HW: Week 6

36-350 – Statistical Computing

Week 6 – Spring 2021

Name: Jacky liu

Andrew ID: jackyl1

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

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

```
## -- Attaching packages --------------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.1
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

HW Length Cap Instructions

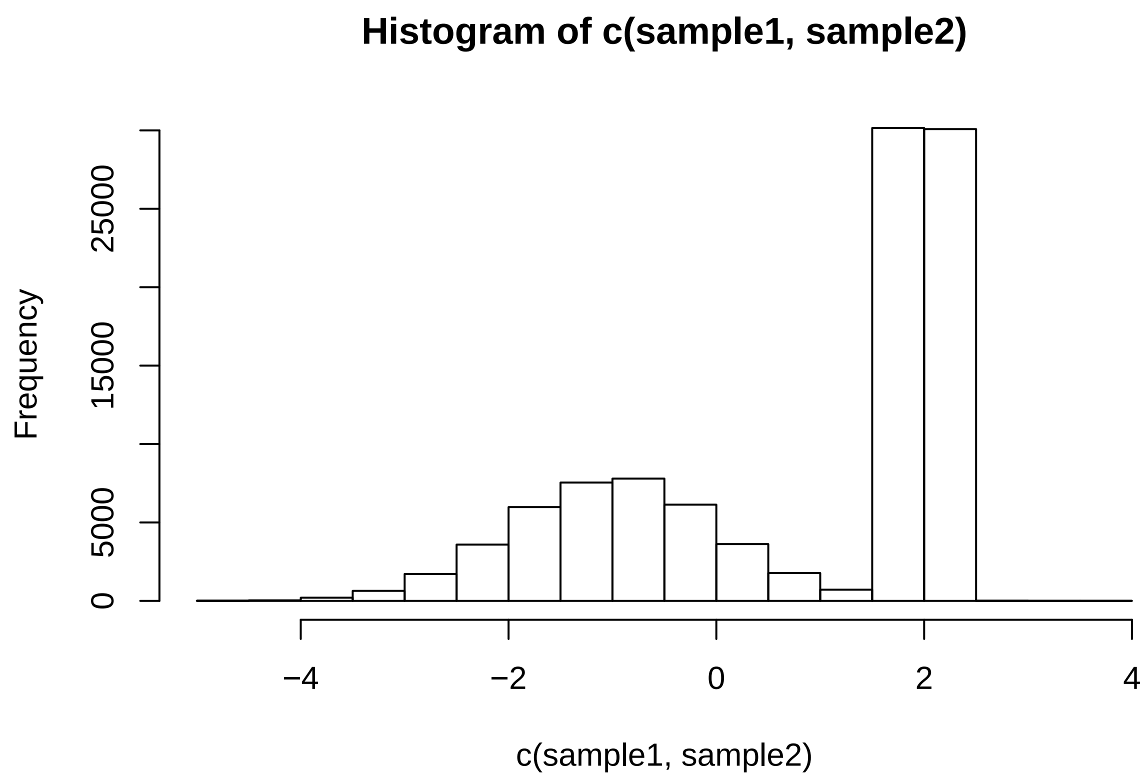
- If the question requires you to print a data frame in your solution e.g. `q1_out_df`, you must first apply **head(q1__out__df, 30)** and **dim(q1__out__df)** in the final knitted pdf output for such a data frame.
- Please note that this only applies if you are knitting the `Rmd` to a `pdf`, for Gradescope 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

(10 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(420)
sample1 = rnorm(40000, -1, 1)
sample2 = rnorm(60000, 2, 1/9)
hist(c(sample1, sample2))
```



Question 2

(10 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 $E[0.4X + 0.6Y]) = 0.4E[X] + 0.6E[Y] = -0.4 + 1.2 = 0.8$).

```r
mean(c(sample1,sample2))
```

```
## [1] 0.8050653
```

