Lab-6: Assignment

Dr. Purna Gamage

Problem 1 :

Discuss these following examples with your class mates and explain each case and comment on the results.

a. Throwing Dice as Multinomial Distribution

A multinomial distribution is a distribution that shows the likelihood of the possible results of a experiment with repeated trials in which each trial can result in a specified number of outcomes that is greater than two. A multinomial distribution could show the results of tossing a dice, because a dice can land on one of six possible values. By contrast, the results of a coin toss would be shown using a binomial distribution because there are only two possible results of each toss, heads or tails.

Two additional key characteristics of a multinomial distribution are that the trials it illustrates must be independent (e.g., in the dice experiment, rolling a five does not have any impact on the number that will be rolled next) and the probability of each possible result must be constant (e.g., on each roll, there is a one in six chance of any number on the die coming up).

b. Rolling a die N=100 times

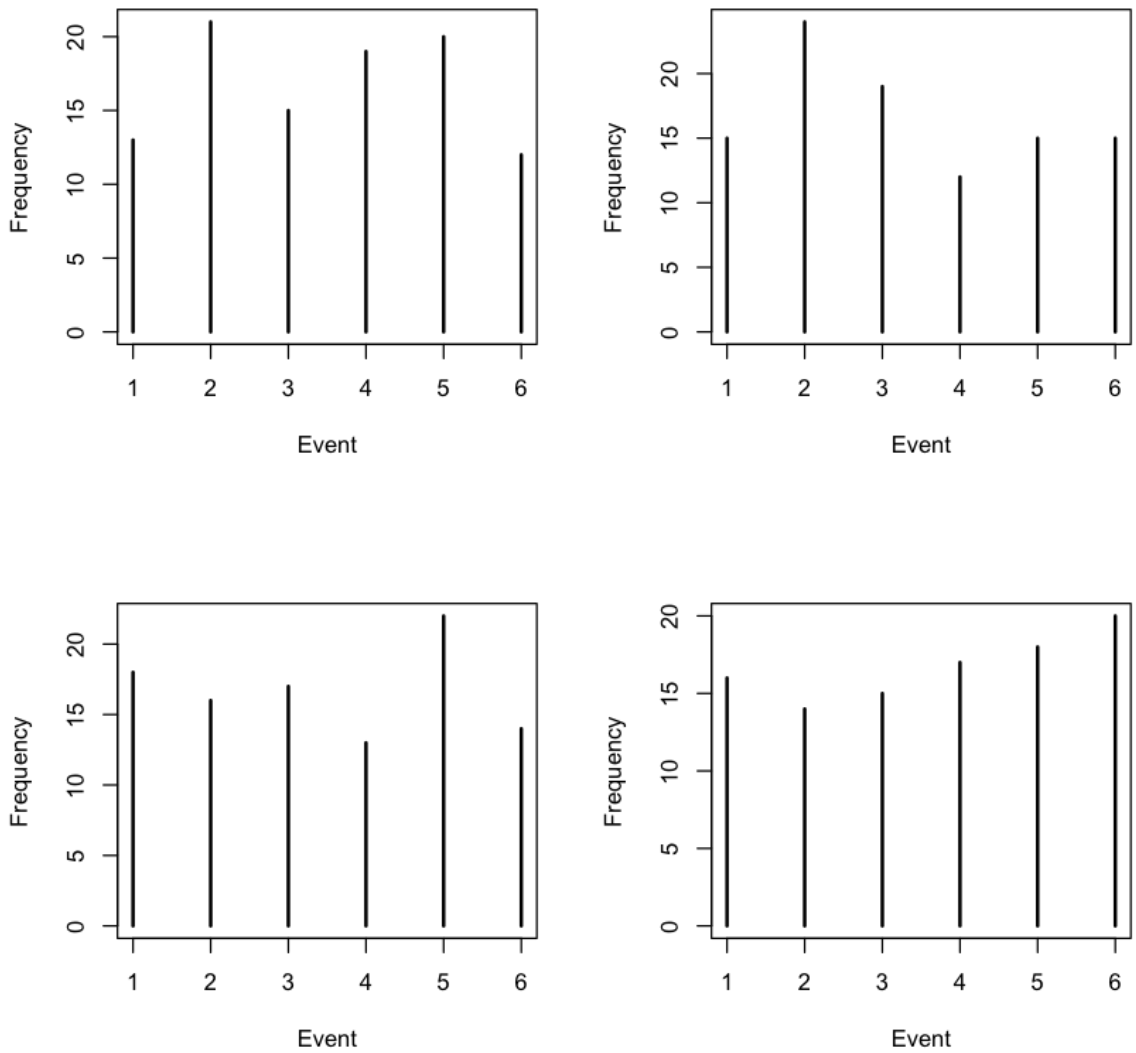
what is happening here?

```
one.dice <- function() {
  dice <- sample(1:6, size = 1, replace = TRUE)
  return(dice)
}
one.dice() # what is happening here?? try this several times.
```

2

The one.dice() calls the one.dice function, and the function generates one random number from 1 to 6.

```
# what is hapening here?

par(mfrow=c(2,2))

for (i in 1:4){
    sims <- replicate(100, one.dice())
    table(sims)
    table(sims)/length(sims)
    plot(table(sims), xlab = 'Event', ylab = 'Frequency')
}
```



The par() funciton creates 2 x 2 plot, and mfrow allows to create multiple figures in a row.

