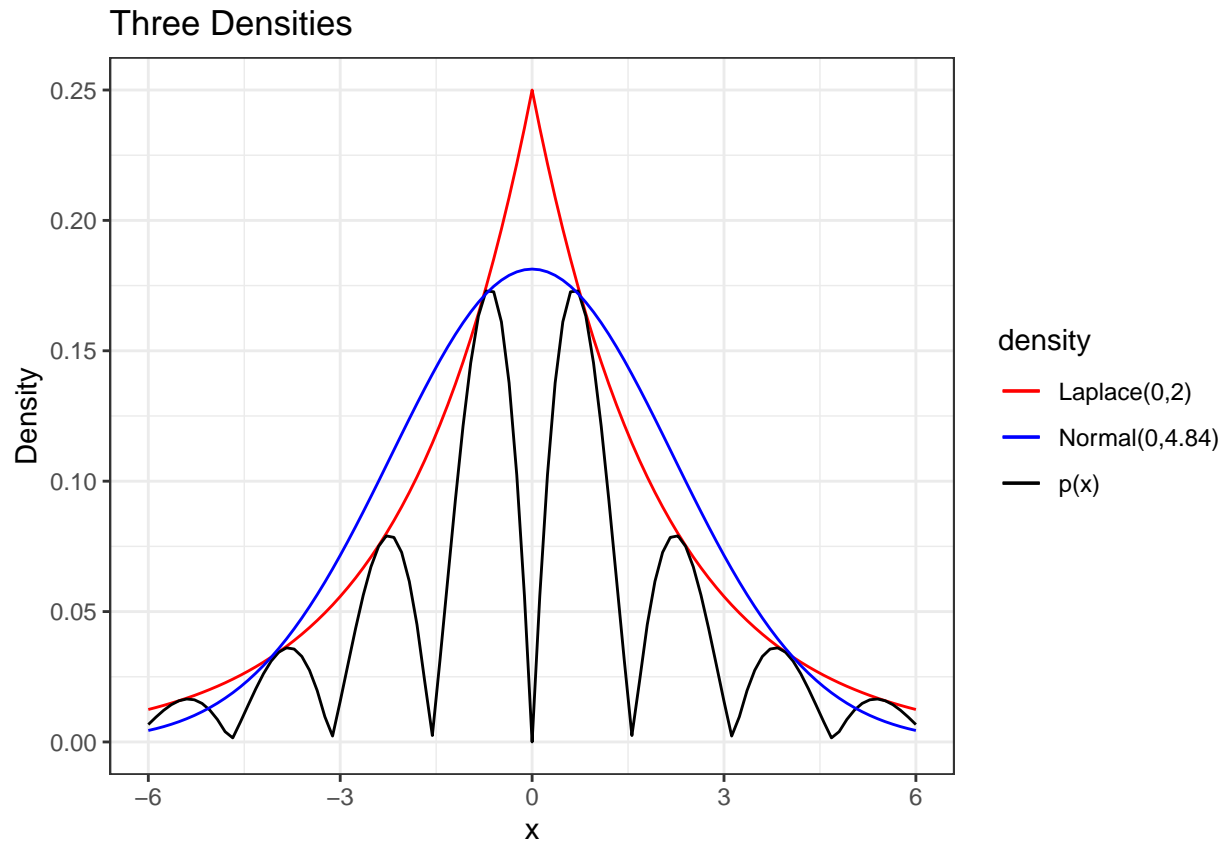
#Method 1

```
ggplot(aes(x=x), data = data.frame(x=0) )+  
  stat_function(fun = function(x) dlaplace(x, 0, 2), aes(colour = "Laplace(0,2)"))+  
  stat_function(fun = function(x) dlaplace(x, 0, 2)*abs(sin(2*x)), aes(colour = "p(x)"))+  
  stat_function(fun = function(x) dnorm(x, 0, 2.2), aes(colour = "Normal(0,4.84)"))+  
  xlim(c(-6,6))+theme_bw()+scale_y_continuous(name = "Density")+scale_colour_manual(name="density",value=c("red","green","blue"))
```