Question 3

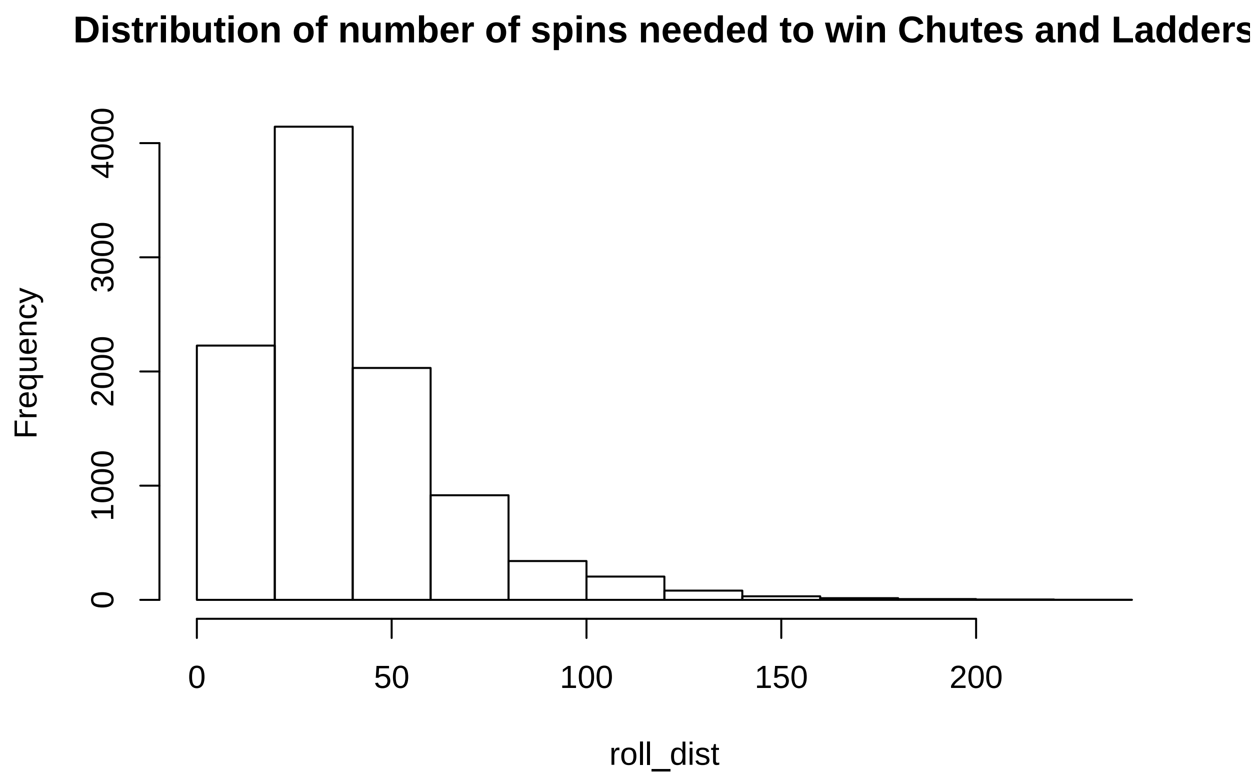
(10 points)

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.)

Modified code from hw2 to count rolls

```r
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.bottom = c(6, 11, 19, 24, 26,53,60,73, 75,78)
chute.top = c(16, 49, 62,87, 47,56, 64, 93, 95, 98)

playChutes = function(){
  playing = TRUE
  pos = 0
  num_rolls = 0
  while(playing){
    if(pos == 100){
      playing = FALSE
      #cat("Game Over. Final Position: ",pos,"\n")
      return(num_rolls)
      break
    }
    else {
      roll = sample(1:6, size=1)
      num_rolls = num_rolls + 1
      #cat("Roll: ", roll, "\n")
      if(pos + roll > 100){
        #cat("Over 100", "\n")
        next
      }
      pos = pos + roll
      #cat("Pos: ", pos, "\n")
      l = which(ladder.bottom == pos)
      c = which(chute.top == pos)
      if(length(l) > 0) {
        pos = ladder.top[l]
        #cat("Climbed up the ladder from ", ladder.bottom[l], " to ", ladder.top[l], "\n")
      }
      else if(length(c) > 0) {
        pos = chute.bottom[c]
        #cat("Went down the chute from ", chute.top[c], " to ", chute.bottom[c], "\n")
      }
    }
  }
}
```

```r
set.seed(420)
num_sim = 10000
roll_dist = vector(length=num_sim)
for(i in 1:num_sim){
  roll_dist[i] = playChutes()
}
hist(roll_dist, main = "Distribution of number of spins needed to win Chutes and Ladders")
```



```r
cat("The mean number of spins to win is: ", mean(roll_dist), "\n")
```

```
## The mean number of spins to win is:  39.3062
```

```r
cat("The minimum number of spins to win is: ", min(roll_dist))
```

```
## The minimum number of spins to win is:  7
```

