Lab: Week 6

36-350 – Statistical Computing

Week 6 – Fall 2020

Name: Kimberly Zhang

Andrew ID: kyz

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

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

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## -- Conflicts ------------------------------------------------------------------------------------ tid
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Simulation

Question 1

(5 points)

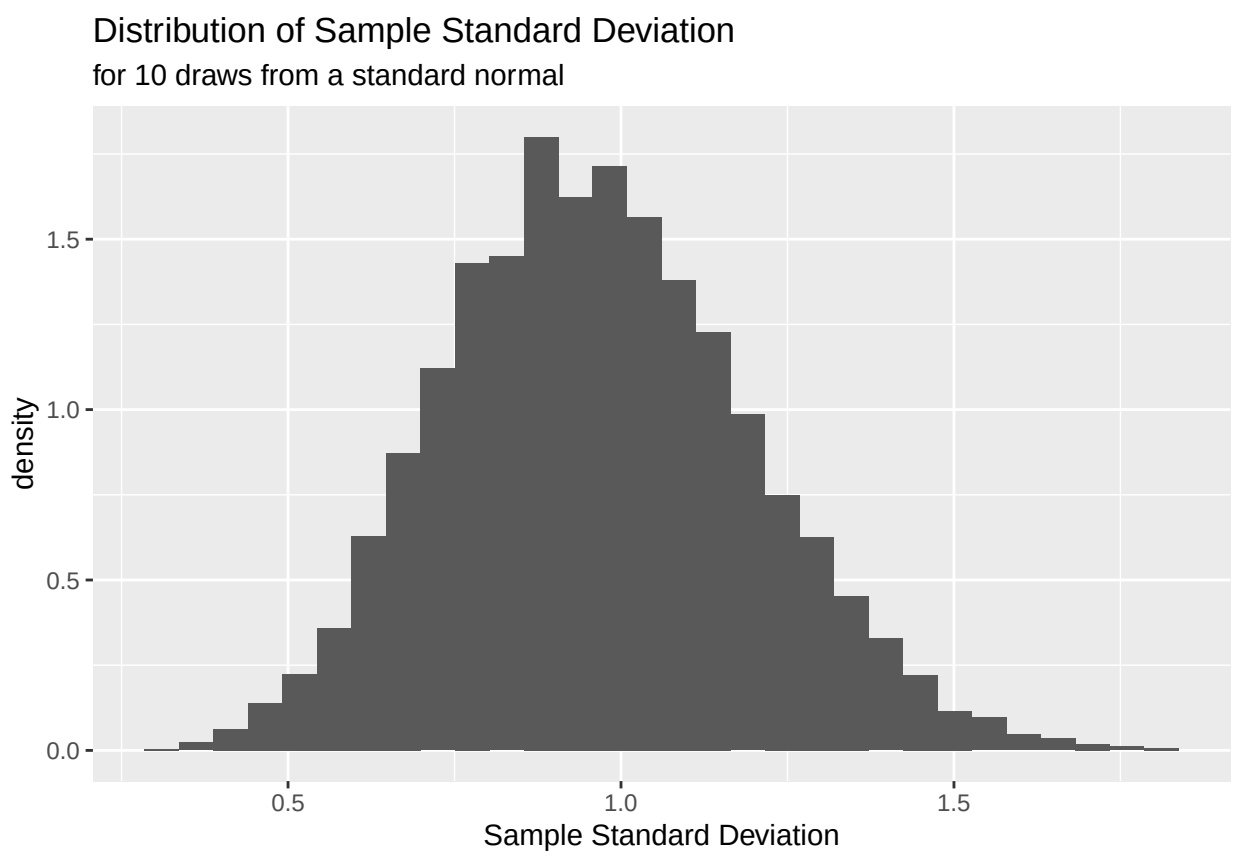
Notes 6A (3-5) and Notes 6C (5)

What is the distribution of the sample standard deviation for 10 draws from a standard normal? Create an empirical distribution by repeating the process of sampling 5000 times. Make a density histogram of your result using `ggplot`. (Pass the argument `aes(y=..density..)` to `geom_histogram()` to change the default frequency histogram into a density histogram). Test the hypothesis that the empirical distribution is itself normal. (Display the p-value.) Google to find an appropriate test of normality. Be sure to set random number seeds here and below for reproducibility. (Also note that `ggplot()` expects a data frame as input, not a vector; you should simply create a one-column data frame from your vector of sample standard deviations and pass that into `ggplot()`.)

```
set.seed(101)
n.obs = 5000
n.data = 10
sample = matrix(rnorm(n.data*n.obs), ncol = n.data)
res = apply(sample, 1, sd)
res.df = data.frame(sample = res)
ggplot(data = res.df, mapping = aes(x = sample)) +
```

```
  geom_histogram(aes(y= ..density..)) +
  labs(x="Sample Standard Deviation",
       title = "Distribution of Sample Standard Deviation",
       subtitle = "for 10 draws from a standard normal")
```

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




```
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.99546, p-value = 2.298e-11
```

```
The distribution should be normal for large enough n.
```

Question 2

(5 points)

