STATS 2107
Statistical Modelling and Inference II
Solutions
Workshop 5: Sampling distributions part 2

Matt Ryan

Semester 2 2022

Contents

The sampling distribution of the P-value

The Normal hypothesis test

Consider the hypothesis test on $X_1, X_2 \ldots, X_n$ where $X_i \sim N(\mu, \sigma^2)$ and σ^2 is known. The simple null hypothesis is

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_a : \mu \neq \mu_0$$

with test statistic

$$Z^* = \frac{\bar{X} - \mu_0}{\sigma/\sqrt{n}} \sim N(0,1)$$

Let's think about the p-value

By definition, the P-value is

$$P = P(|Z| > z^*)\,,$$

where z^* is the observed value of the test statistic.

What if I told you this a random variable?

The p-value as a random variable

If the X_i, $i = 1, 2, \ldots, n$ are not yet observed, then

$$Z^* = \frac{\bar{X} - \mu_0}{\sigma/\sqrt{n}} \sim N(0,1)$$

is random. Hence

$$P = P(|Z| > Z^*)$$

is random

A thrilling question

If P is random, what is its distribution?

How can we simulate a p-value

To explore the distribution of the p-value, we need 3 things:

1. A null distribution (known values of μ and σ^2).
2. Some data from the null distribution.
3. To calculate the P-value.

A null distribution

Let's suppose that $X_1, X_2, \ldots, X_n$ are i.i.d. $N(0,1)$ for simplicity. Then the null hypothesis we are testing is

$$H_0 : \mu = 0\,.$$

How do we get data?

The easiest way to get data is to simulate it using R. Let's simulate a sample of $n = 100$ observation, which we can do with

```
rnorm(n = 100, mean = 0, sd = 1)
```

Get the P-value

To do this, we need to calculate the test statistic z^*, and calculate

$$P = P(|Z| > z^*) = 2P(Z < -|z^*|)\,.$$

R code for the p-value

```
x <- rnorm(n = 100, mean = 0, sd = 1)
z <- mean(x)/(1/sqrt(100))
p <- 2*pnorm(-abs(z))
```

How does this help?

These are the steps to simulate a single p-value. If we do this LOTS and LOTS of times, we can then plot the simulated distribution to see how it looks (with a histogram).