Question 4

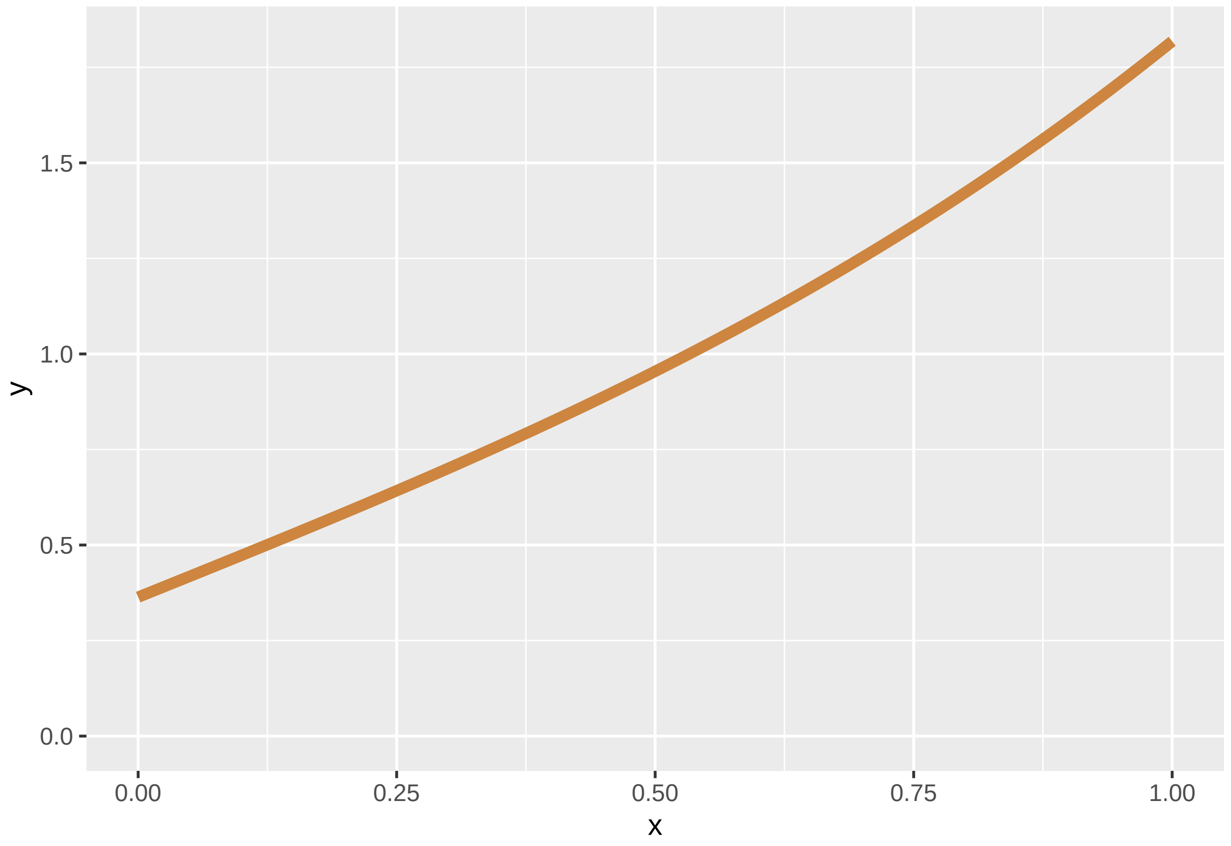
(10 points)

You are given the following distribution:

$$f(x) = \frac{4}{11}(x^3 + 3x + 1) \quad x \in [0, 1]$$

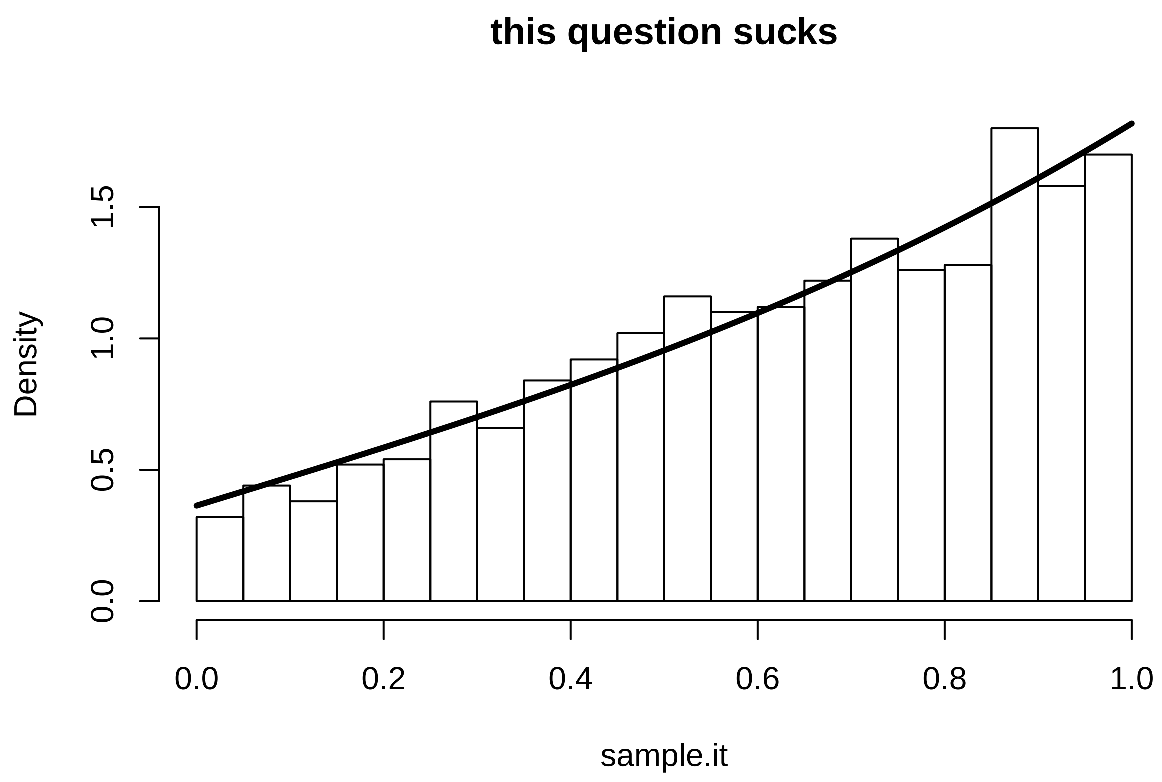
It looks like this:

```r
x = seq(0,1,by=0.001)
fx = 4*(x^3+3*x+1)/11
ggplot(data=data.frame(x=x,y=fx),mapping=aes(x=x,y=y)) +
  geom_line(col="peru",size=2) + ylim(0,max(fx))
```