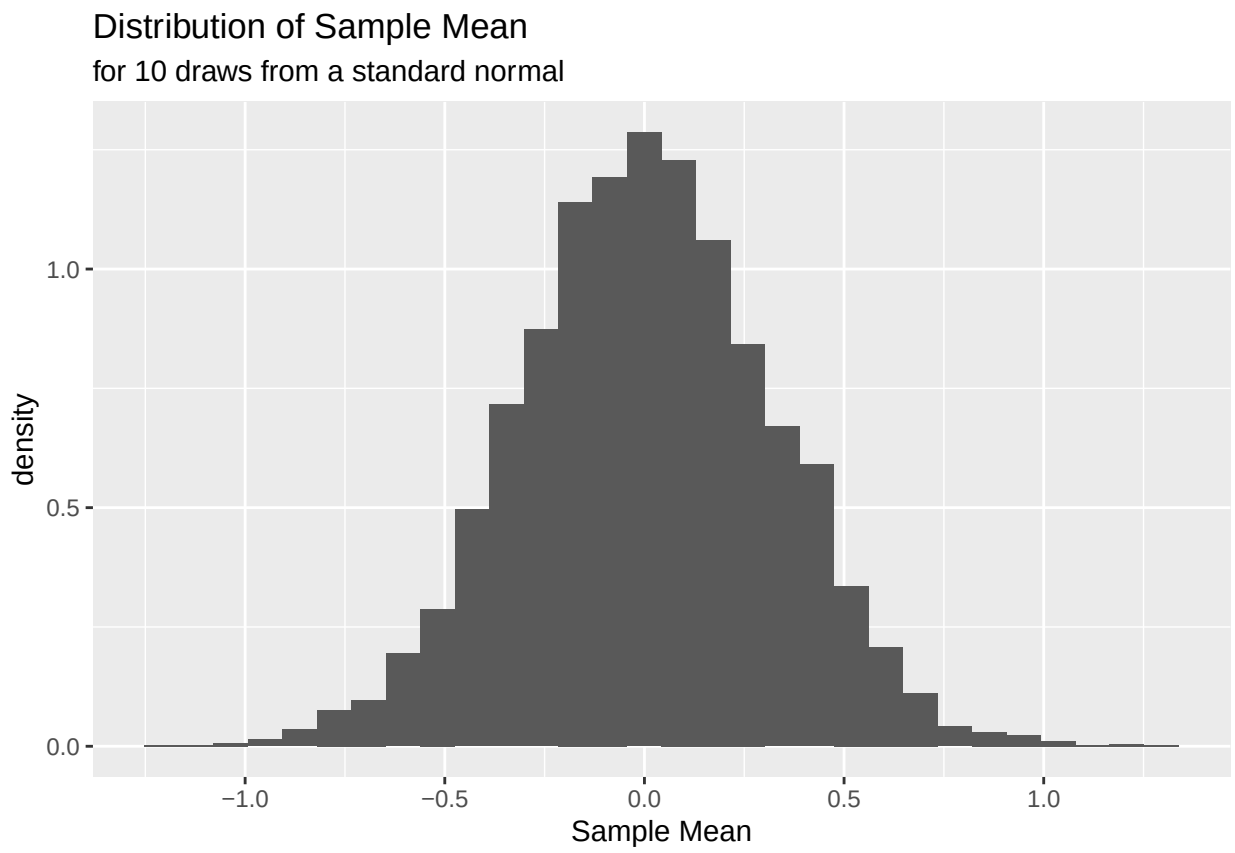
Notes 6A (3-5) and Notes 6C (5)

Repeat the last question, but replace the sample standard deviation with the sample mean. Given the p-value, would you say that the distribution of sample means is closer to being normal than the distribution of sample standard deviations?

```
set.seed(101)
n.obs = 5000
```

```
n.data = 10
sample = matrix(rnorm(n.data*n.obs), ncol = n.data)
res = rowMeans(sample)
res.df = data.frame(sample = res)
ggplot(data = res.df, mapping = aes(x = sample)) +
  geom_histogram(aes(y= ..density..)) +
  labs(x="Sample Mean",
       title = "Distribution of Sample Mean",
       subtitle = "for 10 draws from a standard normal")
```

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




```
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.99947, p-value = 0.165
```

```
Yes, given the p-values, distribution of sample means is closer to being normal
than the distribution of sample standard deviations.
```