This is where you come in

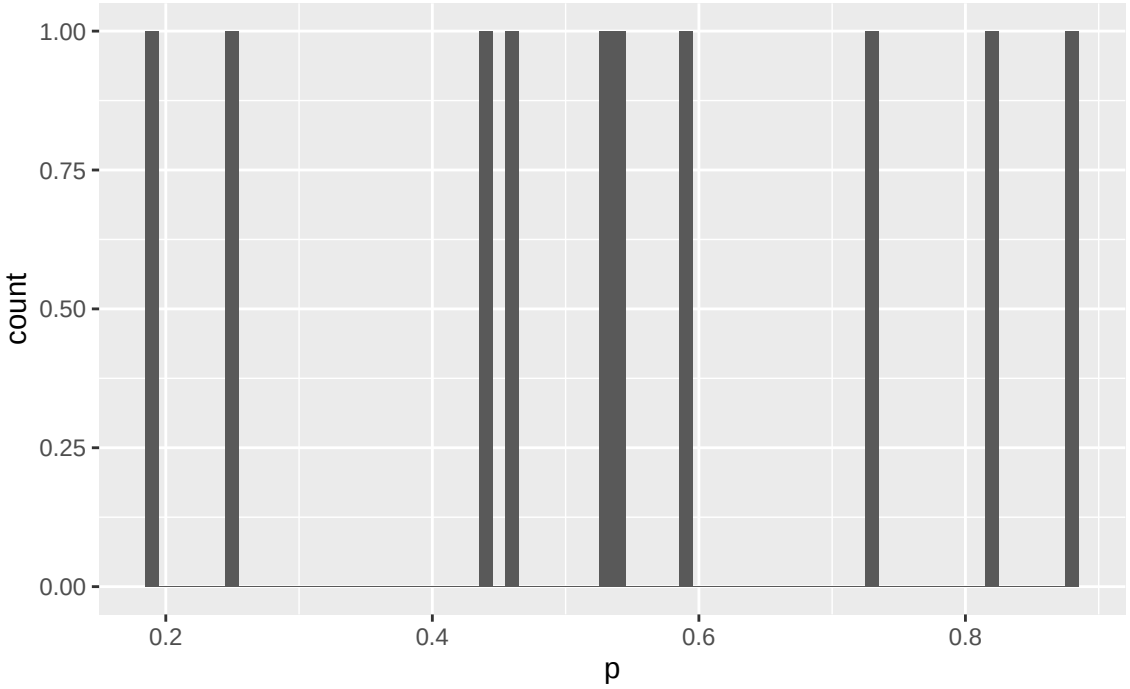
Your turn

What to do

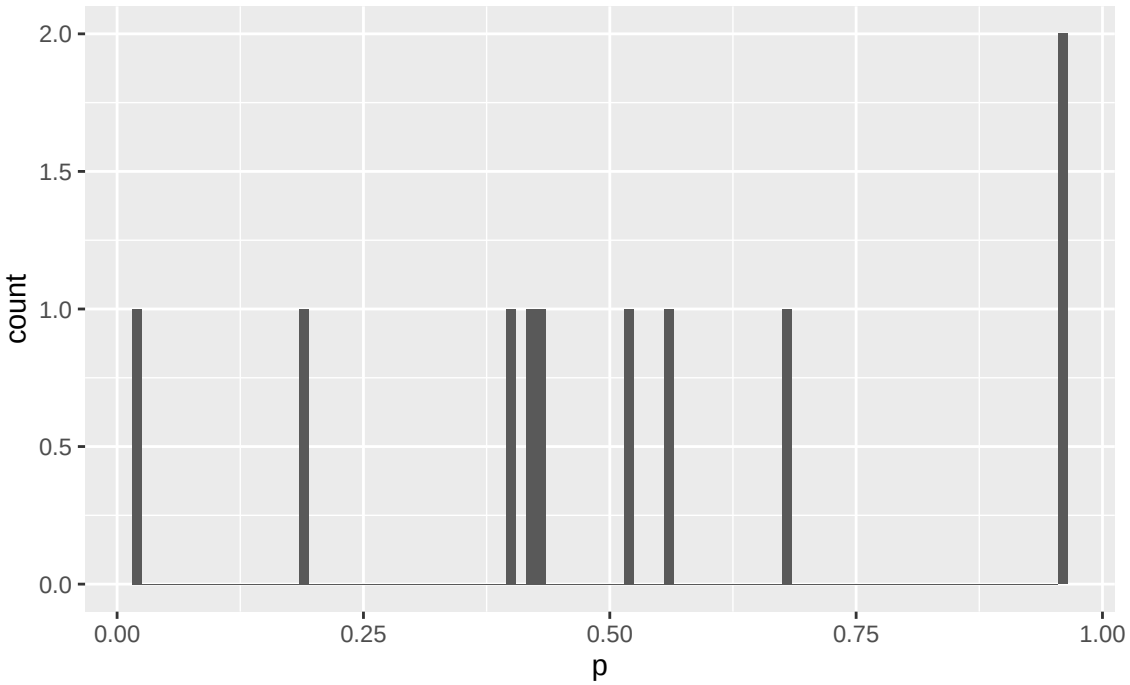
1. Write R code to simulate $N = 10$ p-values. Generate a histogram of these p-values. **Hint: Use a for loop**

Solutions:
Below are three ways to do this, going from a simple for loop to a fancy function.