Code up an inverse transform sampler that allows you to efficiently sample 1000 data from this distribution. Initially you will work with this and say "but…I cannot easily invert the CDF, because it's a quartic and all." To which I say "`polyroot()`, which will give you one real root between 0 and 1." To which you will say, "how do I extract that real root?" To which I will say "If you save the output of `polyroot()` as `p`, then the real roots are given by `w = which(abs(Im(p))<1.e-6`." (The 1.e-6 is a check against round-off error.) You then determine which value of `Re(p)[w]` is within the pdf bounds. Histogram your sample with the function line overlaid, and save your sample as `sample.it`. Note: pass a new argument to your histogram, `breaks=seq(0,1,by=0.05)`.

```r
set.seed(69)
u = runif(1000)
sample.it = vector(length=1000)
for(i in 1:1000){
  p = polyroot(c(-u[i], 4/11, 6/11, 0, 1/11))
  w = which(abs(Im(p))<1.e-6)
  va = Re(p)[w]
  sample.it[i] = va[1]
}
hist(sample.it, probability=TRUE, breaks=seq(0,1,by=0.05), main = "this question sucks")
lines(x,fx, lwd=3)
```

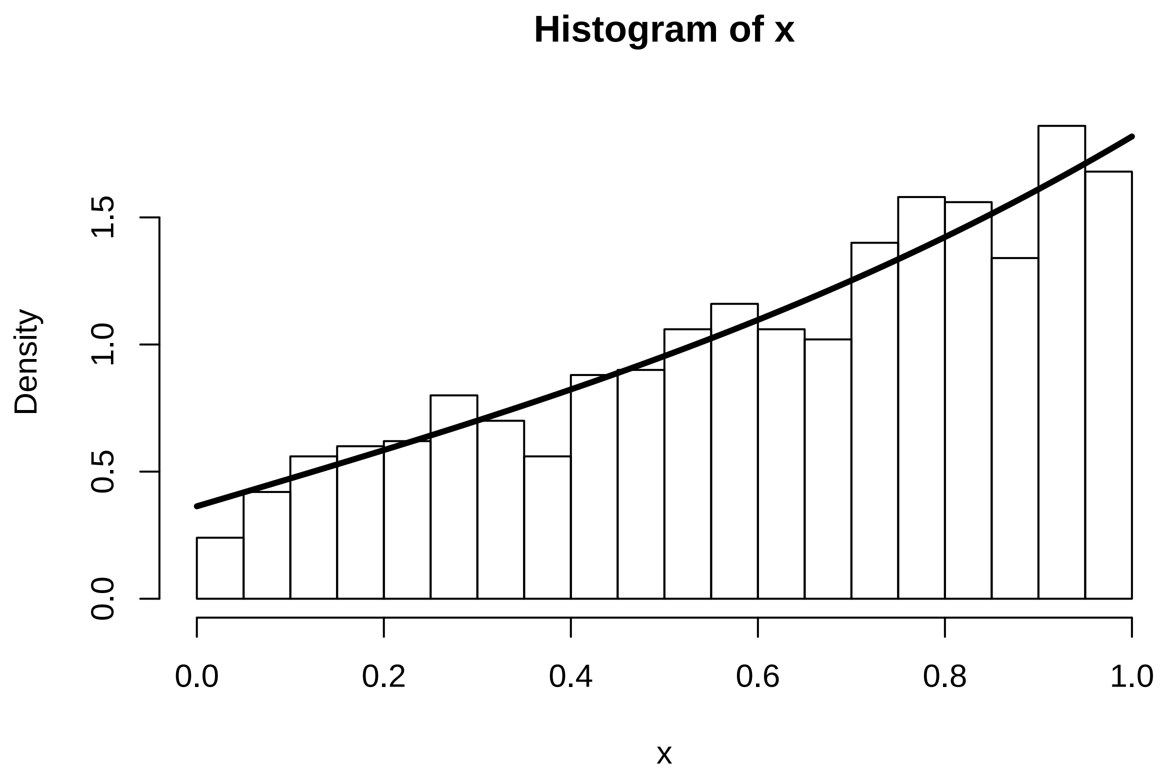


Question 5

(10 points)

Code up a rejection sampler that allows you to also sample 1000 data from this pdf given in the last question. Again, histogram your sample and overlay the pdf. Save your sample as `sample.rs`.

```
set.seed(666)
k = 1000
x = rep(NA,k)
ii = 1
while ( ii <= k ) {
  x[ii] = runif(1,min=0,max=1)
  if ( runif(1,min=0,max=2) < 4*(x[ii]^3+3*x[ii]+1)/11 ) { # then sample along the vertical direction
    ii = ii + 1
  }
}
hist(x,probability=TRUE,breaks=seq(0,1,by=0.05))
lines(seq(0,1,by=0.001),fx, lwd=3)
```



```
sample.rs = x
```