Question 3

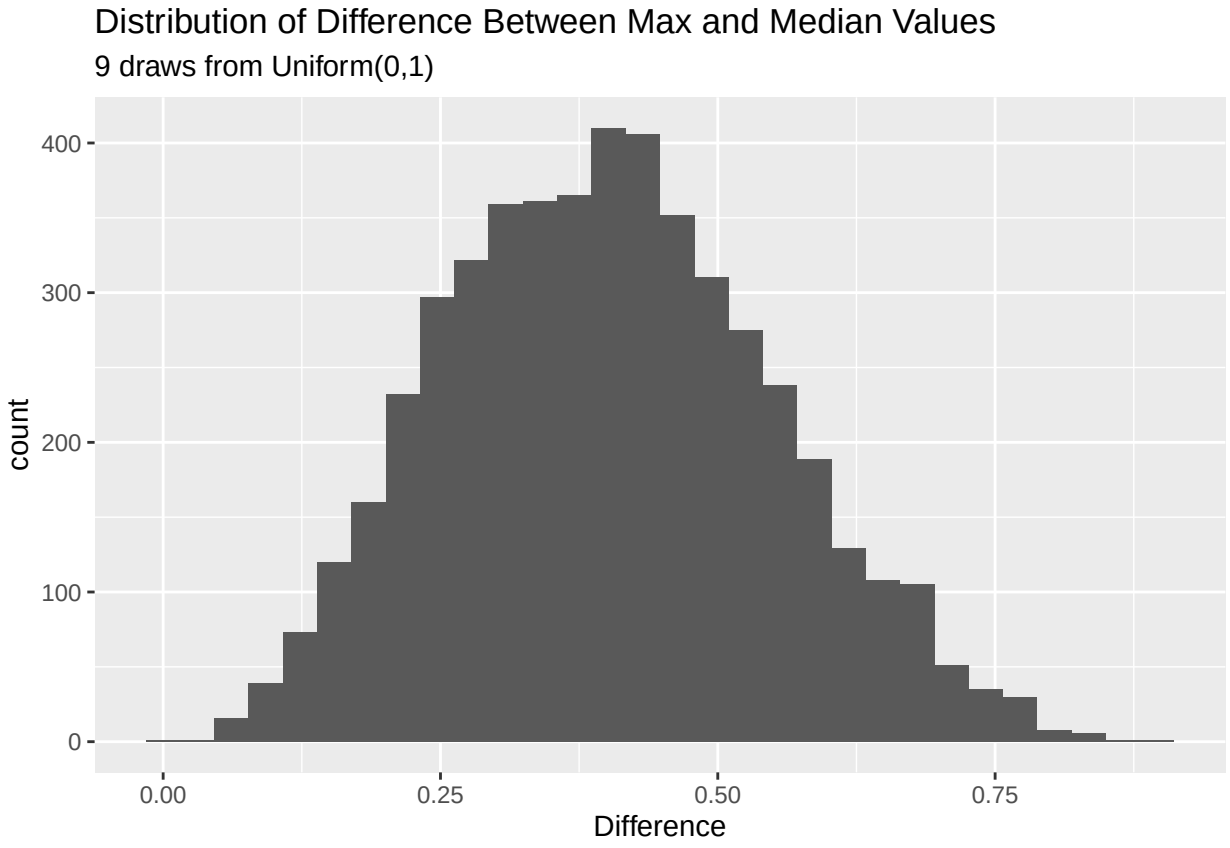
(5 points)

Notes 6A (3-5) and Notes 6C (5)

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.

```
set.seed(101)
n.obs = 5000
n.data = 9
data = matrix(runif(n.data*n.obs), ncol = n.data)
diff = apply(data, 1, max) - apply(data, 1, median)
ggplot(data = data.frame(res = diff), mapping = aes(x = diff)) +
geom_histogram() + labs(
title="Distribution of Difference Between Max and Median Values",
x= "Difference", subtitle = "9 draws from Uniform(0,1)")
```

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




```
mean(diff)
```

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

```
sd(diff)
```

```
## [1] 0.1469405
```

Question 4

(5 points)

Notes 6A (2-5)

Estimate the mean and standard error for the sum of three rolled fair dice using 5,000 simulated rolls. Note that there are potentially several ways by which you might attack this problem; how you go about it is up to you, as long as you get a valid final answer. (Which should be close to 10.5.)

```
set.seed(101)
r1= sample(1:6, size = 5000, replace = TRUE)
r2= sample(1:6, size = 5000, replace = TRUE)
r3= sample(1:6, size = 5000, replace = TRUE)
data = cbind(r1, r2, r3)
res = rowSums(data)
mean(res)
```

```
## [1] 10.5366
```

```
sd(res)
```

