HW: Week 6

36-350 – Statistical Computing

Week 6 – Spring 2022

Name: Henry Wu

Andrew ID: zhenyuw

You must submit **your own** lab as a PDF file on Gradescope.

```
suppressWarnings(library(tidyverse))
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

HW Length Cap Instructions

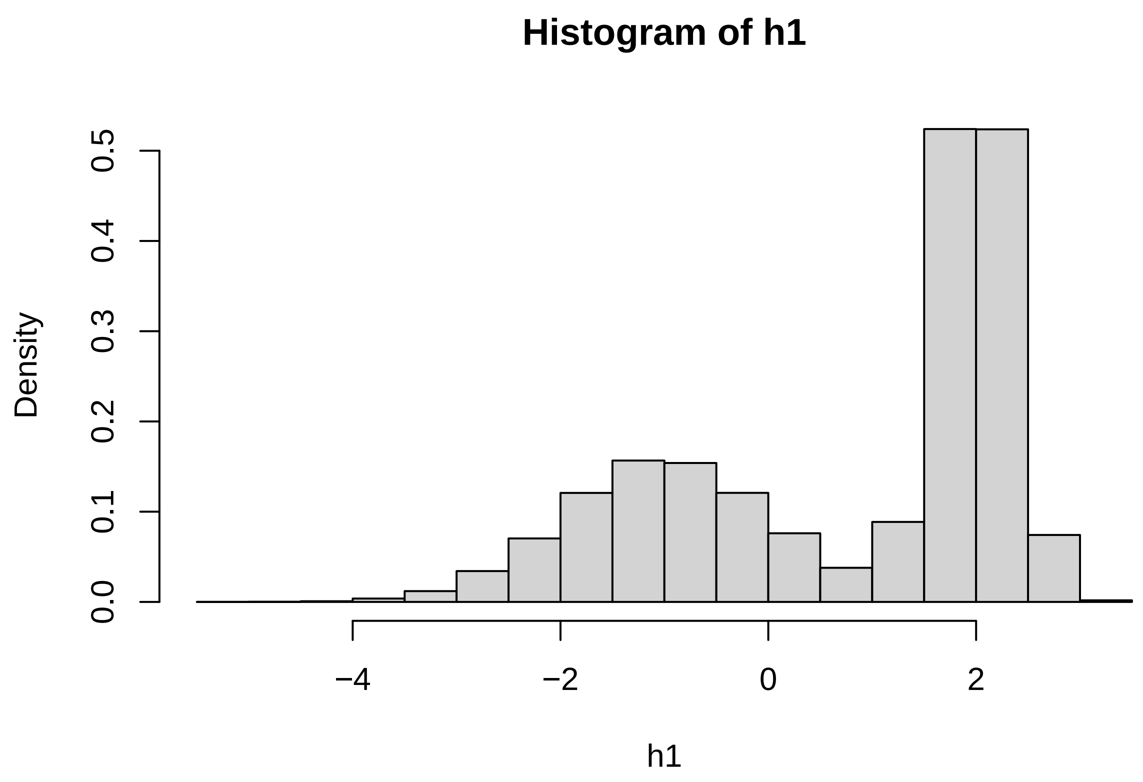
- If the question requires you to print a data frame in your solution e.g. `q1_out_df`, you must first apply **head(q1__out__df, 30)** and **dim(q1__out__df)** in the final knitted pdf output for such a data frame.
- Please note that this only applies if you are knitting the `Rmd` to a `pdf`, for Gradescope submission purposes.
- If you are using the data frame output for visualization purposes (for example), use the entire data frame in your exploration
- The **maximum allowable length** of knitted pdf HW submission is **30 pages**. Submissions exceeding this length *will not be graded* by the TAs. All pages must be tagged as usual for the required questions per the usual policy
- For any concerns about HW length for submission, please reach out on Piazza during office hours

Question 1

(20 points)

Create a Gaussian mixture model sampler. In this sampler, a datum has a 40% chance of being sampled from a N(-1,1) distribution, and a 60% chance of being sampled from a N(2,1/9) distribution. Sample 100,000 data and create a density histogram of your result. Hint: use `sample()` with `replace` set to `TRUE` and an appropriate vector for `prob` in order to determine which of your 100,000 data should randomly be assigned to distribution 1 as opposed to distribution 2. Also note that if you create a sample of data from distribution 1 and another sample from distribution 2, you can simply combine them by doing, e.g., `x = c(sample1,sample2)`, where `x` becomes a vector of length 100,000.