The for loop creates 4 graphs, and the replicate function runs one.dice() function 100 times and assign the 100 values in the 'sims' variable.

Now, 'sims' variable contains 100 integers that are randomely generated by one.dice() function.

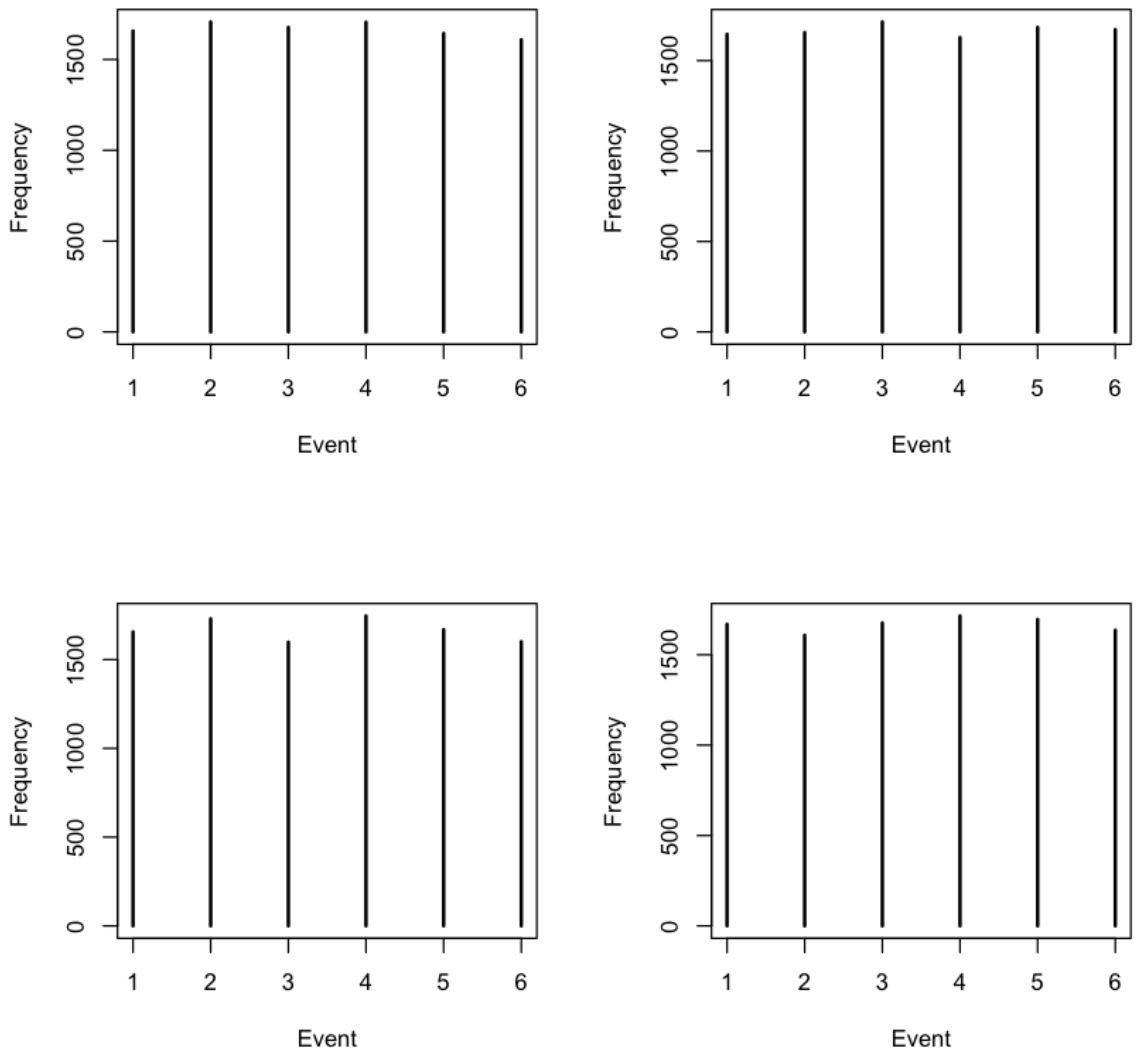
The table function gets the frequency of each value in sims variable.

Then, we get the percentage by running the table(sims)/length(sims) and plot them.

c. Rolling a die N=10000 times.

what is happening here?