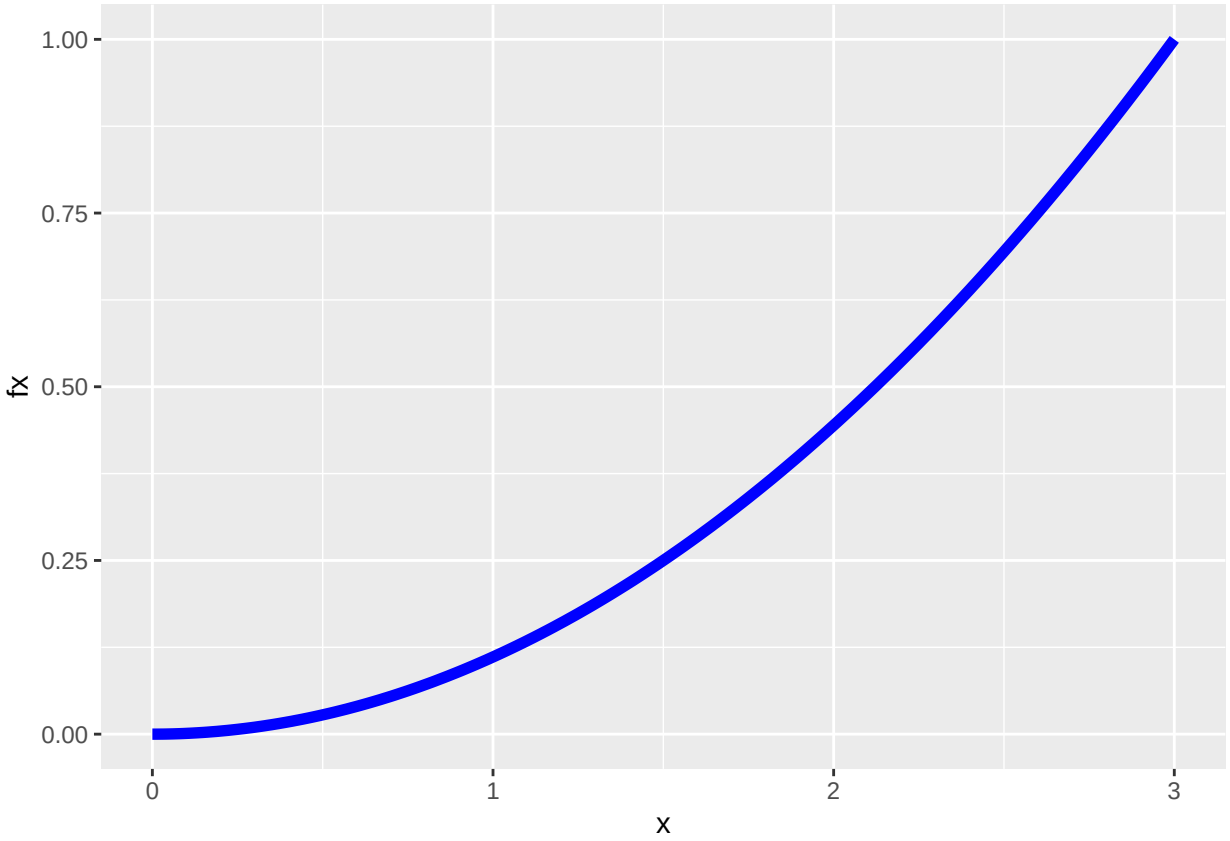
```
## [1] 2.98014
```

You are given the following probability density function:

$$f(x) = \frac{x^2}{9} \quad , \quad x \in [0, 3]$$

It looks like this:

```
x = seq(0,3,by=0.01)
fx = x^2/9
df.pdf = data.frame("x"=x,"fx"=fx)
ggplot(data=df.pdf,mapping=aes(x=x,y=fx)) + geom_line(col="blue",size=2)
```



Question 5

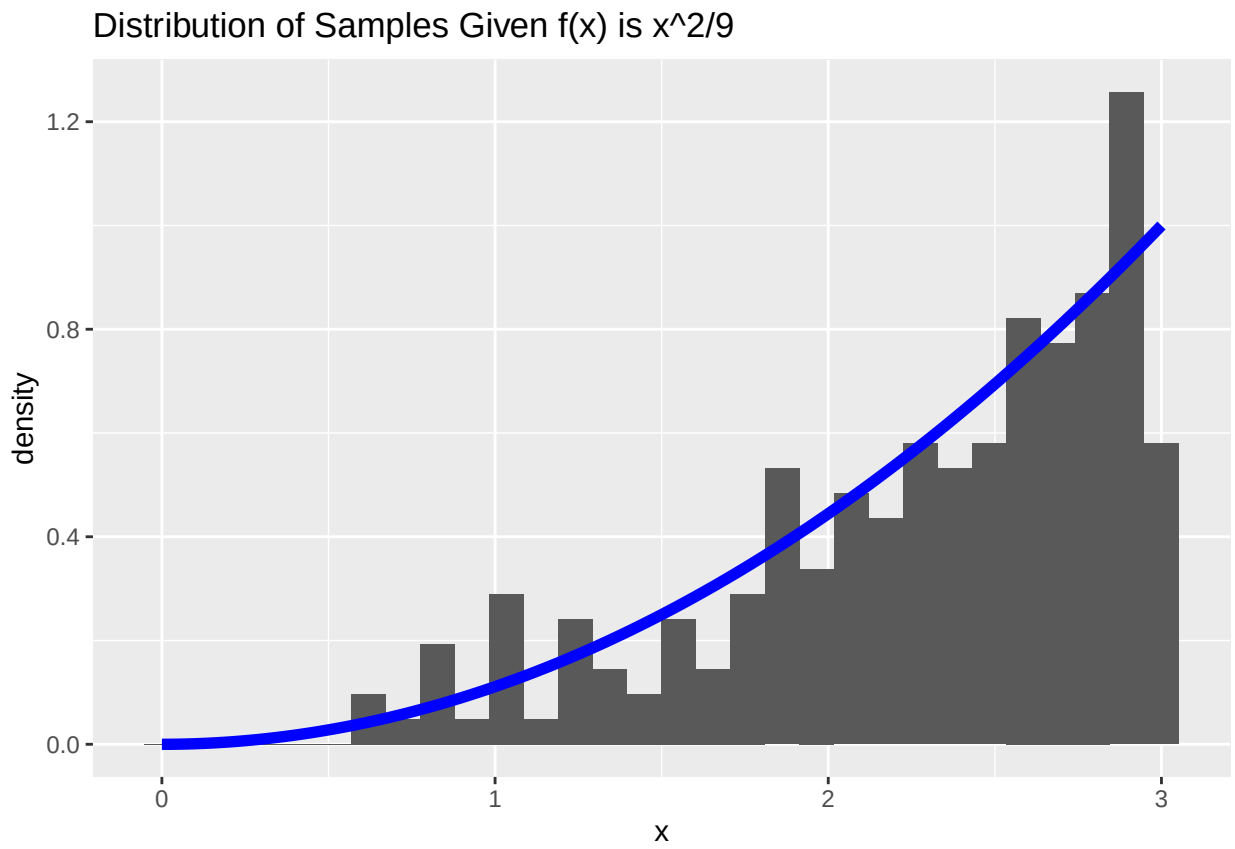
(10 points)

