Homework#2 - Die experiment and Music Shuffle

Riyesh Nath

2025-09-12

GROUP MEMBERS: Aqeel Choudhury, Heidy Collado and Ege Yanik

DIE EXPERIMENT (PART 1)

SUMMARY:

In this experiment, we will try to prove the fairness of a die that was heated in an oven for 20 minutes. We want to claim that this die is not fair.

• We first have null hypothesis to make claim that our dice is fair and we try to prove this claim wrong.

- **Null Hypothesis**: The die is fair. The probability of getting 6 in one roll is 1/6 (P(getting 6) = 1/6)

• We then have alternative hypothesis which if proven right will contradict our null hypothesis. This will disprove that the dice is fair.

- **Alternative Hypothesis**: P(getting 6) != 1/6

• We have random variable for the experiment.

- **Random variable**: The number of times we get 6 in our total (n) rolls.

• Here we know that probability of getting 6 if the dice is fair. It is 1/6. We also want to see true or false for getting an outcome. Given that we can use Bernoulli trial here.

TEST

Below is how our Bernoulli distribution would look like:

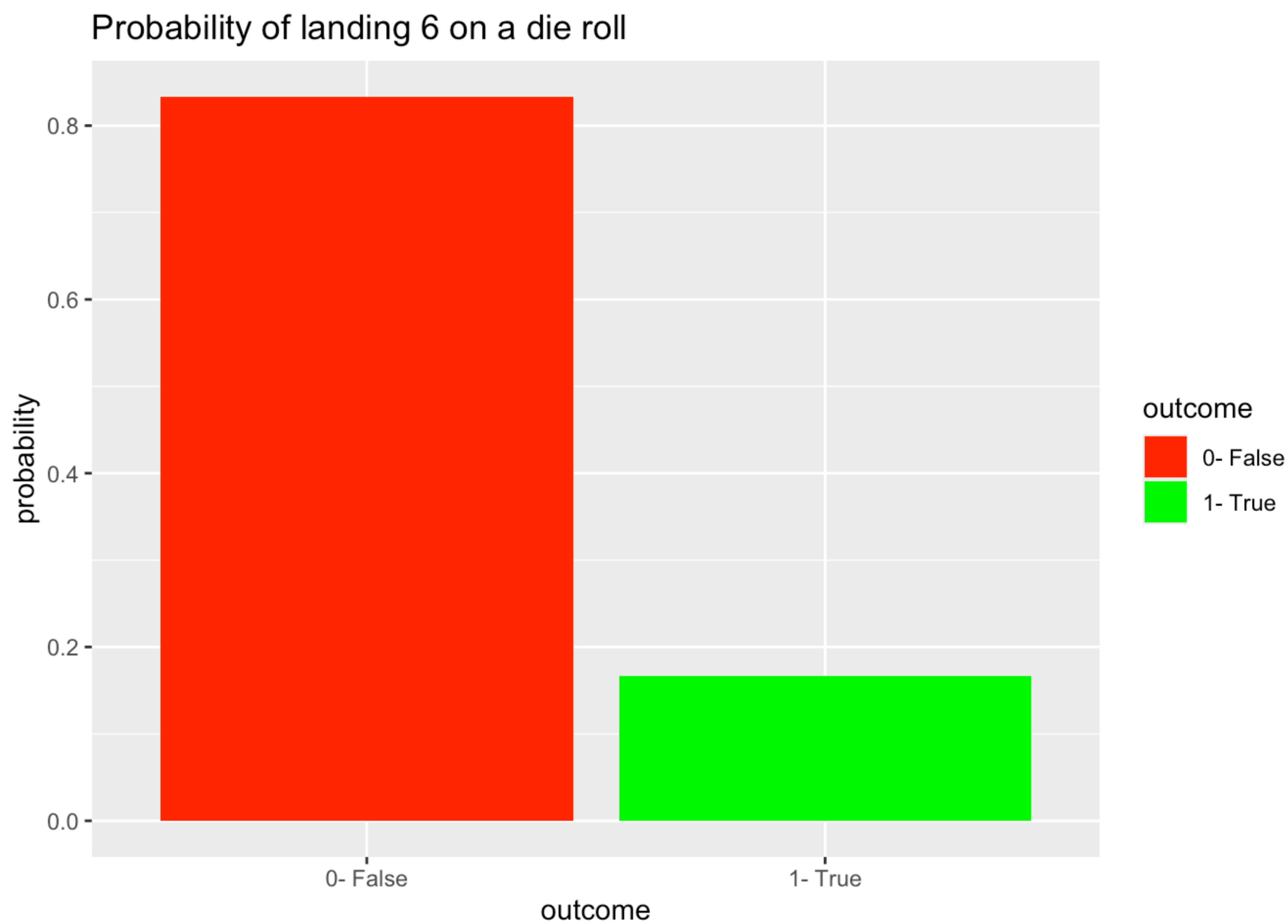
```r
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

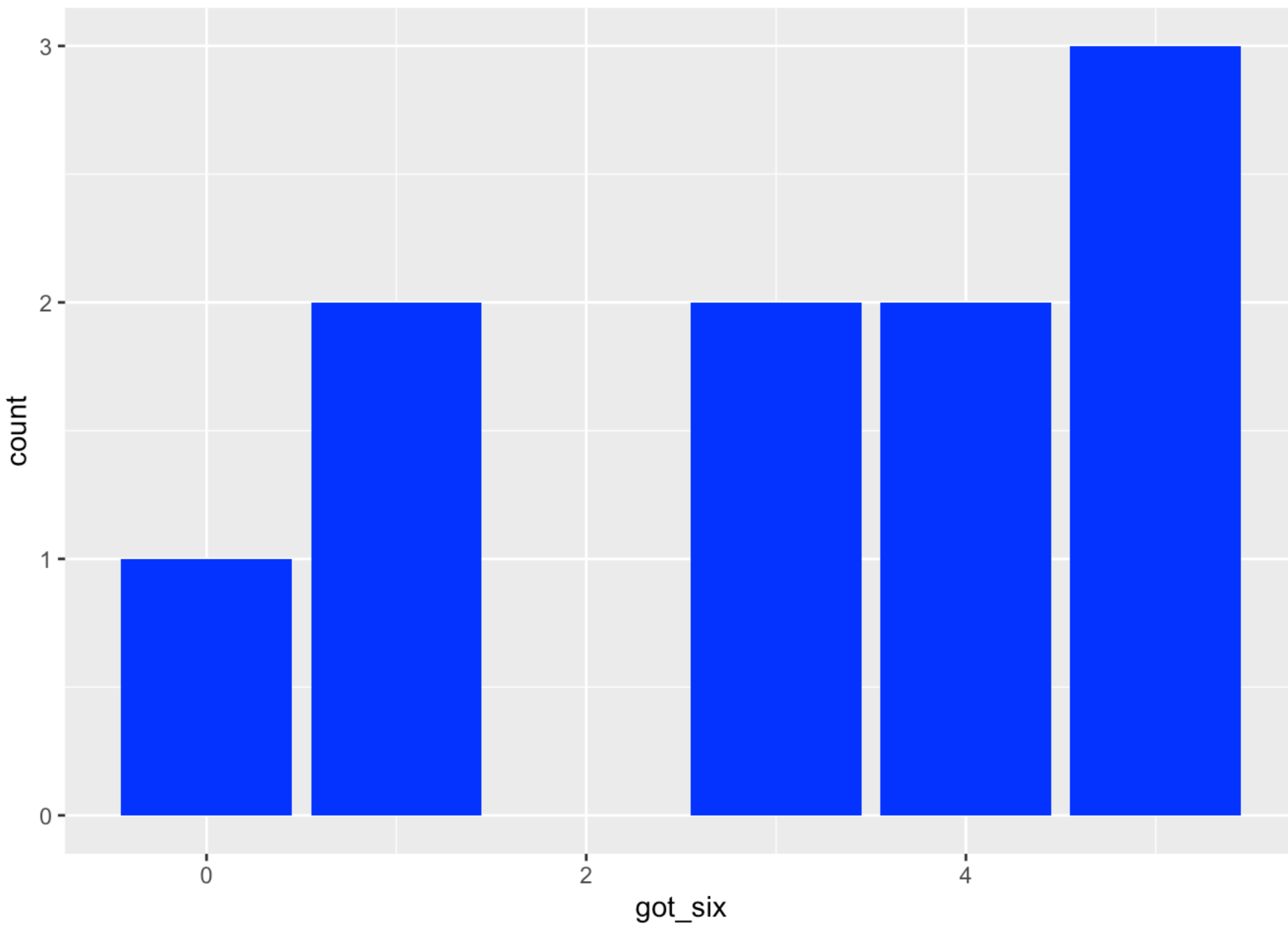
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
probability_of_rolling_6 <- 1/6
data <- data.frame(
  outcome=c("0- False","1- True"),
  probability=c(1-probability_of_rolling_6, probability_of_rolling_6)
)

ggplot(data, aes(x=outcome, y=probability, fill=outcome)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("red", "green") )+
  labs(title="Probability of landing 6 on a die roll")
```