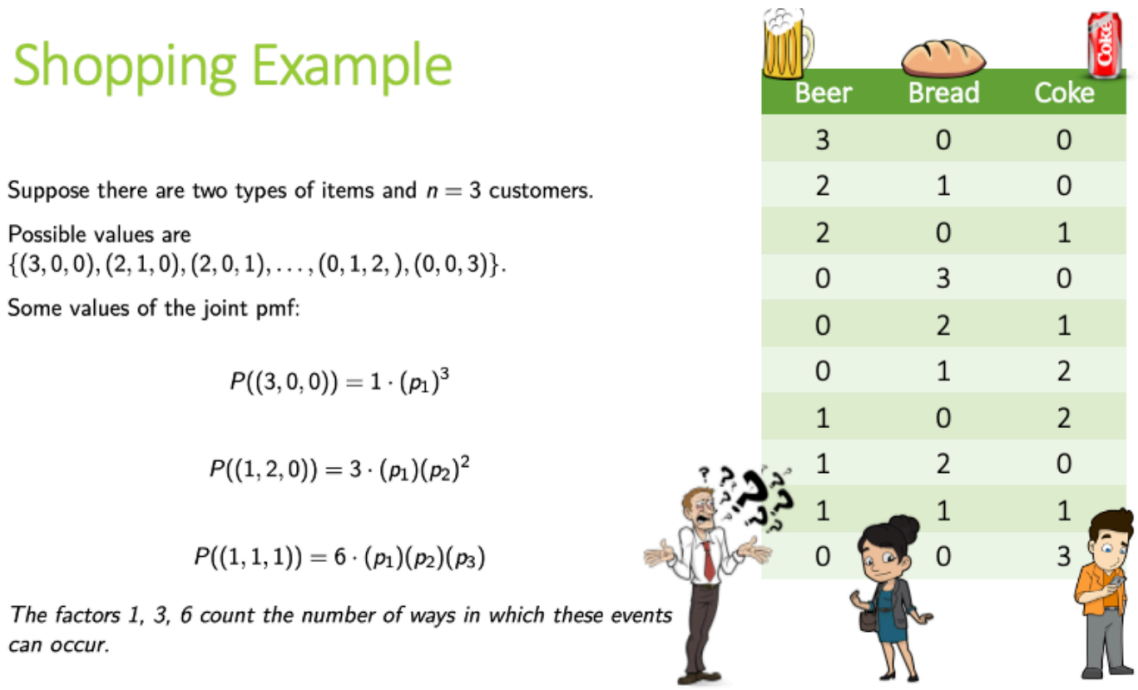
```
#what is hapening here?

par(mfrow=c(2,2))

for (i in 1:4){
    sims <- replicate(10000, one.dice())
    table(sims)
    table(sims)/length(sims)
    plot(table(sims), xlab = 'Event', ylab = 'Frequency')
}
```



This code runs 10000 times, and 10000 randomly generated values are assigned to 'sims' variable, which gives us more frequency and plot the 4 graphs.

Problem 2. Multinomial distribution and its Marginals

From the class example



Let's say that Molly, Ryan and Mr.Bob are buying beer(x1) ,bread(x2) and coke(x3) with probabilities (3/5,1/5,1/5).

a. What is the probability that only 1 of them will buy beer, 2 of them will buy Bread , none will buy coke ? Compare the result with theoretical probability.

```
prob_1 <- 3*(3/5)*(1/5)*(1/5)
prob_1
```