Question 6

(10 points)

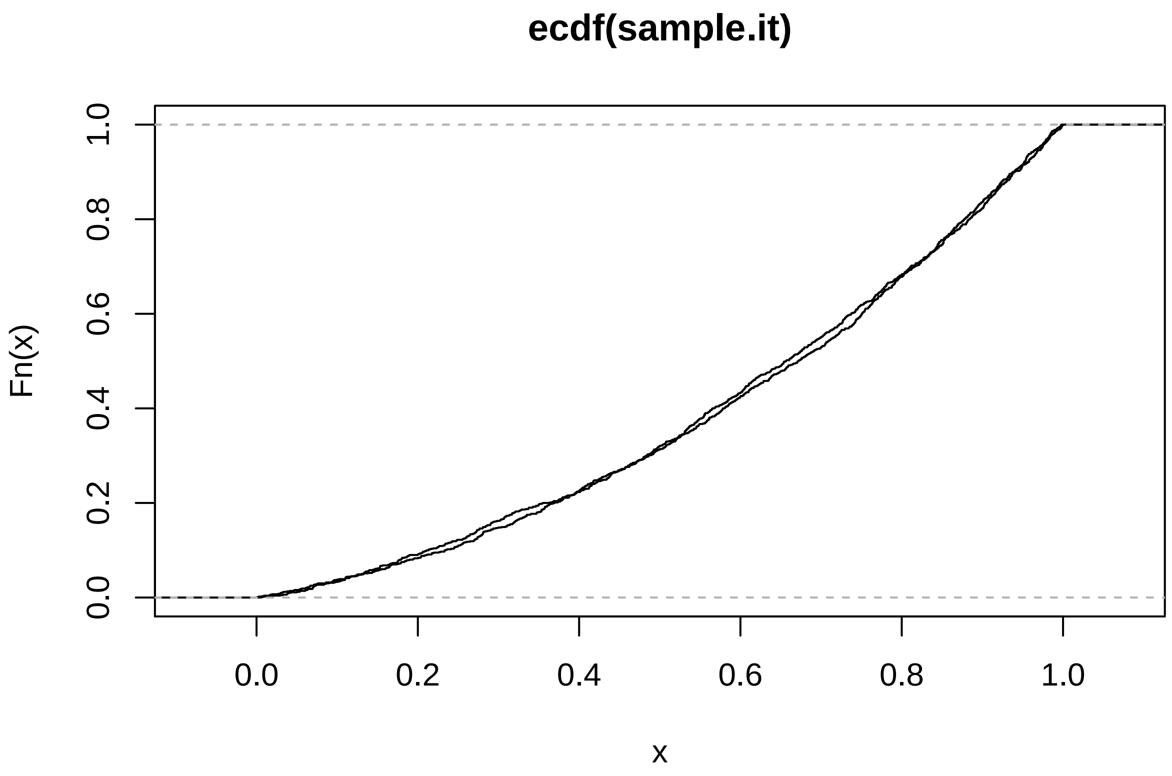
Test the hypothesis that `sample.it` and `sample.rs` are both sampled from the same parent population. (I mean they are, but...) Either recall how you would do a two-sample test or Google how you would do it. (Note: I'm not talking about a two-sample t-test here! We are not testing the hypothesis that the distribution means are the same. We are testing the hypothesis that both samples are drawn from the same underlying population.) There are various options for doing this; pick one, and display the p-value. If it less than 0.05, we reject the null. (Hint: it shouldn't be.) In addition, plot the empirical cdfs for both samples; see the documentation for `ecdf()` for help. (To be clear: use the base `R` function `plot()` here, and not `ggplot()`.) Note that to plot a second ecdf on top of the first, you need to call `plot()` a second time, with the argument `add=TRUE`.

```
ks.test(sample.it, sample.rs)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sample.it and sample.rs
## D = 0.028, p-value = 0.828
## alternative hypothesis: two-sided
```

We fail to reject the null.

```r
plot(ecdf(sample.it))
plot(ecdf(sample.rs), add=TRUE)
```