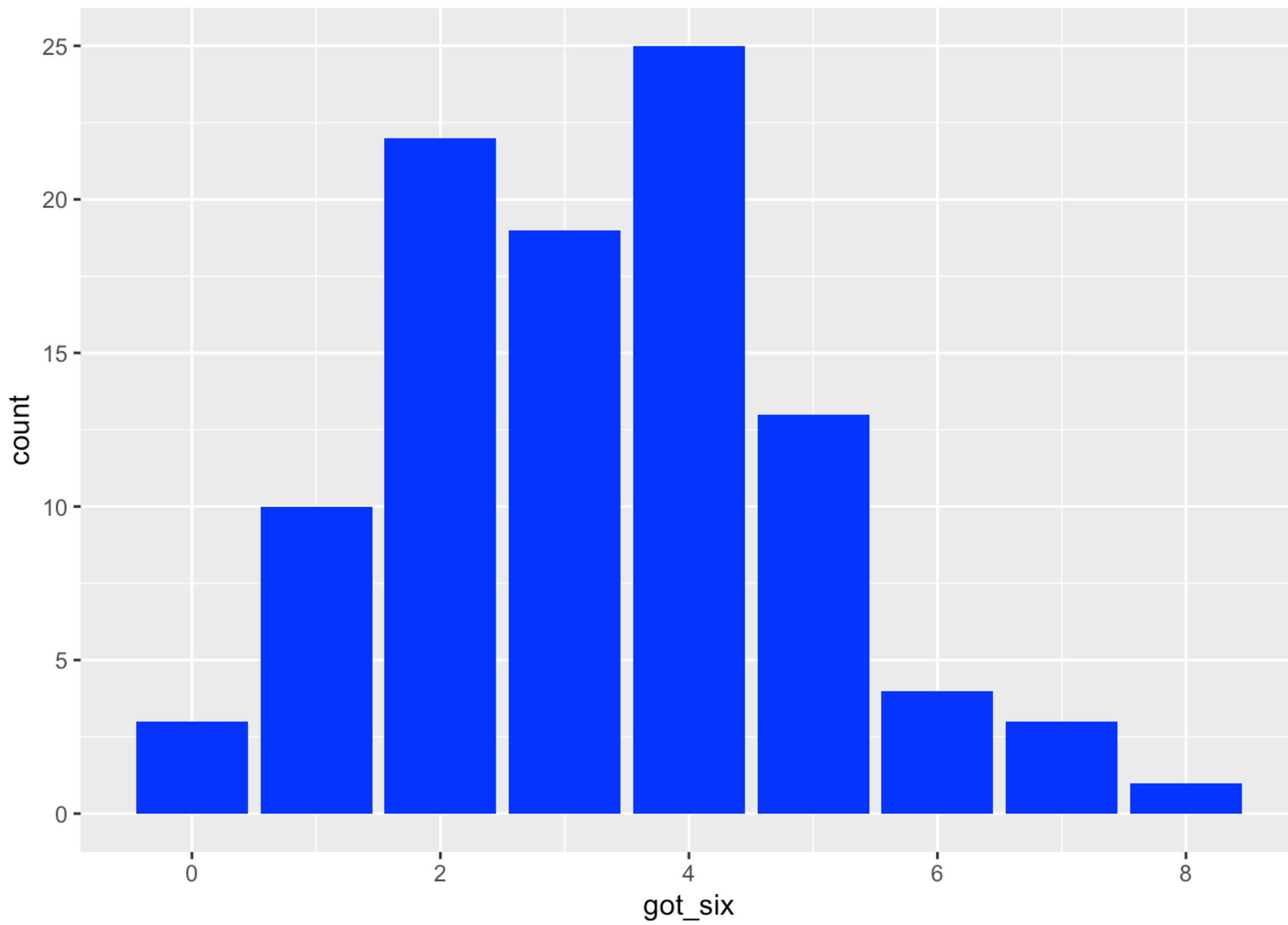


Running multiple experiments with multiple trials in each experiment, we can see how the distribution would look like. Here we will conduct the experiment using a small number of trials and then a large number of trials. This will show that having a larger trial size better represents all data including the tails. **THINK OF INCREASING THE NUMBER OF TRIALS ALLOWS YOU TO SEE COMPLETE RANGE AND INCREASING THE EXPERIMENT GIVES CLEARER PICTURE DUE TO CENTRAL LIMIT THEORM**.