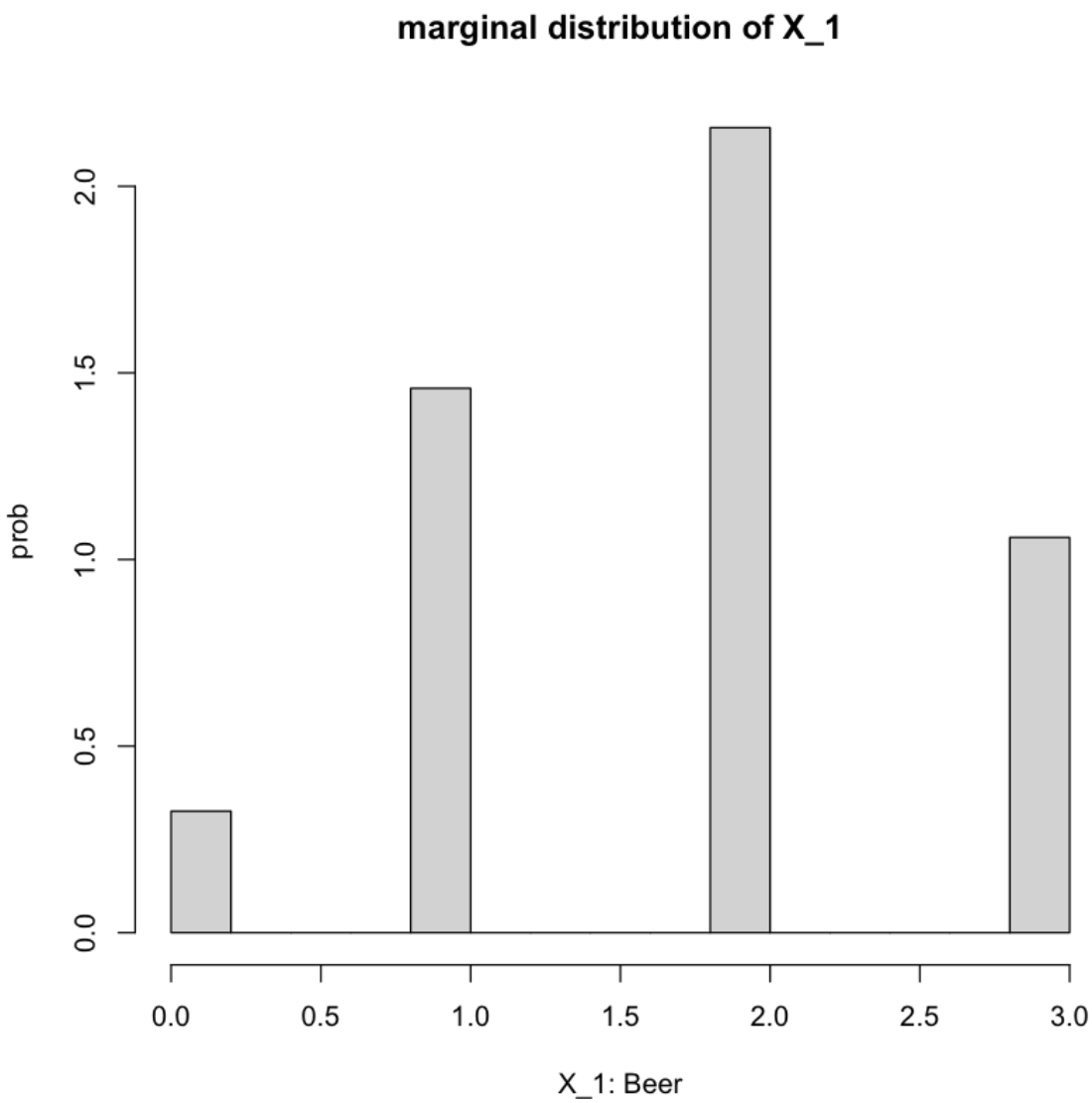
0.072

b. Do a simulation for this scenario and plot the marginal distribution of x1.

```
b_prob <- c(0.6, 0.2, 0.2)
sim <- 10000
samples <- 3 # how many to take

df <- as.data.frame(rmultinom(n=sim, size=samples,prob=b_prob))
df <- t(df)

# change column nams
colnames(df) <- c('Beer', 'Bread', 'Coke')
```

```r
# make it to dataframe
rownames(df) <- NULL
df <- as.data.frame(df)
# check beer == 1, bread == 2
# divide it to sims
prob_2 <- length(df[df$Beer == 1 & df$Bread == 2,]$Beer)/sim
prob_2

# plot the marginal distribution of X_1
hist(df$Beer, prob = prob_2, xlab = 'X_1: Beer', ylab='prob', main='marginal distribution
```

0.0729



Compare both theoretical probability and the scenario probability

```
compare <- prob_1/prob_2
compare
```

0.987654320987654

Problem 3:

Discuss this with your class mates and comment on the Plots. What can you observe from each plot?

Helpful links to answer this question:

-> Contour plot also gives the densities.

https://blog.revolutionanalytics.com/2016/02/multivariate_data_with_r.html

-> Then we have these ellipses; the circular symmetric version of complex normal distribution. https://en.wikipedia.org/wiki/Elliptical_distribution

ellipse: https://en.wikipedia.org/wiki/Ellipse

-> http://cs229.stanford.edu/section/gaussians.pdf

This tells you how when correlation coefficient increases the distribution spread and how the ellipses look like. -> https://online.stat.psu.edu/stat505/book/export/html/636

```
library(tidyverse)
library(mvtnorm)
library(plotly)
library(MASS)
library(ggplot2)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom

Attaching package: 'plotly'


The following object is masked from 'package:ggplot2':

    last_plot
```

```
The following object is masked from 'package:stats':

    filter


The following object is masked from 'package:graphics':

    layout



Attaching package: 'MASS'


The following object is masked from 'package:plotly':

    select


The following object is masked from 'package:dplyr':

    select
```

Source: https://data-se.netlify.app/2018/12/13/visualizing-a-multivariate-normal-distribution/

Simulate multivariate normal data

First, let's define a covariance matrix Σ:

```r
sigma <- matrix(c(4,2,2,3), ncol = 2)
sigma
```

A matrix: 2 x 2 of type dbl

4 | 2 |
2 | 3 |

Then, simulate observations n = n from these covariance matrix; the means need be defined, too. As the rank of our covariance matrix is 2, we need two means:

```r
means <- c(0, 0)
n <- 1000

set.seed(42)
x <- rmvnorm(n = n, mean = means, sigma = sigma)
str(x)
head(x)
```

```
 num [1:1000, 1:2] 2.314 1.053 0.716 2.848 3.839 ...
```

A matrix: 6 x 2 of type dbl

2.3139150	-0.15442375
1.0527522	1.24094662
0.7162789	0.05340542
2.8479495	0.69465309
3.8388378	1.03195246
3.7900042	4.47972608

You can see that X is bivariately normal distributed.