Notes 6A (6-7) and Notes 6C (5)

Code up an inverse transform sampler that allows you to efficiently sample 200 data from this pdf. Histogram your sample, and overlay the line showing the pdf. To do this in `ggplot`, you would use a structure like `ggplot(...) + geom_histogram(...) + geom_line(...)`. The main issue is that the data are histogramming are not the data you would be passing to the line. So: do the `ggplot(...) + geom_histogram(...)` in the same manner you did in previous questions above, where the data frame you point to is the one containing your sampled points, then, when you add on `geom_line(...)`, specify a data argument that points to the `df.pdf` variable defined above and specify a mapping argument that refers to columns of `df.pdf` (and add on a color argument and a size argument like we did above, if you wish).

```
set.seed(101)
u = runif(200)
x = (27*u)**(1/3)
res.df = data.frame(sample = x)
ggplot(data = res.df, mapping = aes(x = sample)) +
geom_histogram(aes(y= ..density..)) +
geom_line(data=df.pdf,mapping=aes(x=x,y=fx), col="blue", size = 2 ) +
labs(title = "Distribution of Samples Given f(x) is x^2/9", x = "x")
```

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



Question 6

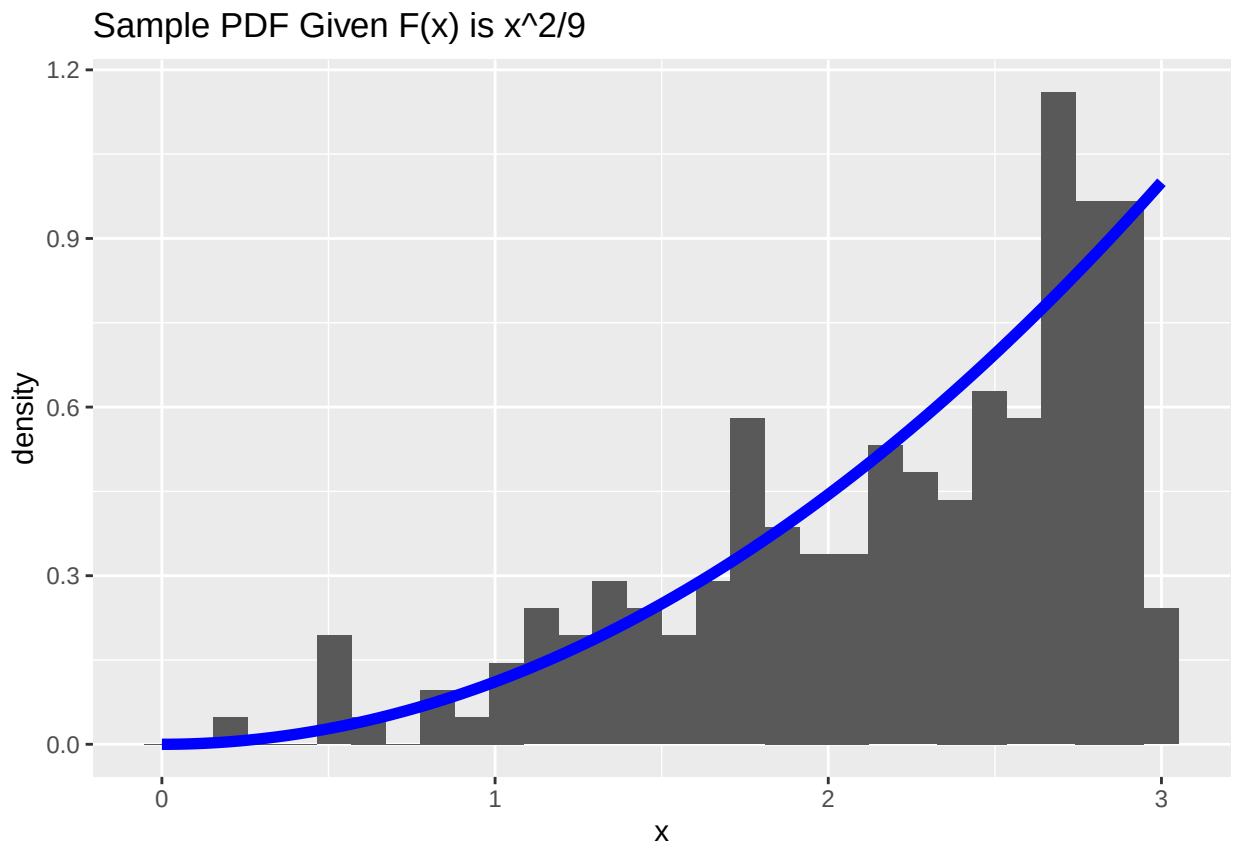
(10 points)