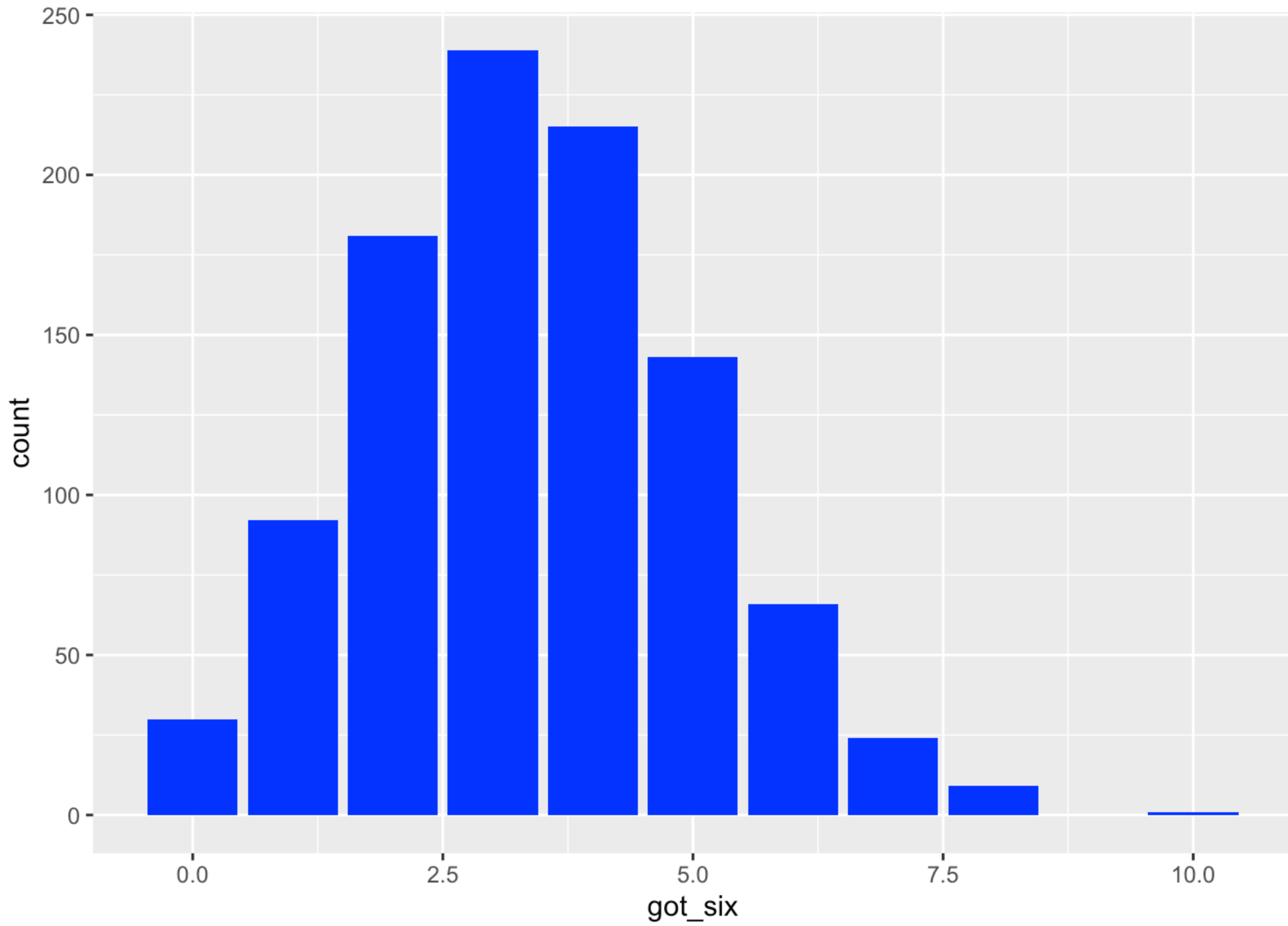
```
create_binomial_dist<- function(num_of_experiment, num_of_trials, probability) {
  results <- rbinom(num_of_experiment, num_of_trials, probability)
  data <- data.frame(got_six = results)

  ggplot(data, aes(x=got_six)) +
    geom_bar(fill = "blue")
}

create_binomial_dist(10, 20, probability_of_rolling_6)
```



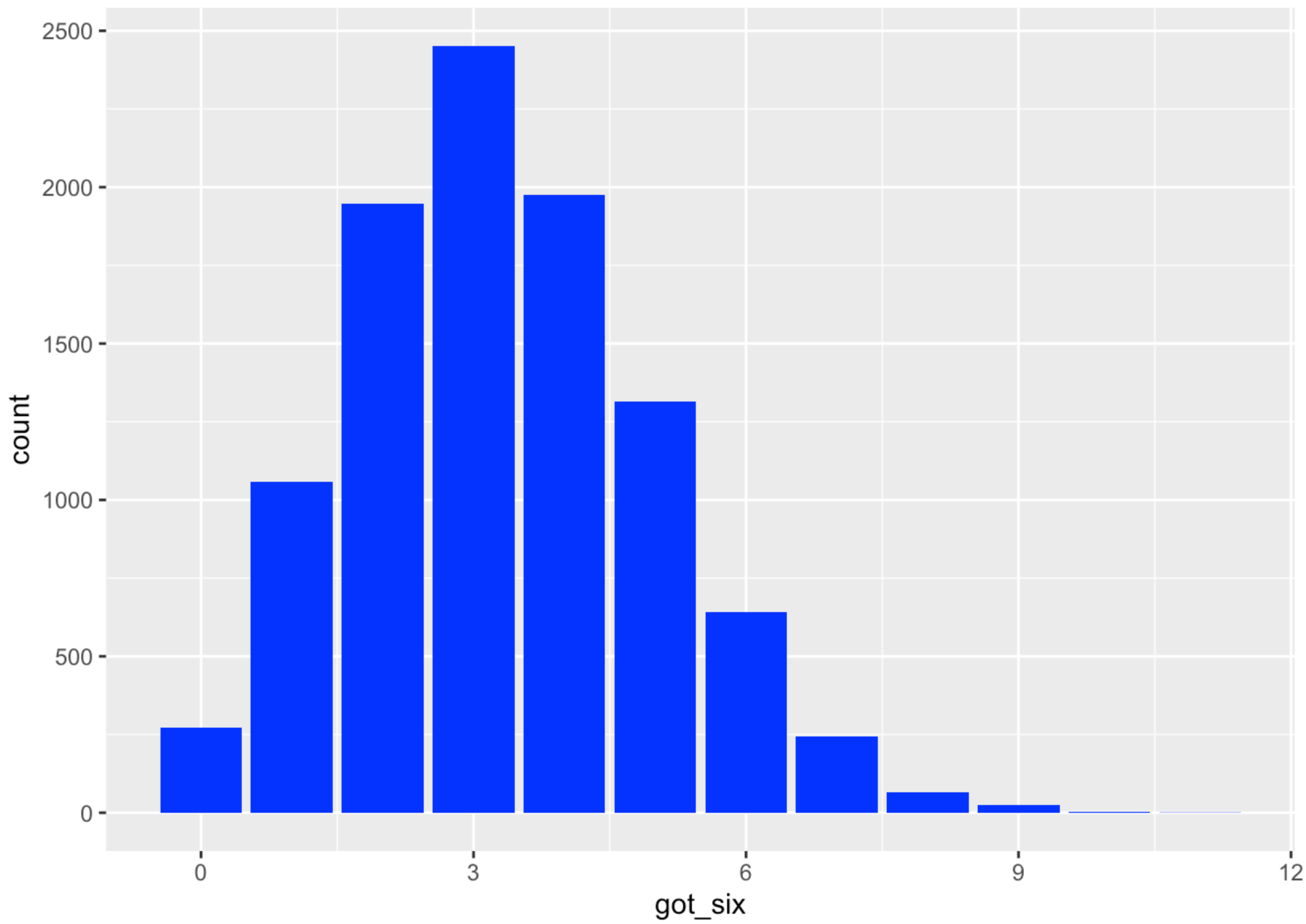

```
create_binomial_dist(100, 20, probability_of_rolling_6)
```