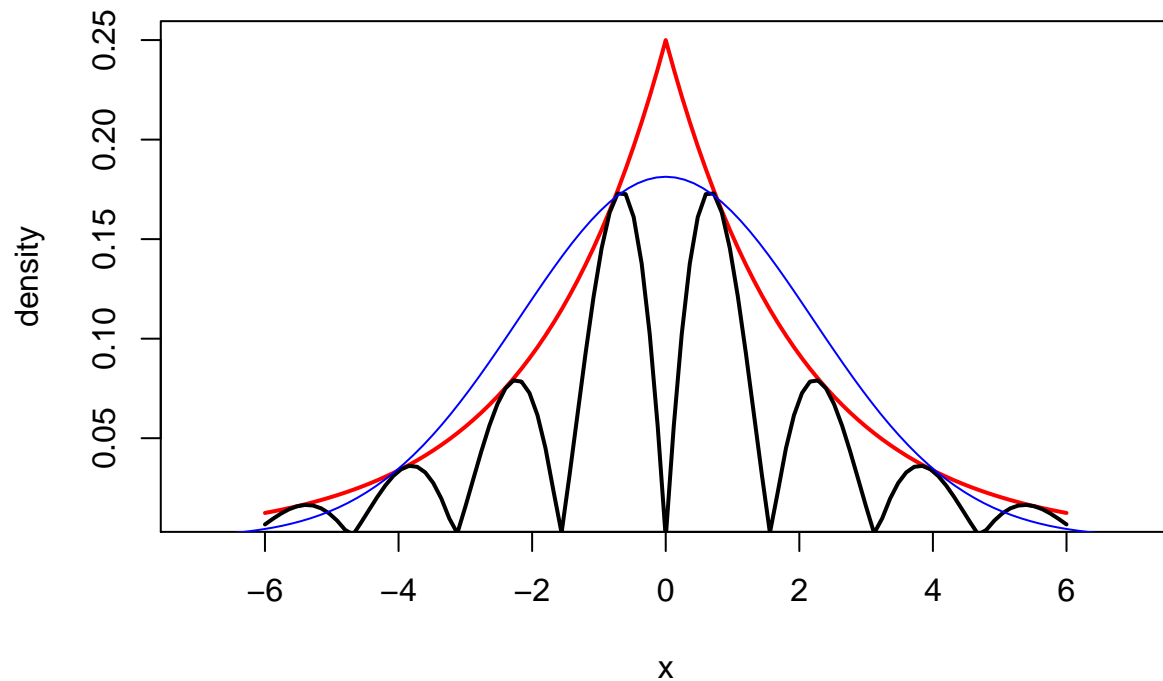
```
## Warning: `mapping` is not used by stat_function()
```

```
## Warning: `mapping` is not used by stat_function()
```

```
## Warning: `mapping` is not used by stat_function()
```



```
#Method 2
curve(dlaplace(x, 0, 2), from=-6, to=6, xlim=c(-7, 7), lwd=2, col = "red", ylab = "density")
curve(dlaplace(x, 0, 2)*abs(sin(2*x)), from=-6, to=6, xlim=c(-7, 7), lwd=2, add = TRUE)
curve(dnorm(x, 0, 2.2), add = TRUE, col = "blue")
```



The above plot shows that we can always envelope $p(x)$ by $Cq(x)$ where $q(x) = \frac{1}{4}e^{-|x|/2}$ is the density for

Laplace(0,2). Thus Laplace(0,2) can be our proposal density

- The other potential proposal density is normal density. We want to see whether normal density can be a proper proposal distribution. As we can see from the blue line of the plot. A $N(0, 4.84)$ density can cover the middle of our density but not the tails. That's because Laplace distribution have heavier tails than normal distribution and our target is originated from laplace distribution.
- Is it possible that if we amplify $N(0, 4.84)$ by a constant, it will cover the target? No. Because of the heavier tail property, the amplified $N(0, 4.84)$ will eventually goes under our target distribution

Solve the problem with Laplace(0,2)

- Follow the algorithm, our proposal density $q(x) = \frac{1}{4}e^{-|x|/2}$ is Laplace(0,2)
- Next, we should find $M = \max \frac{p(\theta|y)}{q(\theta)}$. It can be easily found to be C , the constant part of $p(x)$. If we ignore C , then M should be 1. The following code verifies this.

important code for HW

```
density_ratio<-function(x){#ratio of densities, in HW you should use the exact form of density.
  dlaplace(x, 0, 2)*abs(sin(2*x))/dlaplace(x, 0, 2)
}

M <- optimize(density_ratio, lower = -10, upper = 10, maximum = TRUE)$objective
M
```

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

- Then we should draw sample from $q(\theta)$ and then use another uniform sample to decide whether we should accept or reject the sample.

important code for HW

```
set.seed(123)
n <- 1e6
laplace_sample<- rlaplace(n, 0, 2)
accept <- runif(n) < density_ratio(laplace_sample)/M
samples <- laplace_sample[accept]
head(accept)
```

```
## [1] TRUE TRUE TRUE FALSE TRUE FALSE
```

```
mean(accept)
```

```
## [1] 0.62968
```

```
var(samples)
```

```
## [1] 8.10665
```

```
mean(samples)
```

```
## [1] -0.00868812
```

Estimated 60% quantile interval and 60% HPD(Highest Posterior Density) region

To calculate HPD based on samples we need HDInterval package.

All the follows are important code for HW

```
hd_region <- hdi(density(samples), allowSplit = TRUE, credMass = 0.6)
hd_region
```

```
##           begin           end
## [1,] -2.5035597 -2.0885406
## [2,] -1.2585023 -0.2209545
## [3,]  0.1940647  1.2316125
## [4,]  2.0616508  2.4766699
## attr("credMass")
## [1] 0.6
## attr("height")
## [1] 0.1067766
```

The total length HPD region is given by:

```
sum(hd_region[, "end"]-hd_region[, "begin"])
```

```
## [1] 2.905134
```

The quantile interval and its length is given by:

```
quantile_interval<-quantile(samples,c(0.2,0.8))
quantile_interval
```

```
##           20%           80%
## -2.017739  2.002357
```

```
quantile_interval[2] - quantile_interval[1]
```

```
##           80%
## 4.020097
```

```
curve(dlaplace(x, 0, 2)*abs(sin(2*x)), from=-6, to=6, xlim=c(-7, 7))
segments(x0=hd_region[1, 1], y0=0, x1=hd_region[1, 2], y1=0, col="red", lwd=3)
segments(x0=hd_region[2, 1], y0=0, x1=hd_region[2, 2], y1=0, col="red", lwd=3)
segments(x0=hd_region[3, 1], y0=0, x1=hd_region[3, 2], y1=0, col="red", lwd=3)
segments(x0=hd_region[4, 1], y0=0, x1=hd_region[4, 2], y1=0, col="red", lwd=3)
segments(x0=quantile_interval[1], y0=0.01, x1=quantile_interval[2], y1=0.01, col="blue", lwd=3)
```