Notes 6A (8-9) and Notes 6C (5)

Repeat the previous question, but utilize rejection sampling.

```
set.seed(101)
k = 200
x = rep(NA, k)
ii = 1
while(ii <= k){
  x[ii] = runif(1, min= 0, max = 3)
  if (runif(1, min = 0, max = 1) < x[ii]^2/9){
    ii = ii + 1
  }
}
res.df = data.frame(sample = x)
ggplot(data = res.df, mapping = aes(x = sample)) +
geom_histogram(aes(y= ..density..)) +
geom_line(data=df.pdf,mapping=aes(x=x,y=fx), col="blue", size = 2 ) +
labs(title = "Sample PDF Given F(x) is x^2/9", x = "x")
```

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



Question 7

(5 points)

Notes 6A (3-5)

In 36-225, you learned that an effective rule of thumb regarding the Central Limit Theorem is that if $n \geq 30$, the mean of your sample is at least approximately normally distributed. Let's test this out. Construct repeated samples of 30 data from an Exponential(1) distribution and record the means: are they normally distributed? Simply show the p-value from an appropriate hypothesis test. If it is $\ll 0.05$, we'll conclude the rule of thumb doesn't really hold in this instance.

```
set.seed(101)
n.obs = 5000
data = matrix(rexp(30*n.obs, rate = 1), ncol = 30)
res = rowMeans(data)
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.99276, p-value = 2.834e-15
```

Question 8

(5 points)

Notes 6A (3-5)

Repeat the last question, but for a Uniform(0,1) distribution.

```
set.seed(101)
n.obs = 5000
data = matrix(runif(30*n.obs), ncol = 30)
res = rowMeans(data)
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.99958, p-value = 0.3698
```

Question 9

(5 points)

Notes 6A (2-5)

You have been called to testify in a trial on the alien planet Slybobia. Prosecutors allege that on this planet, whose intelligent population is 25% blue, 25% green, 25% orange, and 25% fuchsia, it would require bias to construct a panel of 10 Slybobians in which at least 4 are orange *and* at least 4 are fuchsia. But such a panel had been constructed. Your expert testimony is needed: what is the probability that such a panel would be constructed if sampling of the population was truly done randomly? Is it smaller than 0.05? If so, that would indicate that you should reject the null hypothesis that the panel composition is unbiased.

```
set.seed(101)
# 1 = blue, 2 = green, 3= orange, 4= fuchsia
n = 10000
counter = 0
for (i in 1:n) {
  s = sample(1:4, size= 10, replace = TRUE)
  if (sum(s == 3) >=4 && sum(s==4) >=4){
    counter = counter + 1
  }
}
counter/n
```

```
## [1] 0.0173
```

```
Because the probability is smaller than 0.05, we reject the null hypothesis.
There is evidence that the panel composition is biased.
```

Question 10

(5 points)

Notes 6A (8-9)

An old classic: what is the value of π? Use rejection sampling in two dimensions (and, oh, 10,000,000 samples) to estimate π. (Think of a unit circle inscribed within a 2×2 box centered at the origin. The box will have an area of 4, and the circle will have an area of π... thus the ratio of the total number of random samples inside the unit circle to the total number of samples will approach $\pi/4$ as the number of samples approaches ∞.) Also display the percentage error of your estimate, which you can compute using `R`'s built-in value of π (i.e., the `R` constant `pi`.)

```r
set.seed(101)
k = 10000000
res = rep(0, k)
ii = 1
while(ii <= k){
x = runif(1)
y =  runif(1)
r2 = x^2+y^2
if (r2 <= 1){
  res[ii] = 1
}
ii = ii + 1
}
estimate = mean(res)*4
per_error = abs(estimate - pi)/pi*100
estimate
```

```
## [1] 3.141648
```

```r
per_error
```

```
## [1] 0.001761731
```

Question 11

(10 points)