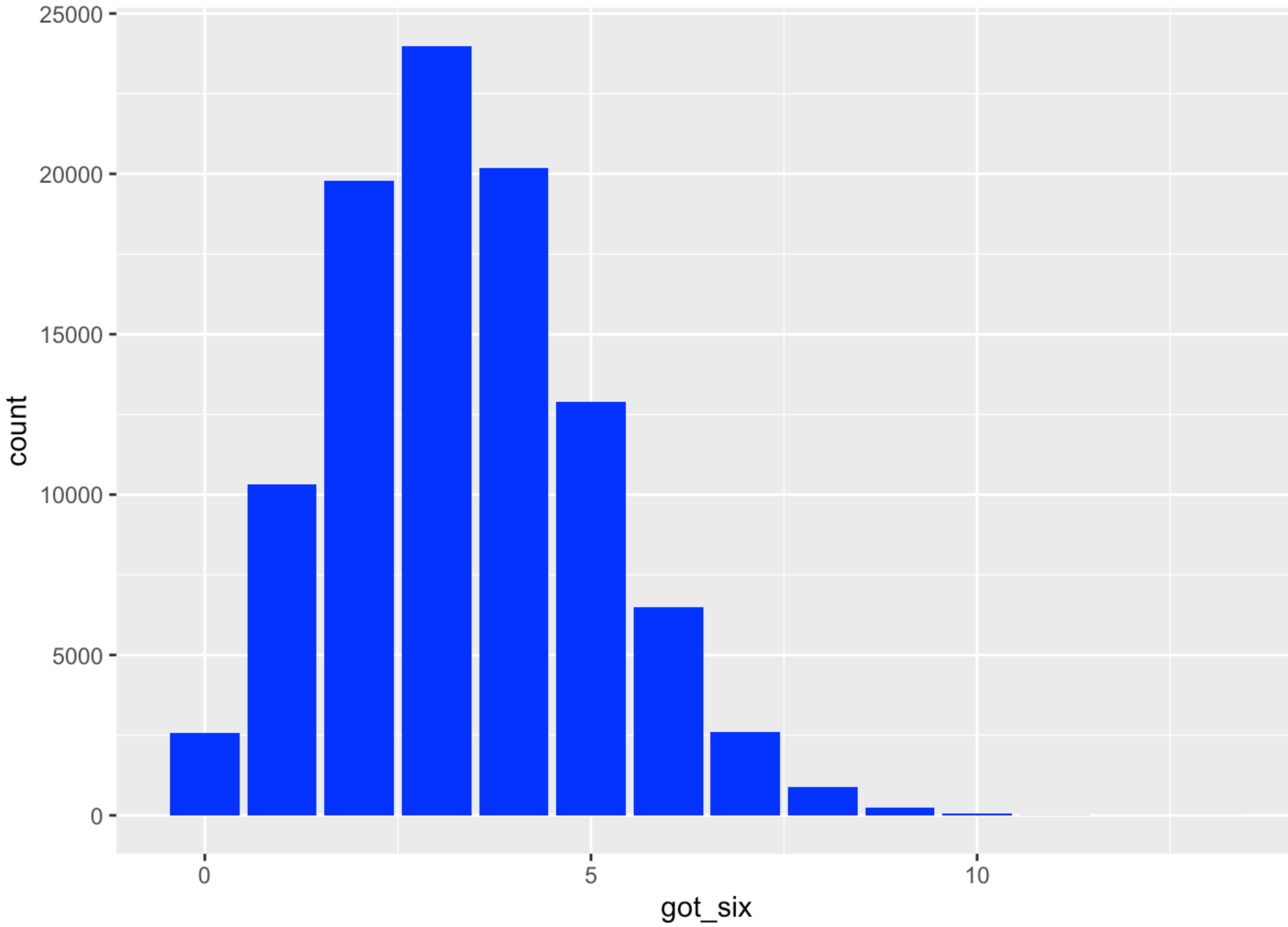

```
create_binomial_dist(1000, 20, probability_of_rolling_6)
```