```
set.seed(101)
n = 100000
s1 = rnorm(n/2, mean = -1, sd = 1)
s2 = rnorm(n/2, mean = 2, sd = 1/3)
x = c(s1, s2)
h1 = sample(x, n, replace = TRUE, prob = c(rep(0.4, n/2), rep(0.6, n/2)))
hist(h1, probability = TRUE)
```



Question 2

(20 points)

What is the mean of the mixture model in Q1? Compute this via importance sampling, with 100,000 sampled points. You should get an answer around 0.8 (which you can actually derive analytically: if $X \sim N(-1,1)$

and $Y \sim N(2, 1/9)$, then $E[0.4X + 0.6Y]) = 0.4E[X] + 0.6E[Y] = -0.4 + 1.2 = 0.8$).

```
set.seed(390)
x2 = rnorm(n, 0, sqrt(3))
h2 = dnorm(x2, 0, sqrt(3))
g = x2

f = function(x){
  r = runif(n, 0, 1)
  f.x = rep(0, n)
  f.x[r <= 0.4] = dnorm(x2[r<=0.4], mean = -1, sd = 1)
  f.x[r > 0.4] = dnorm(x2[r>0.4], mean = 2, sd = 1/3)
  return (f.x)
}


mean(g*f(x2)/h2)
```

```
## [1] 0.8064724
```

Question 3

(20 points)

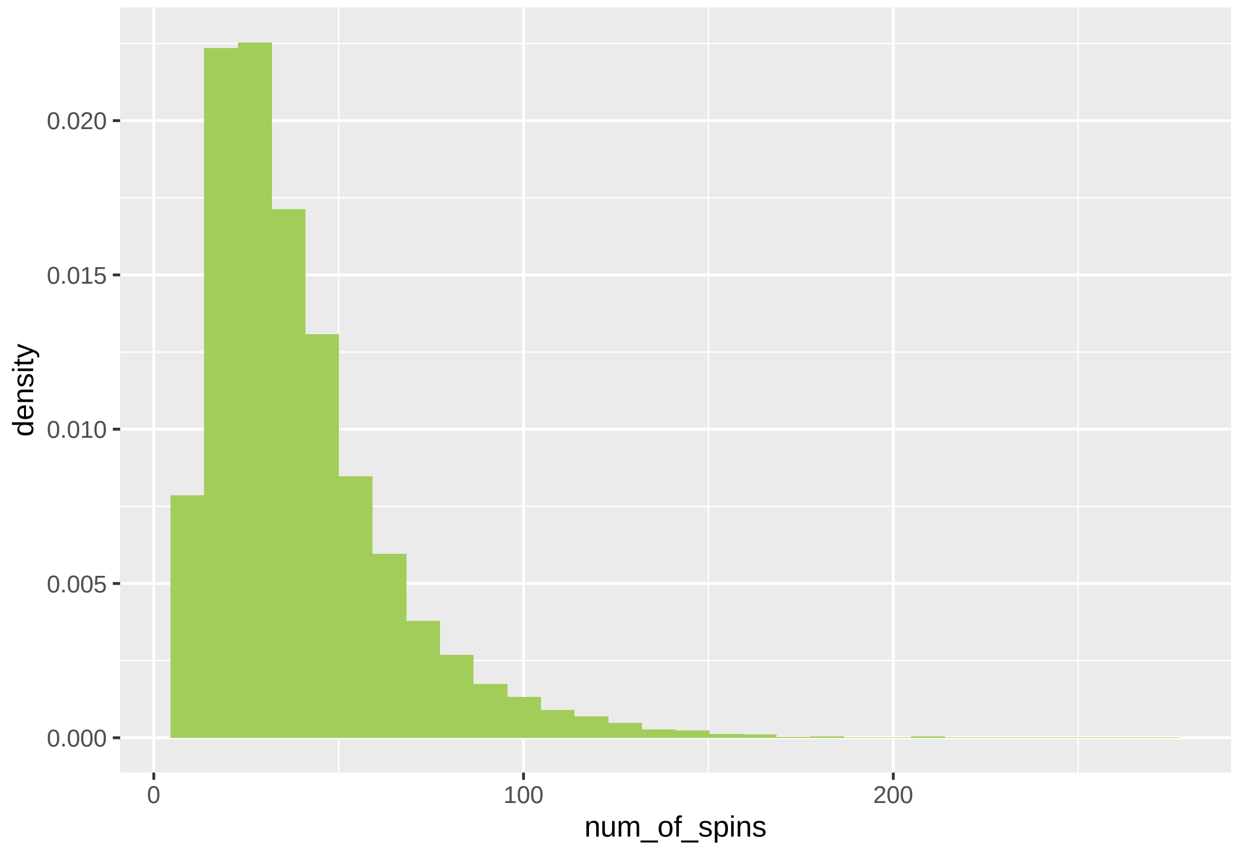
Remember the Chutes and Ladders question? (Q4 of HW 2.) Display a probability histogram that shows the empirical PDF for the number of spins, computed over 10,000 Chutes and Ladders games. Also display the average number of spins that it takes to win the game (approximately 39, give or take) and the minimum number of spins. (Display these two numbers using `cat()`, being see to indicate which is the mean and which is the minimum number of spins.) (Free feel to use your code from HW 2 as a base for what you do here.)