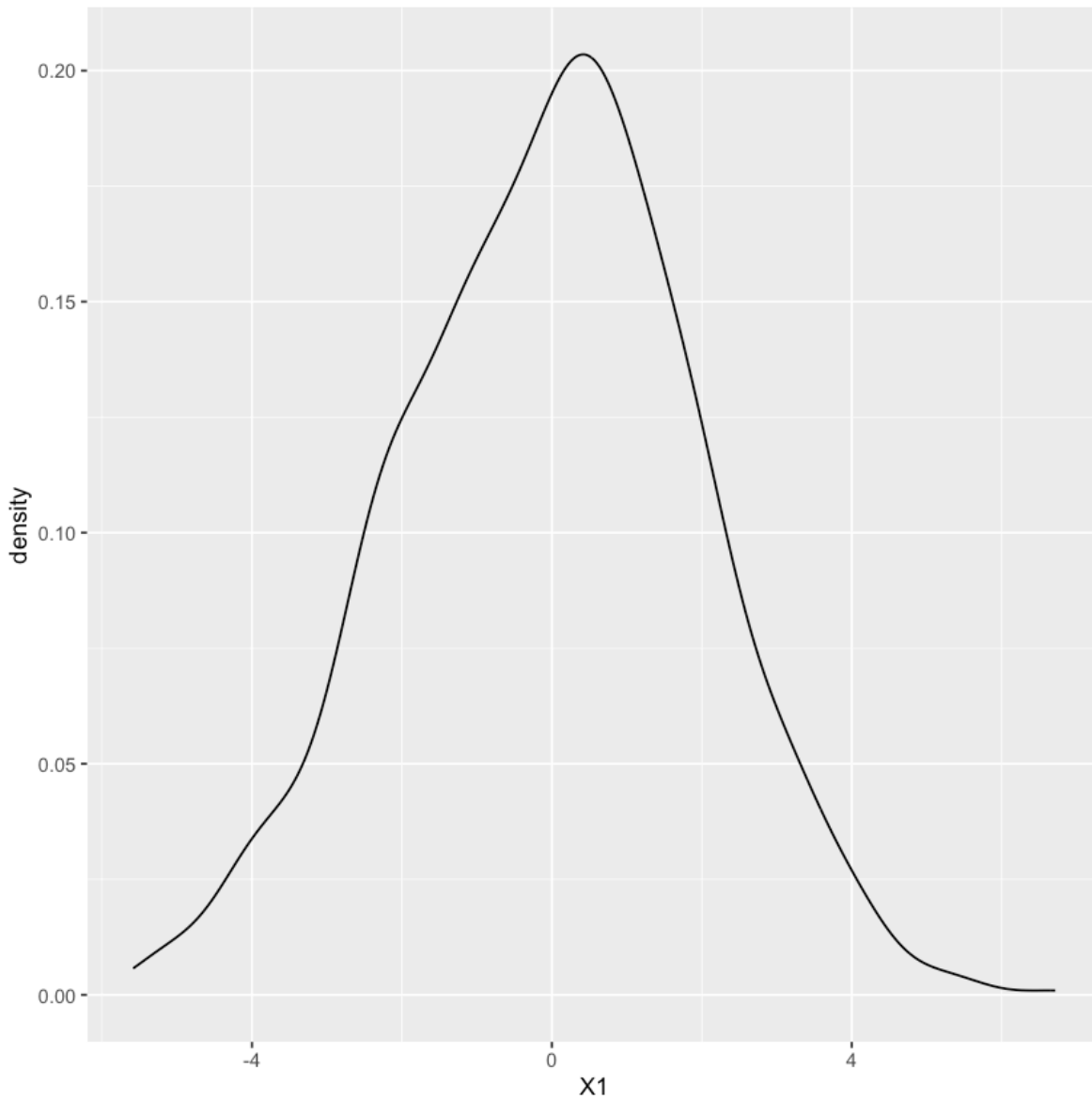
Let's make a data frame out of it:

```r
d <- data.frame(x)
names(d)
```

1. 'X1'
2. 'X2'

a. Plotting univariate (sampled) normal data

```r
## marginal of X1
d %>%
  ggplot(aes(x = X1)) +
  geom_density()
```