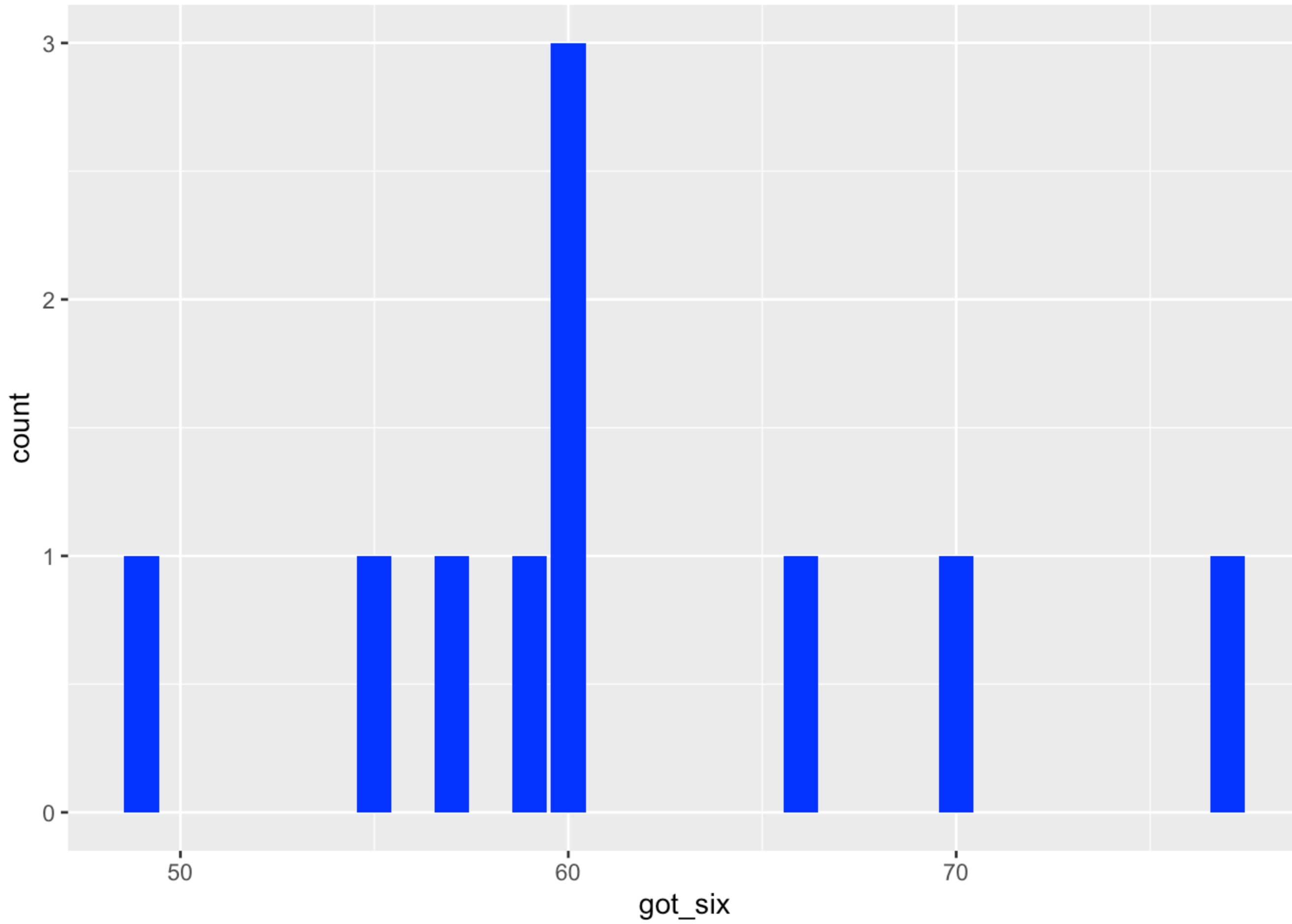

```
create_binomial_dist(10000, 20, probability_of_rolling_6)
```



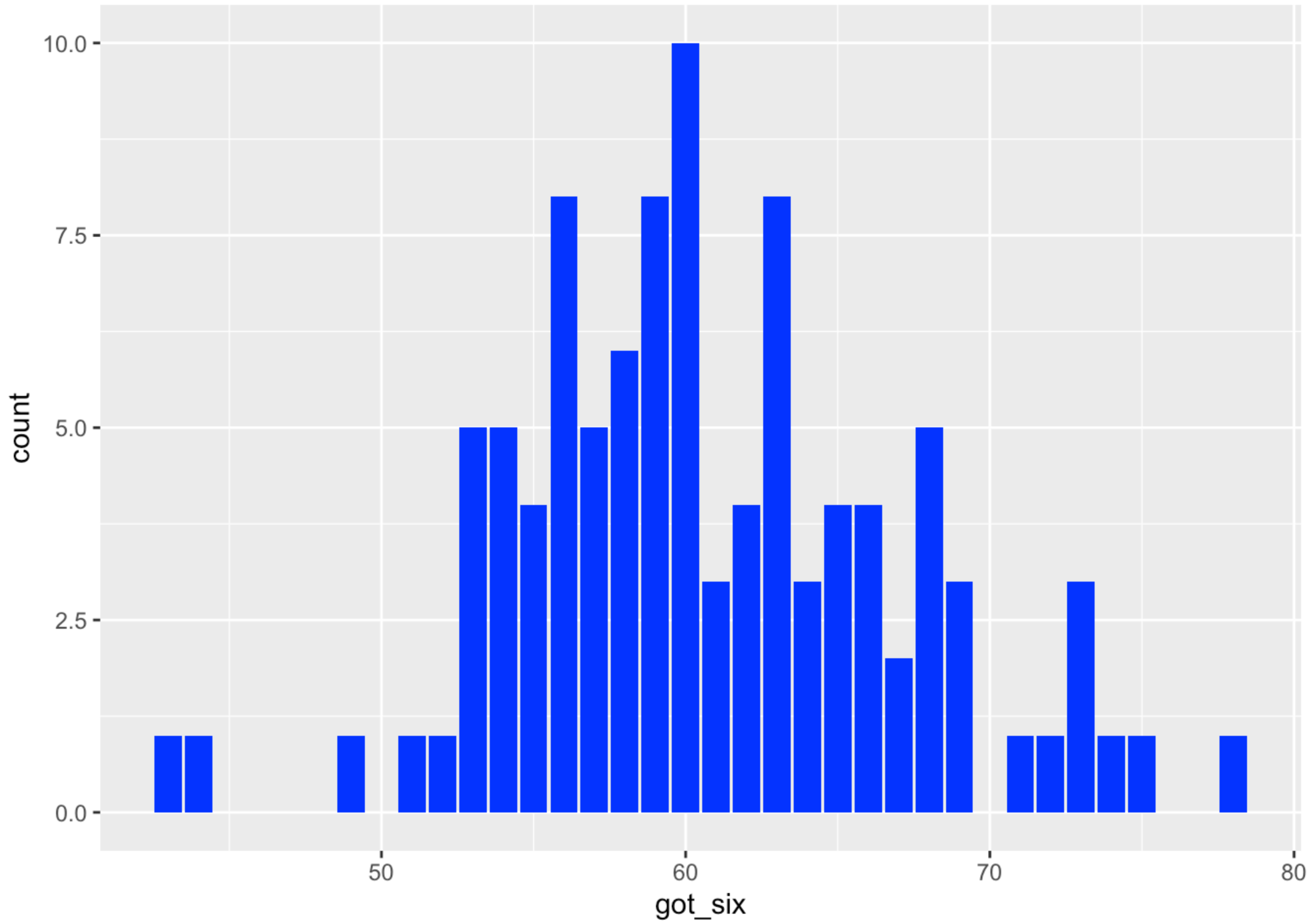

```
create_binomial_dist(100000, 20, probability_of_rolling_6)
```