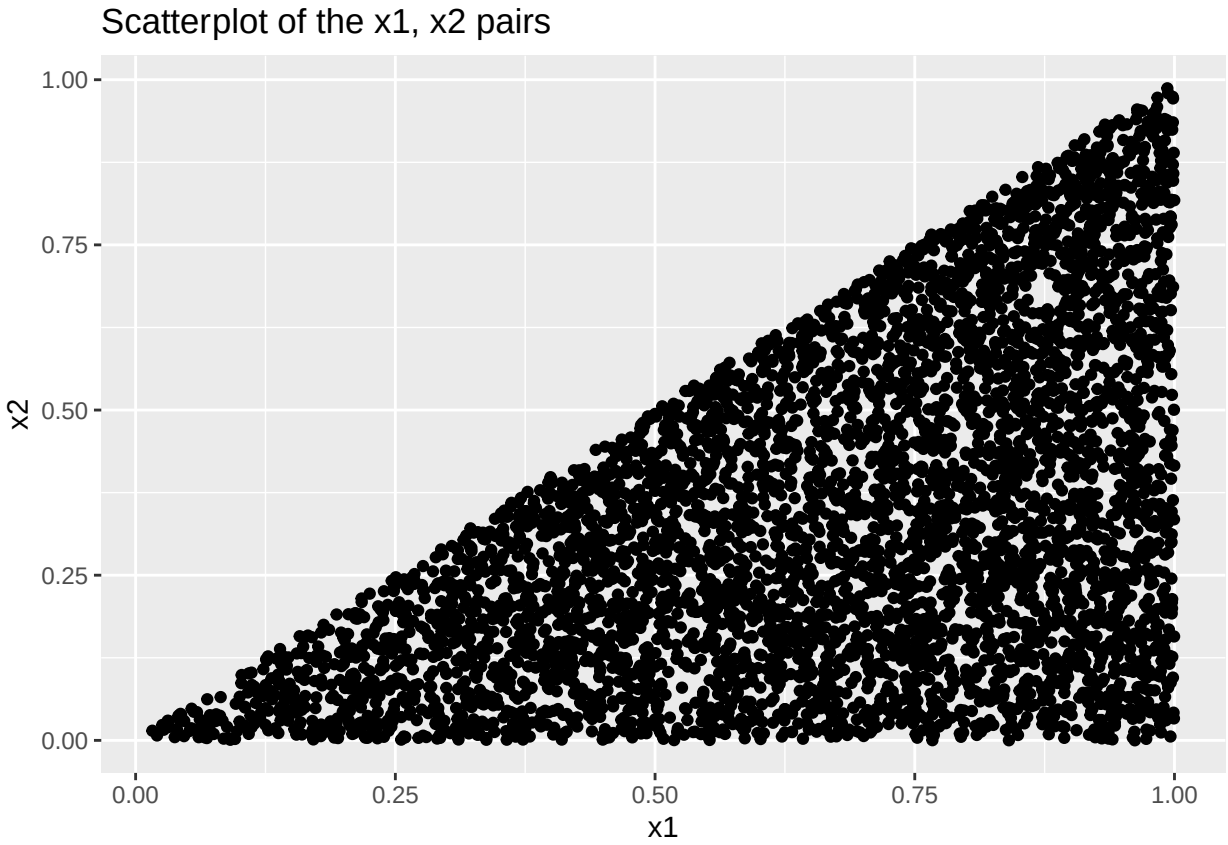
Notes 6A (6-9)

You are given the following bivariate pdf:

$$f(x_1, x_2) = \begin{cases} 3x_1 & 0 \leq x_2 \leq x_1 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

(This is borrowed from Exercise 5.5 of Wackerly 7.) Code a function for sampling data from this distribution. Sample 5000 (x_1, x_2) pairs, then visualize them with a scatter plot. You should observe a greater density of points as you go from the left to the right.

```r
set.seed(101)
k = 5000
x1 = rep(NA, k)
x2 = rep(NA, k)
ii = 1
while(ii <= k){
  x1[ii] = runif(1)
  x2[ii] = runif(1)
  if (x2[ii] <= x1[ii]){
    ii = ii + 1
  }
}
res= data.frame(x= x1, y=x2)
ggplot(data=res,mapping=aes(x=x,y=y)) + geom_point() +
  labs(title= "Scatterplot of the x1, x2 pairs", x= "x1", y= "x2")
```



Integration

Question 12

(5 points)

Notes 6B (3-4)

Compute the integral

$$\int_0^3 x^2 e^{-x^4} dx$$

via Monte Carlo integration, with 100,000 points. You should achieve a value close to 0.30635.

```
set.seed(350)
g = function(x) {
  x^2*exp(-x^4)
}
x = runif(100000, min = 0, max= 3)
w = g(x)/(1/3)
mean(w)
```

```
## [1] 0.3053488
```

Question 13

(5 points)

Notes 6B (3-4)

Compute the integral

$$\int_0^1 \int_0^1 \left(\cos\left(\frac{\pi}{2}x_1\right) + x_2^3\right) dx_1 dx_2$$

via Monte Carlo integration, with 1,000,000 points. Your value should be close to 0.8866.

```
set.seed(350)
g = function(x1, x2) {
  cos(pi/2*x1)+x2^3
}
n = 1000000
x1 = runif(n)
x2 = runif(n)
w = g(x1, x2)
mean(w)
```

```
## [1] 0.886682
```

Question 14

(5 points)

Notes 6A (8-9) and Notes 6C (3-4)

Let's change up that last integral just a bit:

$$\int_0^1 \int_0^{x_2} \left(\cos\left(\frac{\pi}{2}x_1\right) + x_2^3\right) dx_1 dx_2$$