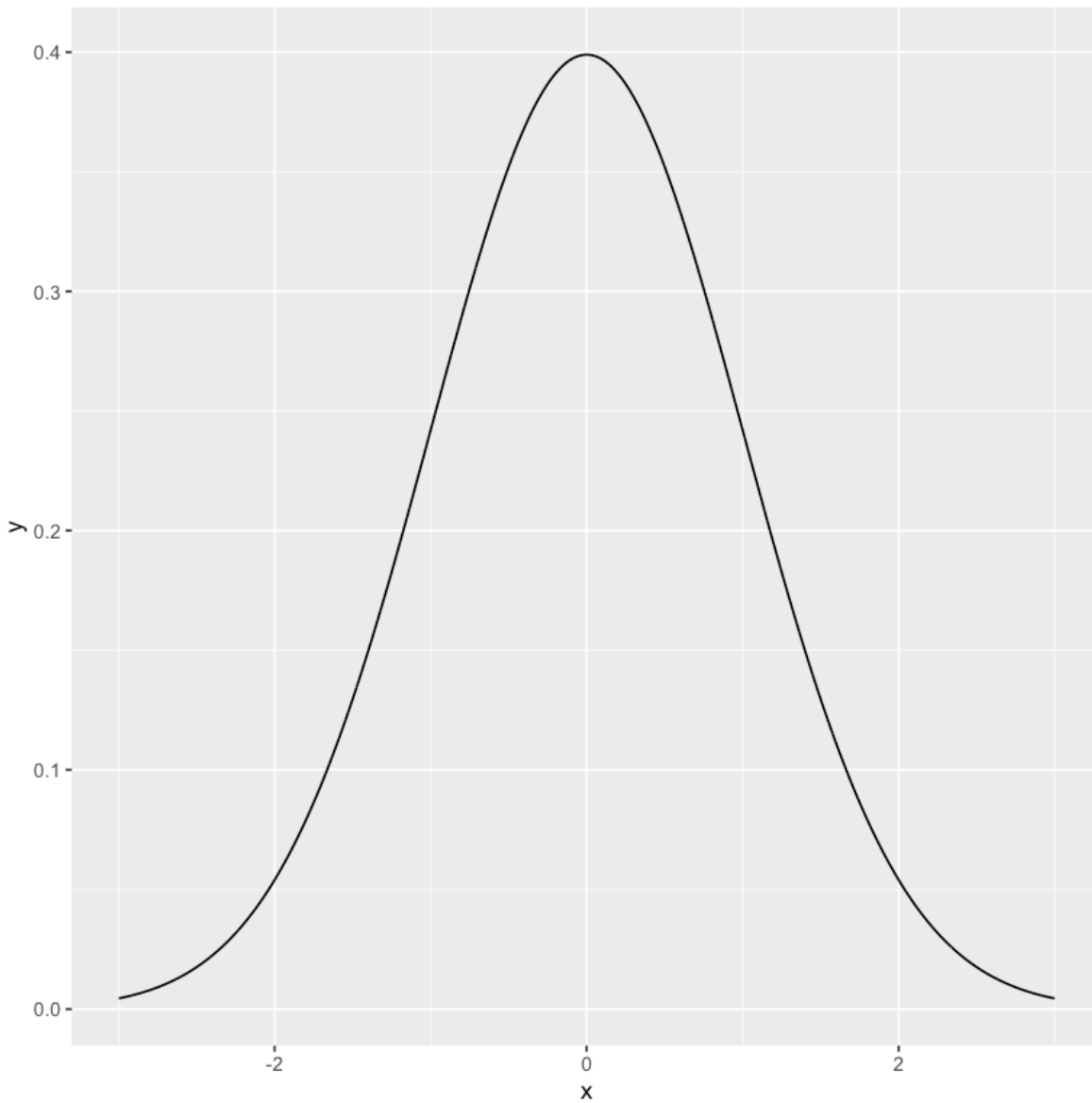
INSERT DISCUSSION COMMENTS

This is the graph of one variable, which is also called 'univariate'. After creating the normal distribution for both X_1 and X_2 by 'rmvnorm()' function, the graph is a plot of density graph for X_1, which is also a plot of marginal distribution of X_1.

b. Plot theoretic normal curve and compare with the above marginal distribution of X1.

```
p1 <- data_frame(x = -3:3) %>%
  ggplot(aes(x = x)) +
  stat_function(fun = dnorm, n = n)

p1
```

```
Warning message:
"`data_frame()` was deprecated in tibble 1.1.0.
i Please use `tibble()` instead."
```