Question 7

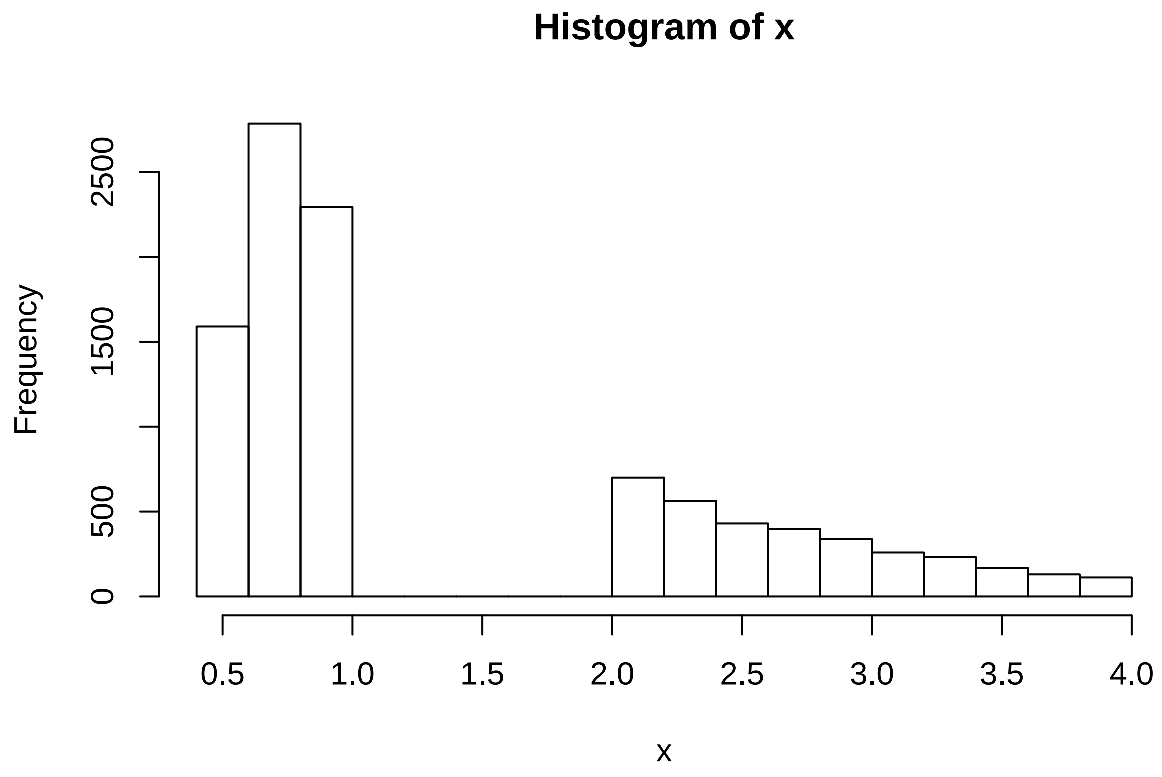
(10 points)

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 $P[0.5 \le X \le 1]$ and $P[2 \le X \le 4]$. Call these two quantities u_{lo} and u_{hi}, and sample random numbers from a Uniform(0,$u_{\mathrm{lo}} + u_{\mathrm{hi}}$) distribution. If the number is $< u_{\mathrm{lo}}$, the sampled number should be mapped to a sample from the lower range, whereas if the number is $> u_{\mathrm{lo}}$, 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
k = 10000
u_lo = pexp(1)-pexp(0.5)
u_hi = pexp(4)-pexp(2)
samp = runif(k, 0, u_lo+u_hi)
```

```r
x = ifelse(samp < u_lo, qexp(pexp(0.5)+samp), qexp(pexp(2)+samp-u_lo))
hist(x)
```



Question 8

(10 points)

And now for something completely different: randomly sampling a name for your new baby, given 1930 Social Security Administration data. (You did know you were having a baby, right?)

You'll see the `babynames` tibble has five columns. Using `dplyr` techniques, extract the rows for which the year is 1930, and then create a new data frame with just the columns `name` and `prop`. `prop` is a vector which gives the proportion of children given a particular name, but for some strange reason it does not sum to 1. Anyway, use a normalized version of this vector to appropriately sample one name from the `name` column. Show that name. Done. (If you set the random number seed to 111, you should get the name "Bernard".)

```r
if ( require(babynames) == FALSE ) {
  install.packages("babynames",repos="https://cloud.r-project.org")
  library(babynames)
}
```

```
## Loading required package: babynames
```

```
normalize = function(x){
  total = sum(x)
  x = x / total
  return(x)
}
babynames = babynames %>% filter(year == 1930)
babynames.df = babynames[c(3,5)]
babynames.df$prop = normalize(babynames.df$prop)
print(sum(normalize(babynames.df$prop))) # ensure that vector is normalized properly
```

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

```
sample(babynames.df$name, 1)
```

```
## [1] "Concepcion"
```

Question 9

(10 points)

Numerically estimate the median of the pdf

$$f(x) = \frac{2.92959}{\sqrt{2\pi}} e^{-x^2/2} \quad x \in [0,1] \, .$$

(This is a truncated normal distribution.) The median is the value y such that

$$\int_0^y f(x) dx = 0.5 \, .$$

A not-elegant way to do this is to use `integrate()` over and over again until you hone in on an integral value of 0.5. Don't do this. A more elegant solution is to determine, via `uniroot()`, the root of the function

$$g(y) = \left(\int_0^y f(x) dx \right) - 0.5 \, ,$$

i.e., the value of y such that $g(y) = 0$. Do do this.

```
f = function(x) (2.92959/sqrt(2*pi))*exp(-x^2/2)
g = function(y) (integrate(f, lower=0, upper=y))$value - 0.5
uniroot(g, interval=c(0,1))$root
```

```
## [1] 0.4417502
```

Question 10

(10 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...)