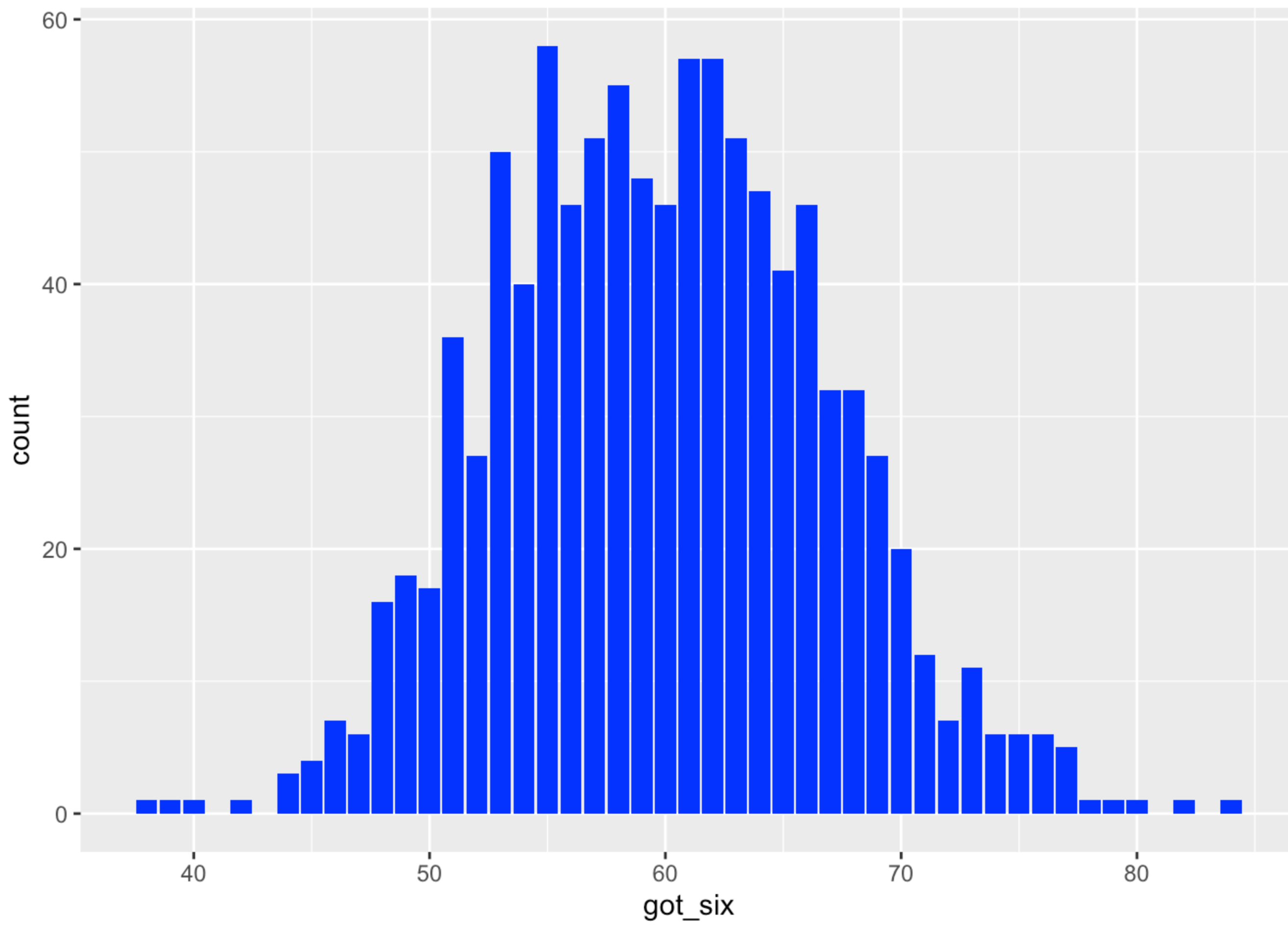

```
create_binomial_dist(10, 360, probability_of_rolling_6)
```



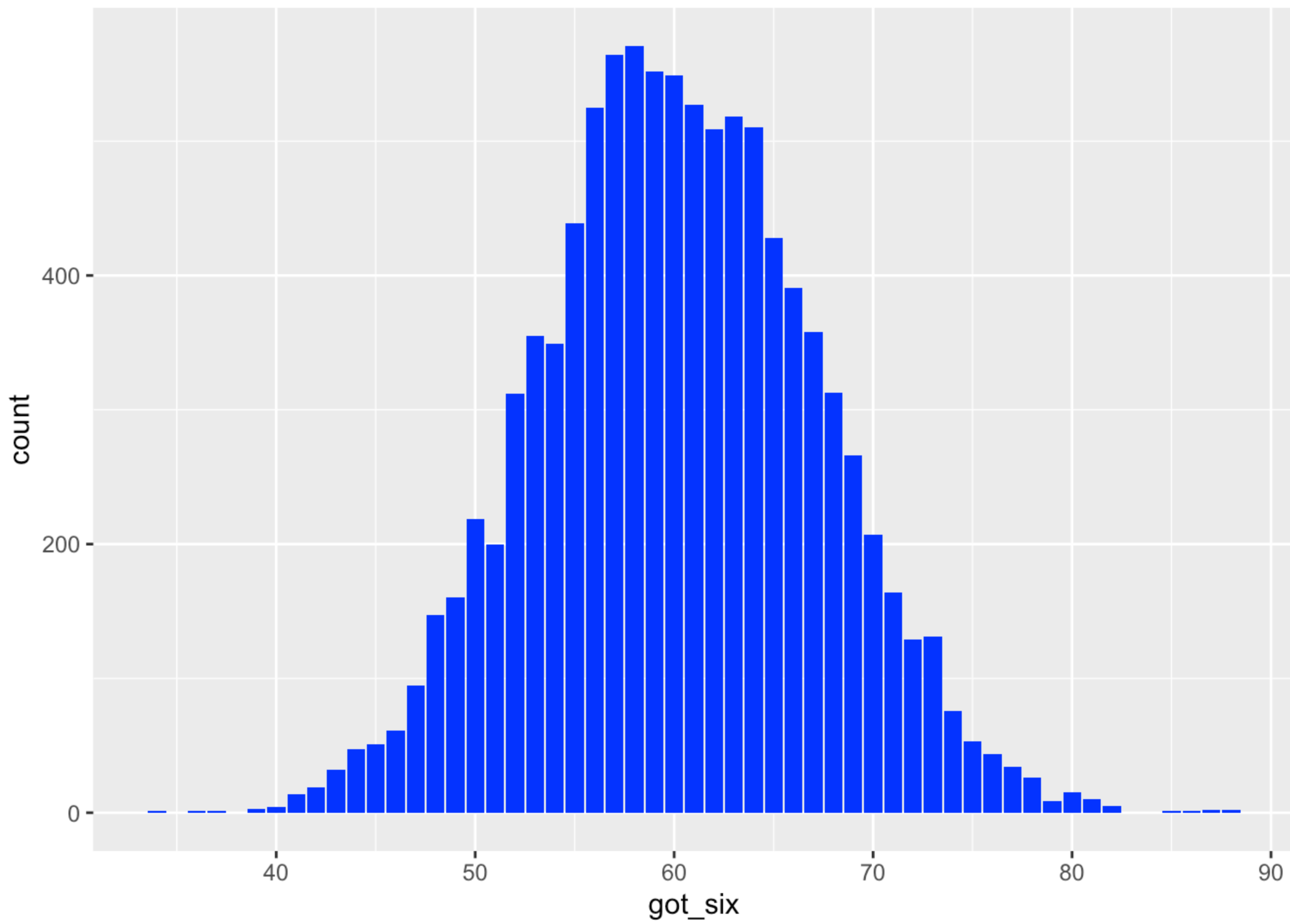

```
create_binomial_dist(100, 360, probability_of_rolling_6)
```