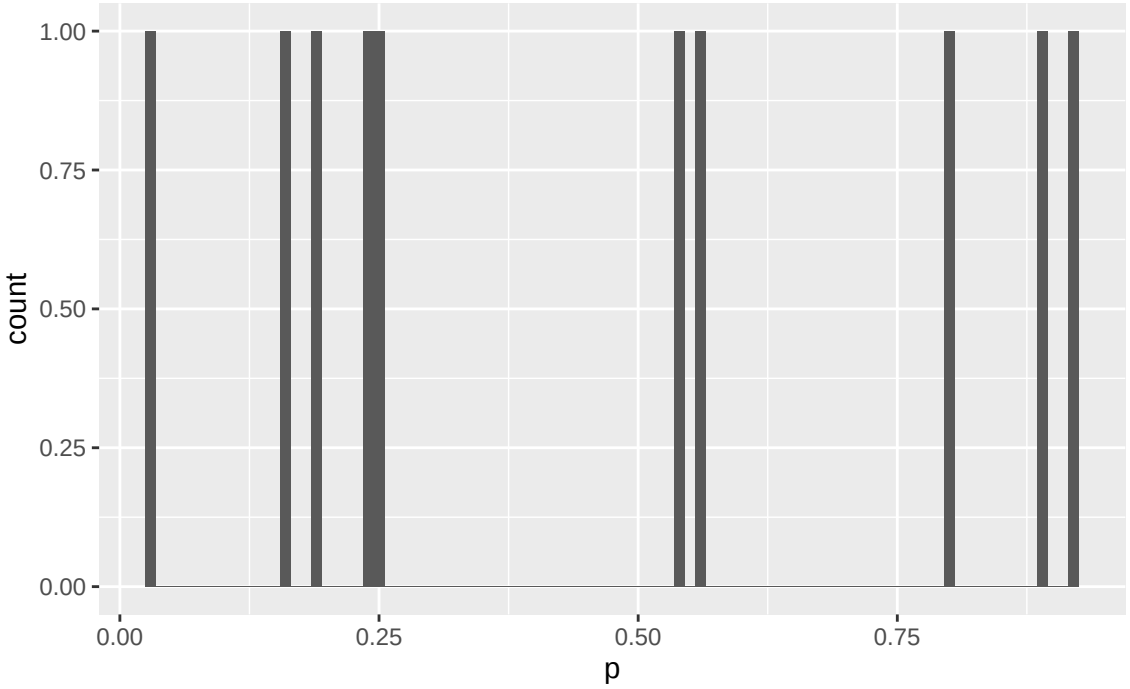
```
# A simple for loop
# Set you parameters
mu <- 0
s <- 1
n <- 100
N <- 10
# initialise P
p <- numeric(N)
# Calculate P
for(i in 1:N){
  x <- rnorm(n = n, mean = mu, sd = s)
  z <- (mean(x) - mu)/(s/sqrt(n))
  p[i] <- 2*pnorm(-abs(z))
}

# Plot
ggplot(tibble(p = p), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```




```
# Vectoring the for loop
# This uses the map function from purrr
mu <- 0
s <- 1
n <- 100
N <- 10

# Calculate P
# This code reads:
## Generate rnorm(n = n, mean = mu, sd = s) N times
## Then, for each of these simulations, calculate the p-value, and return the value
## The map_dbl function is the vectorisation part.
p <- N %>%
  rerun(rnorm(n = n, mean = mu, sd = s)) %>%
  map_dbl(function(x){
    z <- (mean(x) - mu)/(s/sqrt(n))
    p <- 2*pnorm(-abs(z))
    return(p)
  })