```
## chutes and Ladders calculations
ladder.bottom = c(1,4,9,21,28,36,51,71,80)
ladder.top = c(38,14,31,42,84,44,67,91,100)
chute.top = c(98,95,93,87,64,62,56,49,48,16)
chute.bottom = c(78,75,73,24,60,19,54,11,26,6)

nextSquare = function(square, move) {
  newSquare = square + move
  if (newSquare > 100) return (square)
  w = which(ladder.bottom == newSquare)
  if (length(w) > 0) newSquare = ladder.top[w]
  w = which(chute.top == newSquare)
  if (length(w) > 0) newSquare = chute.bottom[w]
  return(newSquare)
}

# num return the number of spins it takes to get to 100
num.spin <- function(x) {
  square = x
  count = 0
  while (square != 100) {
    count = count + 1
    spin = sample(1:6,1)
    square = nextSquare(square, spin)
  }
  return(count)
}
#random 10000 games
n3 = 10000
set.seed(103)
R = rep(0, n3)
df3 = data.frame("num_of_spins" = unlist(lapply(R, num.spin)))
ggplot(data=df3, aes(x = num_of_spins)) +
  geom_histogram(fill="darkolivegreen3",aes(y=..density..))
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



```
mean(df3$num_of_spins)
```

```
## [1] 39.155
```

```
min(df3$num_of_spins)
```

```
## [1] 7
```

Question 4

(20 points)

The ratio of the area of a circle to the area of a square into which the circle is inscribed is $\pi/4$. Does this ratio increase or decrease with dimensionality? For instance, what is the ratio of volume of a sphere to the volume of a cube into which the sphere is inscribed? Is it less than $\pi/4$? Compute (and display) the ratio for dimensions 3, 4, . . . , 10. The result that you see has manifestations for, e.g., algorithms based on nearest neighbors, etc. Curse of dimensionality, n'at. (Hint: to do this calculation succinctly, consider putting samples from your uniform distribution into a $k \times d$ matrix, where k is the number of sampled points, and d is the dimensionality. Then you can use `apply()` to determine the distance of the points from the origin and you can easily finish the calculation from there. . .)

```
# FILL ME IN
# unif sample points in cube, then cal ratio in sphere to all
# each d sample 1000 pts
k = 1000
set.seed(319)
res = rep(0,8)
f = function(d) {
  M = matrix(runif(k*d, 0, 1), ncol = d)
  x = apply(M, 1, function(x){norm(x, type = "2")})
  length(x[x<=1])/k
}
df = data.frame(unlist(lapply(3:10, f)), row.names = as.character(3:10))

names(df) = "ratio of pts in sphere vs. in cube"
df
```

```
##    ratio of pts in sphere vs. in cube
## 3                               0.535
## 4                               0.306
## 5                               0.152
## 6                               0.077
## 7                               0.029
## 8                               0.018
## 9                               0.006
## 10                              0.001
```