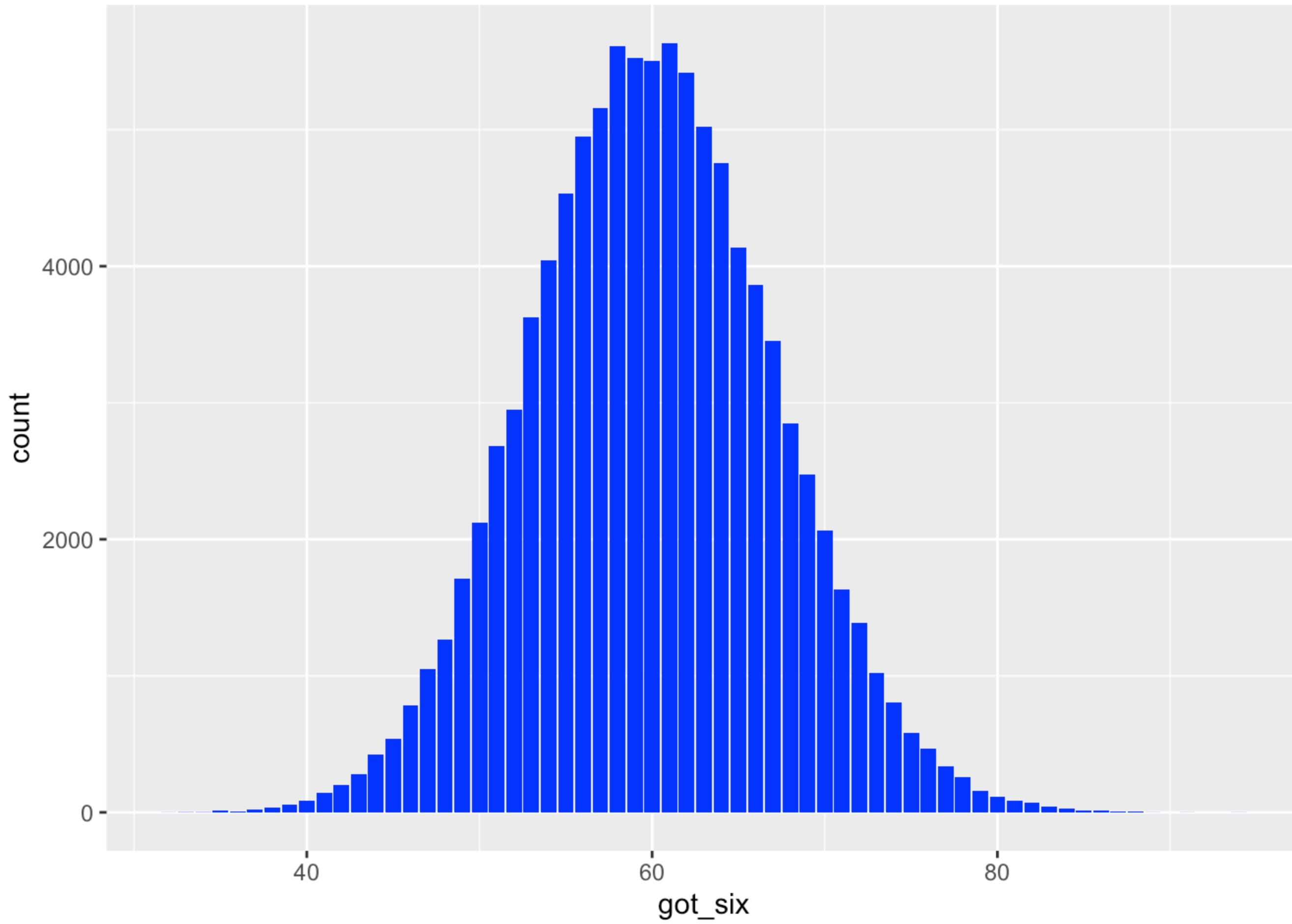

```
create_binomial_dist(1000, 360, probability_of_rolling_6)
```




```
create_binomial_dist(10000, 360, probability_of_rolling_6)
```




```
create_binomial_dist(100000, 360, probability_of_rolling_6)
```



ANALYSIS

In our experiment, we rolled the die 360 times and got 66 6s, 60 5s, 64 4s, 58 3s, 63 2s, 49 1s. Using binomial formula we get: 360C66 * (1/6)^66 * (5/6)^294

```
probability_result <- dbinom(66, size = 360, prob = 1/6)
print(probability_result)
```

```
## [1] 0.03820861
```

However this does not give us anything to show that 6 was rolled unfair amount of times. We can either use our Gaussian like distribution to fit our result and find p-value based on z score or just use binomial test. **I will use binomial test**

HYPOTHESIS TESTING

Here we can do binomial test to find the p-value with significance level of 5%. We will do two tailed test to find the fairness of the die.

```
binom.test(x = 66, n = 360, p = 1/6, alternative = "two.sided")
```

```
##
##  Exact binomial test
##
## data:  66 and 360
## number of successes = 66, number of trials = 360, p-value = 0.3961
## alternative hypothesis: true probability of success is not equal to 0.1666667
## 95 percent confidence interval:
##  0.1447213 0.2272455
## sample estimates:
## probability of success
##              0.1833333
```

We see that the p-value is no less than or equal to .05.