INSERT DISCUSSION COMMENTS

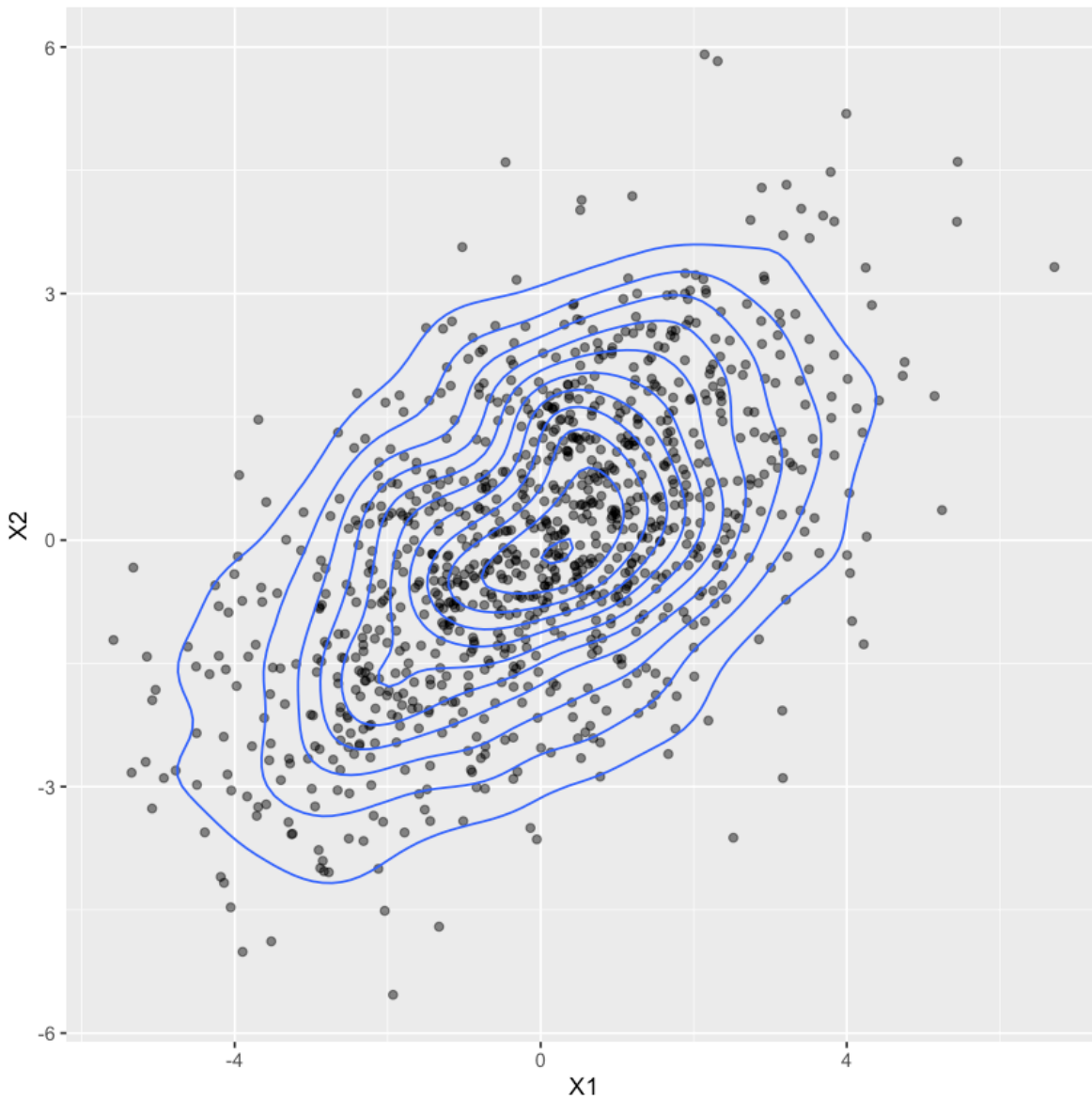
The 'dnorm()' function computes the probability density function(pdf) and specifically computes the density of standard normal distribution at the values specified in X_1.

Comparing to the plot above, which is created my 'rmvnorm()' function, both graphs are almost similar to each other. The graph created by rmvnorm() is a definitely not smoother than the theoretic normal curve, but regardless of the range of the X and Y values, it is clear that both graph are similar to each other.

Plotting multivariate data

c. 2D density

```
p2 <- ggplot(d, aes(x = X1, y = X2)) +
  geom_point(alpha = .5) +
  geom_density_2d()

p2
```