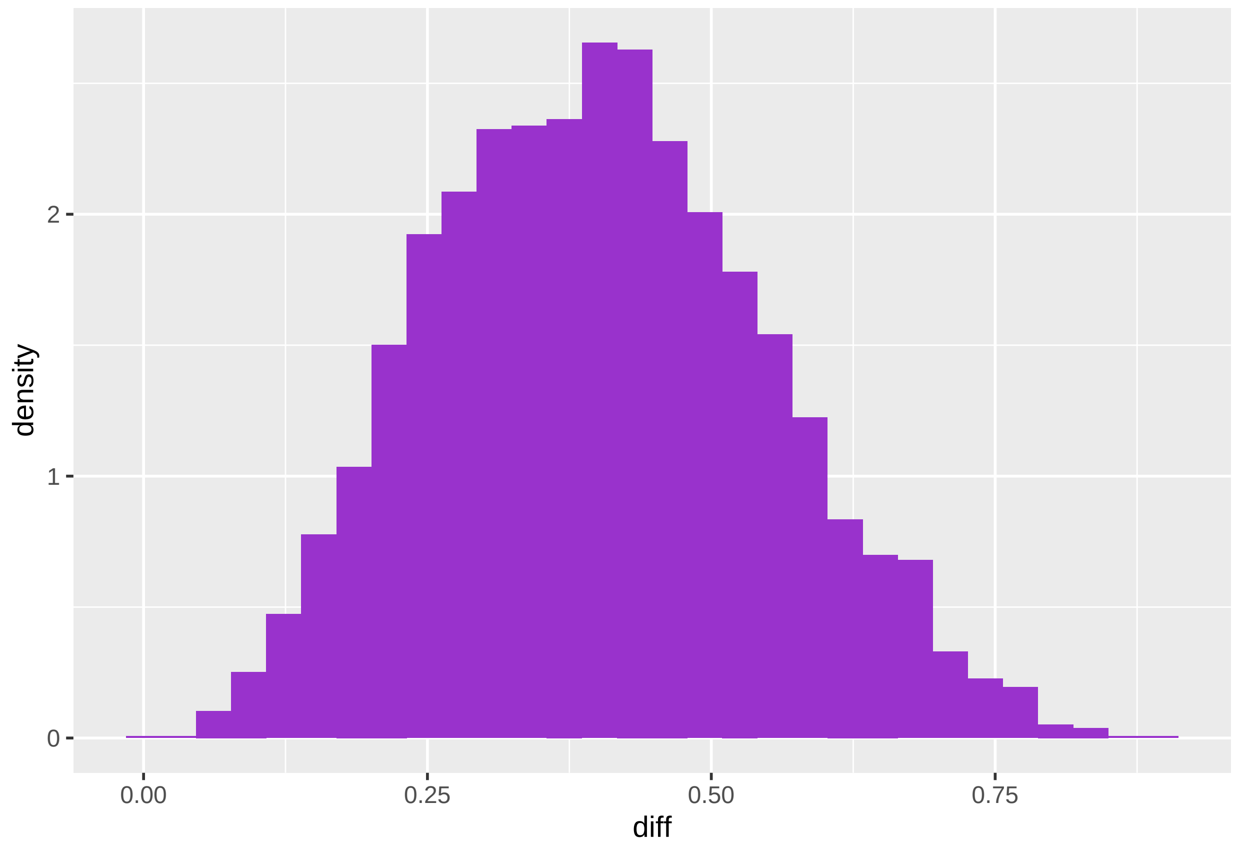
Question 5

(20 points)

What is the probability distribution function for the difference between the maximum value and the median value when you sample nine data from a Uniform(0,1) distribution? Generate an empirical distribution by repeating the process of sampling nine data 5,000 separate times, and record the differences from the maximum and median values. Display a histogram of your result; in addition, display the mean and standard error. The mean should be approximately 0.4.

```
# FILL ME IN
set.seed(101)
data5 = matrix(runif(9*5000, 0, 1), ncol = 9)
df = data.frame("diff"=apply(data5,1,max)-apply(data5,1,median))
ggplot(data=df,mapping=aes(x=diff)) + geom_histogram(fill="darkorchid",aes(y=..density..))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
mean(df$diff)
```

```
## [1] 0.4015588
```

```r
sd(df$diff)/sqrt(5000)
```

```
## [1] 0.002078052
```