Furthermore we do see very little amount of 1s with a value of 49 rolls. However this also gives us p-value more than .05 so nothing odd here either.

```
binom.test(x = 49, n = 360, p = 1/6, alternative = "two.sided")
```

```
##
##  Exact binomial test
##
## data:  49 and 360
## number of successes = 49, number of trials = 360, p-value = 0.1371
## alternative hypothesis: true probability of success is not equal to 0.1666667
## 95 percent confidence interval:
##  0.1024205 0.1759250
## sample estimates:
## probability of success
##              0.1361111
```

CONCLUSION: CANNOT DISPROVE NULL HYPOTHESIS

MUSIC SHUFFLE (PART 2)

In this experiment, we will try to see if a music shuffle in Spotify will be a fair shuffle with 1/n probability for each outcome. Or will we see certain songs appear more than others.

• We first have a null hypothesis to make claim that each song has equal chance of appearing.

- **Null Hypothesis**: P(each song showing up) = 1/n

• We then have alternative hypothesis which if proven right will contradict our null hypothesis. This will disprove that the shuffle is fair.

- **Alternative Hypothesis**: P(each song showing up) != 1/n

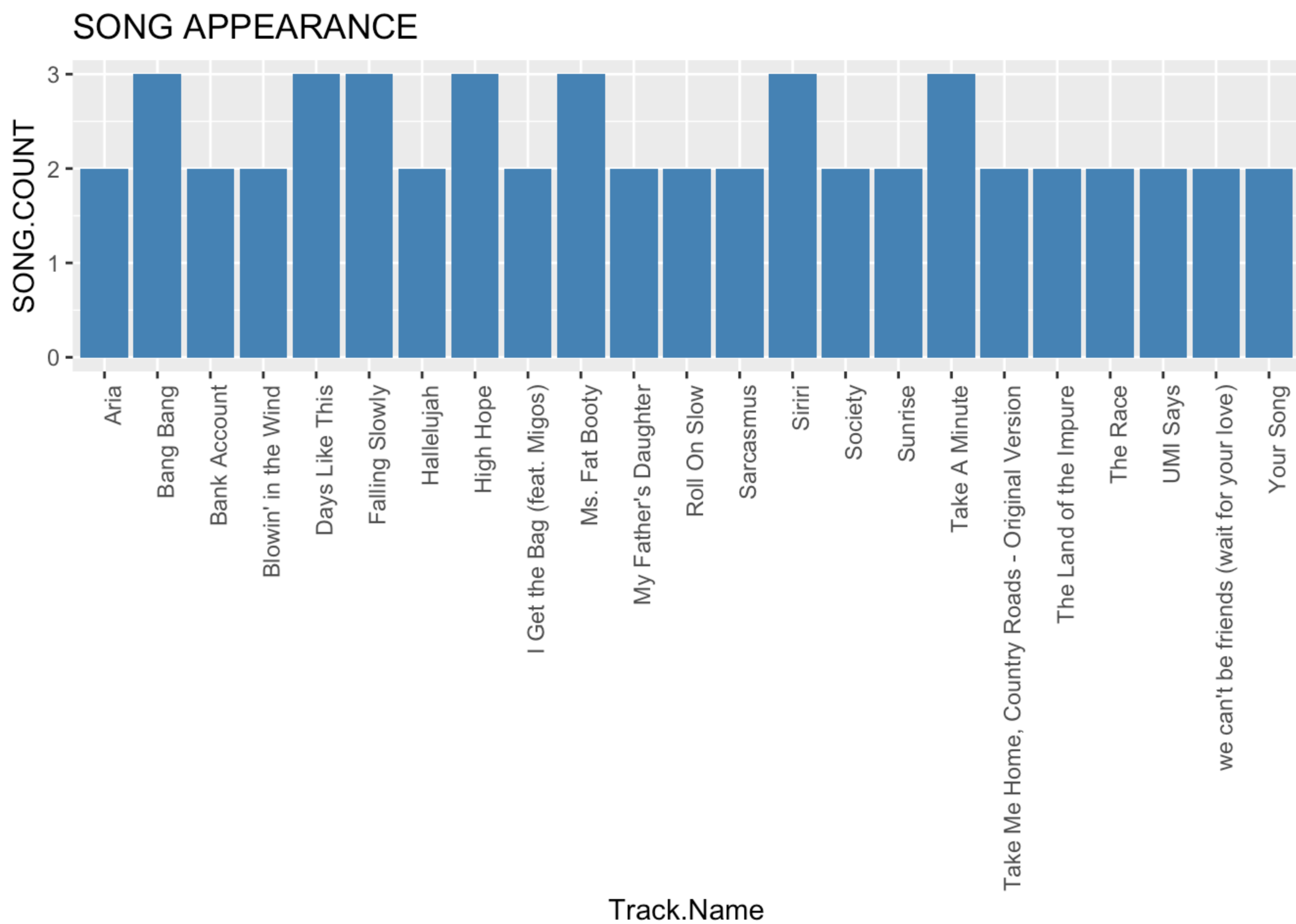
• We have random variable for the experiment.

- **Random variable**: number of times a given song was played

EXPLORE DATA

We use Exportify to get an excel sheet of a playlist of 23 songs. The trial size is limited due to the limitation of time. The playlist has some songs that I listen to on regular basis and some songs I do not listen to. There are several repeated artist to see if the shuffling algorithm latches on to similar genre or creator.

```
dataframe_of_playlist <- read.csv("Test.csv");
attach(dataframe_of_playlist)

ggplot(dataframe_of_playlist, aes(x = Track.Name, y = SONG.COUNT)) +
  geom_col(fill = "steelblue") +
  labs(title = "SONG APPEARANCE") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
print(SONG.COUNT)
```

```
##  [1] 2 3 2 3 2 2 2 2 2 2 3 2 2 2 2 2 2 2 3 3 2 3 3
```

OBSERVATION

The first run played all song once before moving to repeats. The playlist iterates, it iterates each one randomly but plays all songs before iterating again.

TEST

We will do a chi test but the data still isn't sufficient. However, given that all song iterates at least once, I would be a better idea to stop here.

We will use CHI Test to be fairness of shuffle

```
chisq.test(x = SONG.COUNT, p = rep( 1/length(SONG.COUNT), length(SONG.COUNT)))
```

```
## Warning in chisq.test(x = SONG.COUNT, p = rep(1/length(SONG.COUNT),
## length(SONG.COUNT))): Chi-squared approximation may be incorrect
```

```
##
##  Chi-squared test for given probabilities
##
## data:  SONG.COUNT
## X-squared = 2.1132, df = 22, p-value = 1
```

CONCLUSION

We are not able to disprove the null hypothesis. However, I am not certain if the shuffle algorithm randomizes or creates weight as a song is not play so every song gets played atleast once. NOTE: This experiment in incomplete. We would need to look at how shuffle works if used on playlist that is used more often. The playlist in this experiment was freshly created.