```r
# computes the volume of n-dimensional sphere of radius 1
calc_vol_sphere = function(radius, n){
  return( (pi^(n/2)) / factorial(n/2) * radius^n )
}
sphere_vols = vector(length=8)
cube_vols = vector(length=8)
# assume our cubes has length 2, so sphere must have radius 1 to be inscribed.
for(i in 3:10){
  cube_vols[i-2] = 2^i
  sphere_vols[i-2] = calc_vol_sphere(1, i)
}
ratios = sphere_vols / cube_vols
for(i in 3:10){
  cat("Dimension", i, "ratio:", ratios[i-2], "\n")
}
```

```
## Dimension 3 ratio: 0.5235988
## Dimension 4 ratio: 0.3084251
## Dimension 5 ratio: 0.1644934
## Dimension 6 ratio: 0.08074551
## Dimension 7 ratio: 0.03691223
## Dimension 8 ratio: 0.01585434
## Dimension 9 ratio: 0.0064424
## Dimension 10 ratio: 0.002490395
```

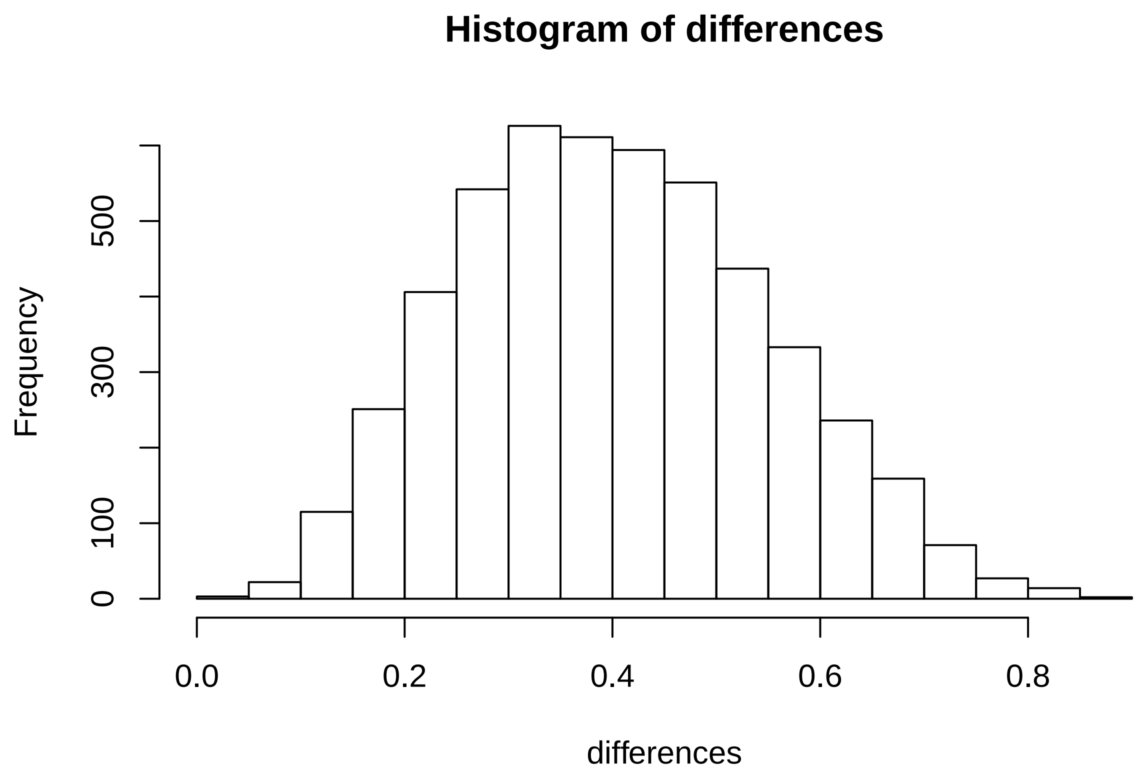
The ratio decreases with dimensionality.

Question 11

(10 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.

```r
set.seed(69)
n = 5000
differences = vector(length=n)
for(i in 1:n){
  sample = runif(9)
  differences[i] = max(sample) - median(sample)
}
hist(differences)
```