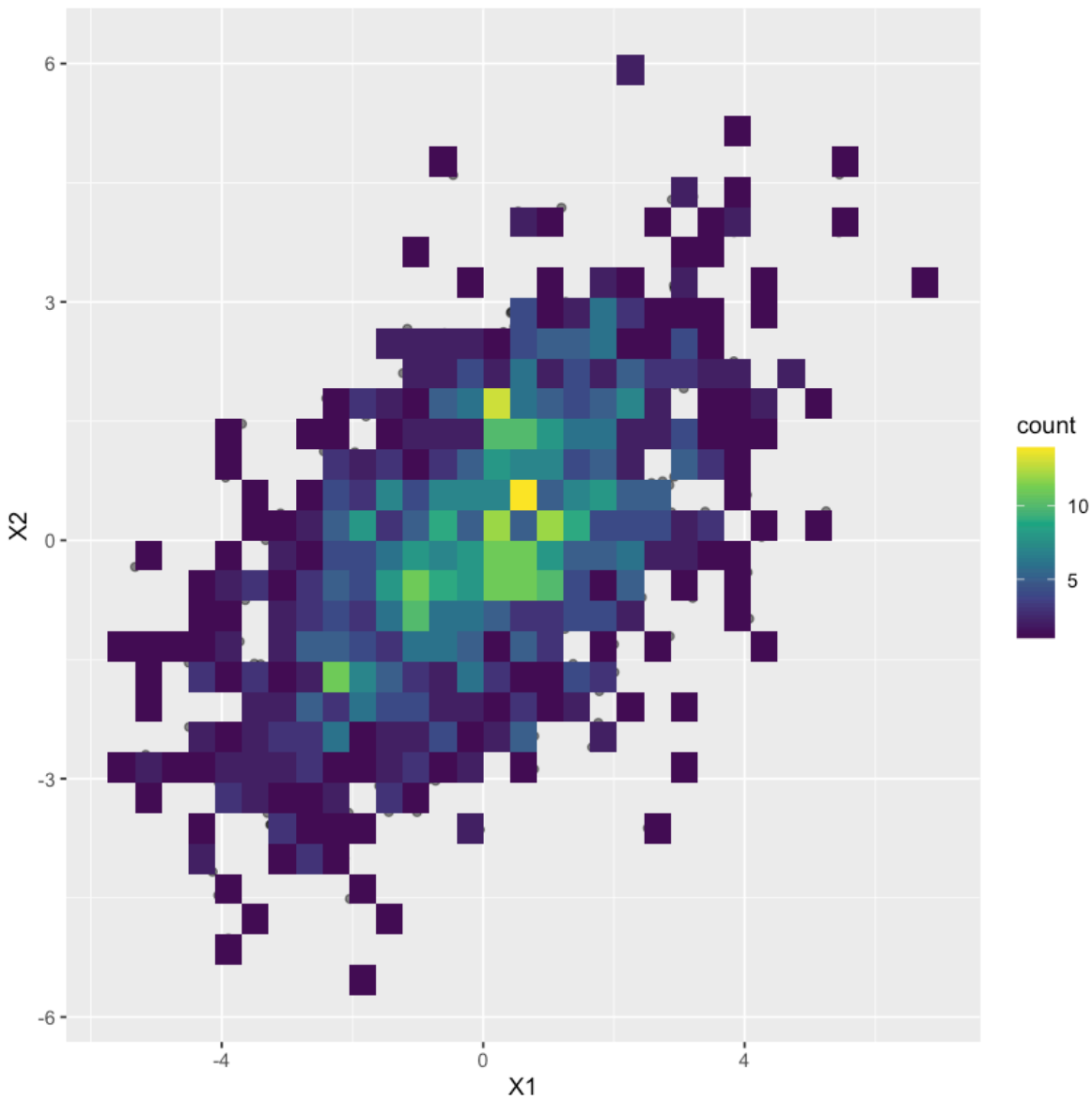
INSERT DISCUSSION COMMENTS

The graph above is a plot the density distribution of both X_1 and X_2 variable: multivariate normal data.

e. Contour plot

Geom binhex https://ggplot2.tidyverse.org/reference/geom_hex.html

```r
p3 <- ggplot(d, aes(x = X1, y = X2)) +
  geom_point(alpha = .5) +
  geom_bin2d() +
  scale_fill_viridis_c()

p3
```



INSERT DISCUSSION COMMENTS

This is a heatmap graph and gives us more specific view of both X_1 and X_2 variables with filled-in colors. This can aesthetically give us the information of the counts of the values. For example, there are more values nearby $(X_1, X_2) = (0.2, 0.2)$.

f. 2D scatter plot and heatmap with plotly

```
(p <- plot_ly(d, x = ~X1, y = ~X2))
```

```
No trace type specified:
  Based on info supplied, a 'scatter' trace seems appropriate.
  Read more about this trace type -> https://plotly.com/r/reference/#scatter

No scatter mode specifed:
  Setting the mode to markers
  Read more about this attribute -> https://plotly.com/r/reference/#scatter-mode

No trace type specified:
  Based on info supplied, a 'scatter' trace seems appropriate.
  Read more about this trace type -> https://plotly.com/r/reference/#scatter

No scatter mode specifed:
  Setting the mode to markers
  Read more about this attribute -> https://plotly.com/r/reference/#scatter-mode

HTML widgets cannot be represented in plain text (need html)
```

```
add_histogram2d(p)
```

```
HTML widgets cannot be represented in plain text (need html)
```

INSERT DISCUSSION COMMENTS

First graph is a scatter plot of 1000 variable for both X_1 and X_2. The heatmap graph gives us the density of the values, which means how many values are in distributed in a certain range.

g. 2D contour with plotly

```
add_histogram2dcontour(p)
```

```
HTML widgets cannot be represented in plain text (need html)
```

INSERT DISCUSSION COMMENTS

This graph also shows us how both variables: X_1 and X_2 are distributed, but the difference between the heatmap graph is that this graph is has visualized the contour lines of 2D density of points.

h. 3D plot: Surface

```
dens <- kde2d(d$X1, d$X2)

plot_ly(x = dens$x,
        y = dens$y,
        z = dens$z) %>% add_surface()
```

```
HTML widgets cannot be represented in plain text (need html)
```

INSERT DISCUSSION COMMENTS

The 'kde2d()' function calculates a 2D kernel density estimate. Kernel Density Estimate is an application of smoothing the probability density function. It is a non-parametric method to estimate the pdf and random variable based on kernels and weights.

As a resulte, the graph is showing us the calculation of the KDE(z axis) for both X_1 as an x axis and $X_24 as an y axis.

i. 3D Scatter

First, compute the density of each (X1, X2) pair.

```
d$dens <- dmvnorm(x = d)
```

Now plot a point for each (X1, X2, dens) tuple.

```r
p4 <- plot_ly(d, x = ~ X1, y = ~ X2, z = ~ dens,
              marker = list(color = ~ dens,
                            showscale = TRUE)) %>%
  add_markers()

p4
```

```
HTML widgets cannot be represented in plain text (need html)
```

INSERT DISCUSSION COMMENTS

While 'rmvnorm()' function generates random samples from a multivariate normal distribution, 'dmvnorm()' function calculates the probability density funciton(pdf) of a multivariate normal distribution.

The plot function takes X_1 and x axis and X_2 as y axis and the z axis is the calculation of pdf of multivariate normal distribution.

Problem 4: Topic Modeling (No need to submit)

Try Topic Modeling on the `HPCorpus`. Where I have included Harry Potter Texts and the Lord of the Ring texts.

This article explains Topic Modeling in R very clearly (please follow it) https://www.tidytextmining.com/topicmodeling.html

Also,please follow this article on "stopping words" https://smltar.com/stopwords.html