The region of integration is now the triangle with vertices (0,0), (0,1), and (1,1). (Commence hallucinatory flashbacks to 225, if they haven't started already.) Combine a rejection sampler and MC integration to perform this integral. Sample approximately 1,000,000 points in the region of integration and then use those. (By using the word "approximately," I'm encouraging you to be clever in how you do the sampling... like by sampling 2,000,000 points in a box and keeping the roughly 1,000,000 that lie within the triangular region of integration.) Realize that for a bivariate uniform within the region of integration, $f(x_1, x_2) = 2$. (Your answer should be approximately 0.605.)

```
set.seed(101)
n = 1000000
g = function(x1, x2) {
  cos(pi/2*x1)+x2^3
}
x1 = rep(NA, n)
x2 = rep(NA, n)
ii = 1
while(ii <= n){
  x2[ii] = runif(1)
  x1[ii] = runif(1)
  if (x1[ii] <= x2[ii]){
    ii = ii + 1
  }
}
w = g(x1, x2)/2
mean(w)
```

```
## [1] 0.6051605
```

Question 15

(10 points)

Notes 6C (5-8)

You are given the following distribution:

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

"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
```

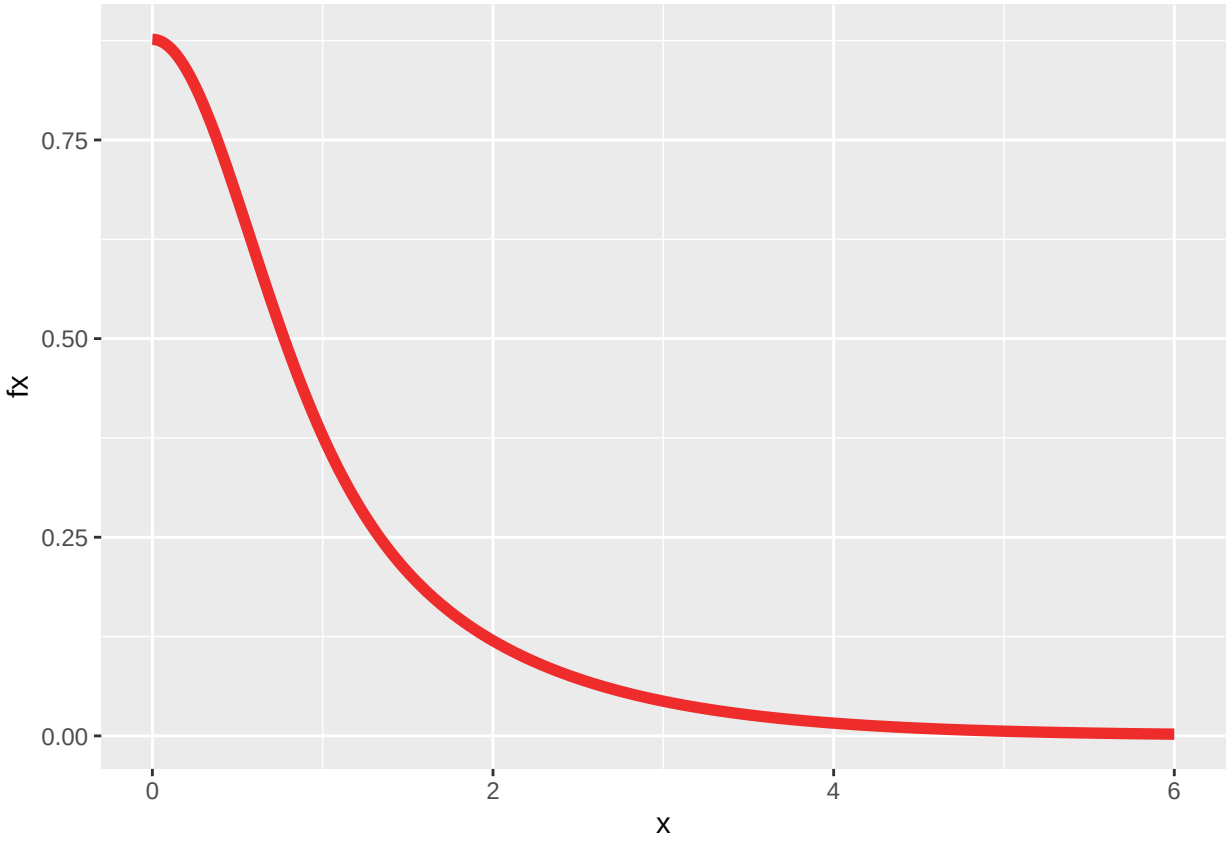
```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
##
## Attaching package: 'VGAM'
```

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

```
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.)

```
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(101)
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) {
  exp(-x*erf(x))/1.140741
}
mean(g*f(x)/h)
```

```
## [1] 0.9503769
```

Question 16

(5 points)

Notes 6C (7)

And now: estimate the standard deviation of $f(x)$. Note: you can reuse many of the variables from above! Your value should again be around 0.95.

```r
exp.x = mean(g*f(x)/h)
x.square = x^2
g = rep(0, n)
g[x>=0] = x.square[x>=0]
exp.x.square = mean(g*f(x)/h)
sqrt(exp.x.square - exp.x^2)
```

```
## [1] 0.951273
```