```
mean(differences)
```

```
## [1] 0.4018366
```

```
se = function(x) sqrt(var(x)/length(x))
se(differences)
```

```
## [1] 0.002071225
```

Question 12

(10 points)

You are given the following distribution:

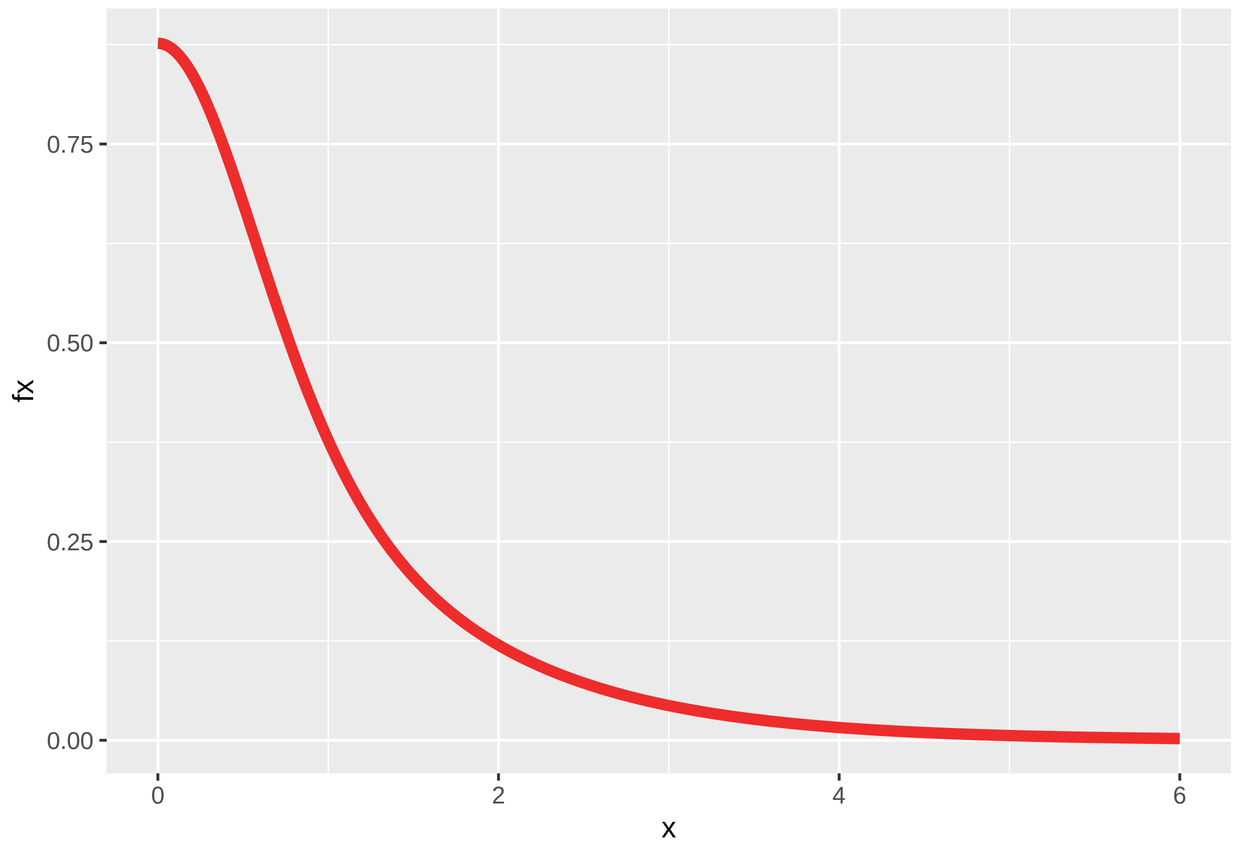
$$f(x) = e^{-x\mathrm{erf}(x)}/1.140741 \qquad x > 0\,.$$

"erf(x)" == the error function with input x. (You'll need to install and load the `VGAM` package to be able to compute the error function.) Here's a plot of $f(x)$:

```
if ( require(VGAM) == FALSE ) {
  install.packages("VGAM",repos="https://cloud.r-project.org")
  library(VGAM)
}
```

```
## Loading required package: VGAM

## Loading required package: stats4

## Loading required package: splines

##
## Attaching package: 'VGAM'

## The following object is masked from 'package:tidyr':
##
##     fill
```

```r
x = seq(0,6,by=0.01)
fx = exp(-x*erf(x))/1.140741
ggplot(data=data.frame(x=x,fx=fx),mapping=aes(x=x,y=fx)) + geom_line(col="firebrick2",size=2)
```



Use importance sampling to estimate the mean of $f(x)$. Use 100,000 data points. Your result should be approximately 0.95. (Hint: a half-normal distribution with `sigma` about 2.5 makes a nice proposal distribution here. See the `extraDistr` package.)

```r
if ( require(extraDistr) == FALSE ) {
  install.packages("extraDistr",repos="https://cloud.r-project.org")
  library(extraDistr)
}
```

```
## Loading required package: extraDistr

##
## Attaching package: 'extraDistr'

## The following objects are masked from 'package:VGAM':
##
##     dfrechet, dgev, dgompertz, dgpd, dgumbel, dhuber, dkumar, dlaplace,
##     dlomax, dpareto, drayleigh, dskellam, dslash, pfrechet, pgev,
##     pgompertz, pgpd, pgumbel, phuber, pkumar, plaplace, plomax,
##     ppareto, prayleigh, pslash, qfrechet, qgev, qgompertz, qgpd,
##     qgumbel, qhuber, qkumar, qlaplace, qlomax, qpareto, qrayleigh,
##     rfrechet, rgev, rgompertz, rgpd, rgumbel, rhuber, rkumar, rlaplace,
##     rlomax, rpareto, rrayleigh, rskellam, rslash

## The following object is masked from 'package:purrr':
##
##     rdunif
```

```r
set.seed(69)
N = 100000
x = rhnorm(N,sigma=2.5)
h = dhnorm(x,sigma=2.5)
g = rep(0,N)
g[x>0] = x[x>0]
f = function(x) {
  f.x = rep(0,length(x))
  f.x[x>0] = dhnorm(x[x>0])
  f.x[x>0] = exp(-x*erf(x))/1.140741
  return(f.x)
}
mean(g*f(x)/h)
```

```
## [1] 0.9496085
```