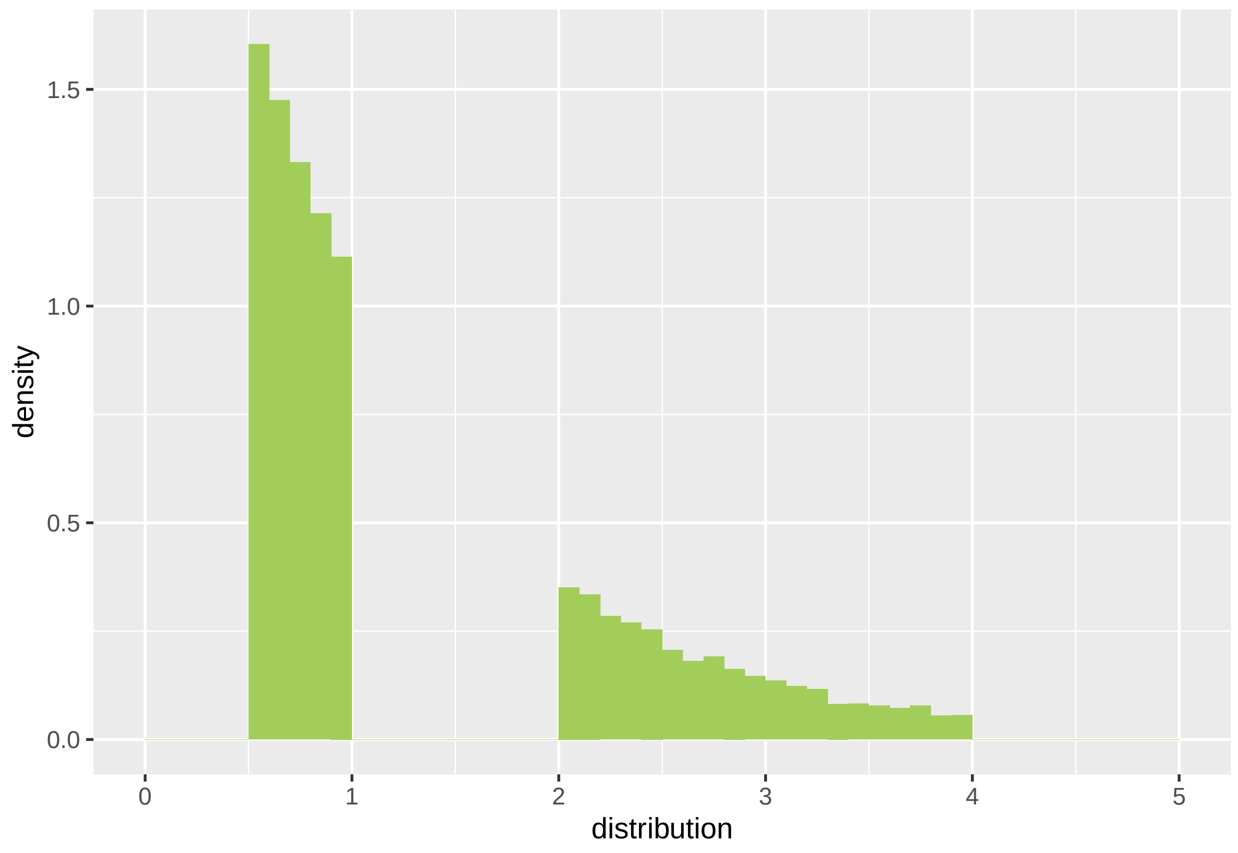
Question 6

(20 points)

Write an inverse transform sampler that samples 10,000 data from a exponential distribution with rate parameter 1, but it only keeps data sampled over the ranges $[0.5, 1]$ and $[2, 4]$. Make a probability histogram of your result. This time, tweak the call to `geom_histogram()` by adding the argument `breaks=seq(0,5,by=0.1)`.

This is a bit tricky. (Note: this is an inverse transform sampler, so every single randomly sampled uniform random variable has to get mapped to a valid value of x.) You might want to start by computing the probabilities $P[0.5 \leq X \leq 1]$ and $P[2 \leq X \leq 4]$. Call these two quantities u_{lo} and u_{hi}, and sample random numbers from a Uniform(0,$u_{\text{lo}} + u_{\text{hi}}$) distribution. If the number is $< u_{\text{lo}}$, the sampled number should be mapped to a sample from the lower range, whereas if the number is $> u_{\text{lo}}$, it should be mapped to a sample from the upper range. Note: to map from you uniform random variables to exponentially distributed ones, pass your uniform r.v.'s into `qexp()`.

```r
# FILL ME IN
set.seed(105)
u_lo = pexp(1, rate = 1) - pexp(0.5, rate = 1)
u_hi = pexp(4, rate = 1) - pexp(2, rate = 1)
dt6 = runif(10000, 0, u_lo + u_hi)
sp1 = (qexp(dt6[dt6<=u_lo]+pexp(0.5, rate = 1), rate = 1))
sp2 = (qexp(dt6[dt6>u_lo]+pexp(2, rate = 1) - u_lo, rate = 1))
sp = data.frame(c(sp1, sp2))
colnames(sp) = "distribution"
ggplot(data=sp, aes(x = distribution)) +
  geom_histogram(fill="darkolivegreen3",aes(y=..density..),
                 breaks=seq(0, 5, by=0.1))
```


density
distribution