# Plot
ggplot(tibble(p = p), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```



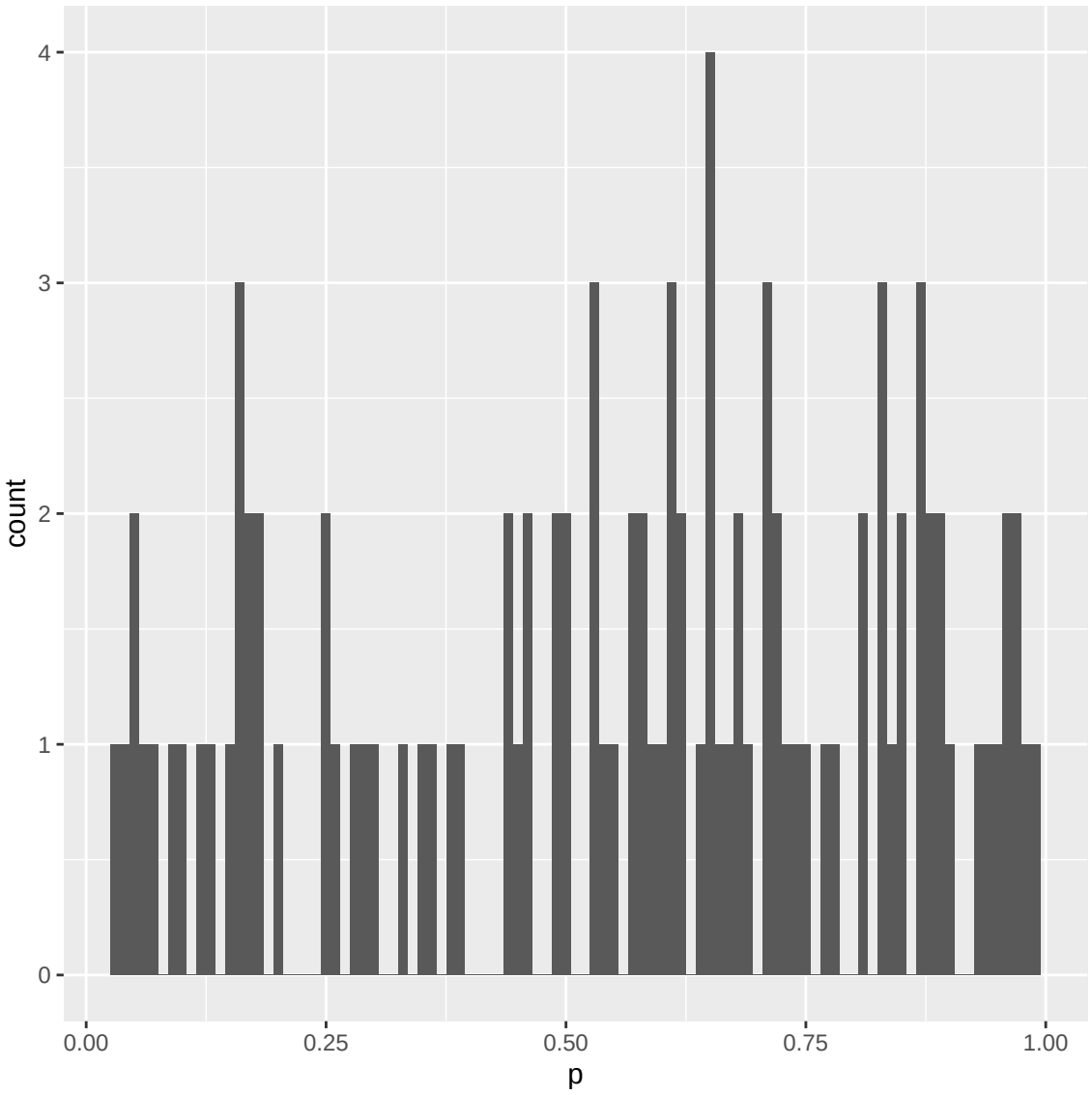

```r
# Define a function for easy repeatability
# I have taken the vectorised code, and turned it into a function
# I have set default values for mu, s, and n so all we need to give it is the number of simulations
# defined be N
generate_p_value <- function(N, mu = 0, s = 1, n = 100){
  p <- N %>%
    rerun(rnorm(n = n, mean = mu, sd = s)) %>%
    map_dbl(function(x){
      z <- (mean(x) - mu)/(s/sqrt(n))
      p <- 2*pnorm(-abs(z))
      return(p)
    })
  return(p)
}

p <- generate_p_value(10)

# Plot
ggplot(tibble(p = p), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```



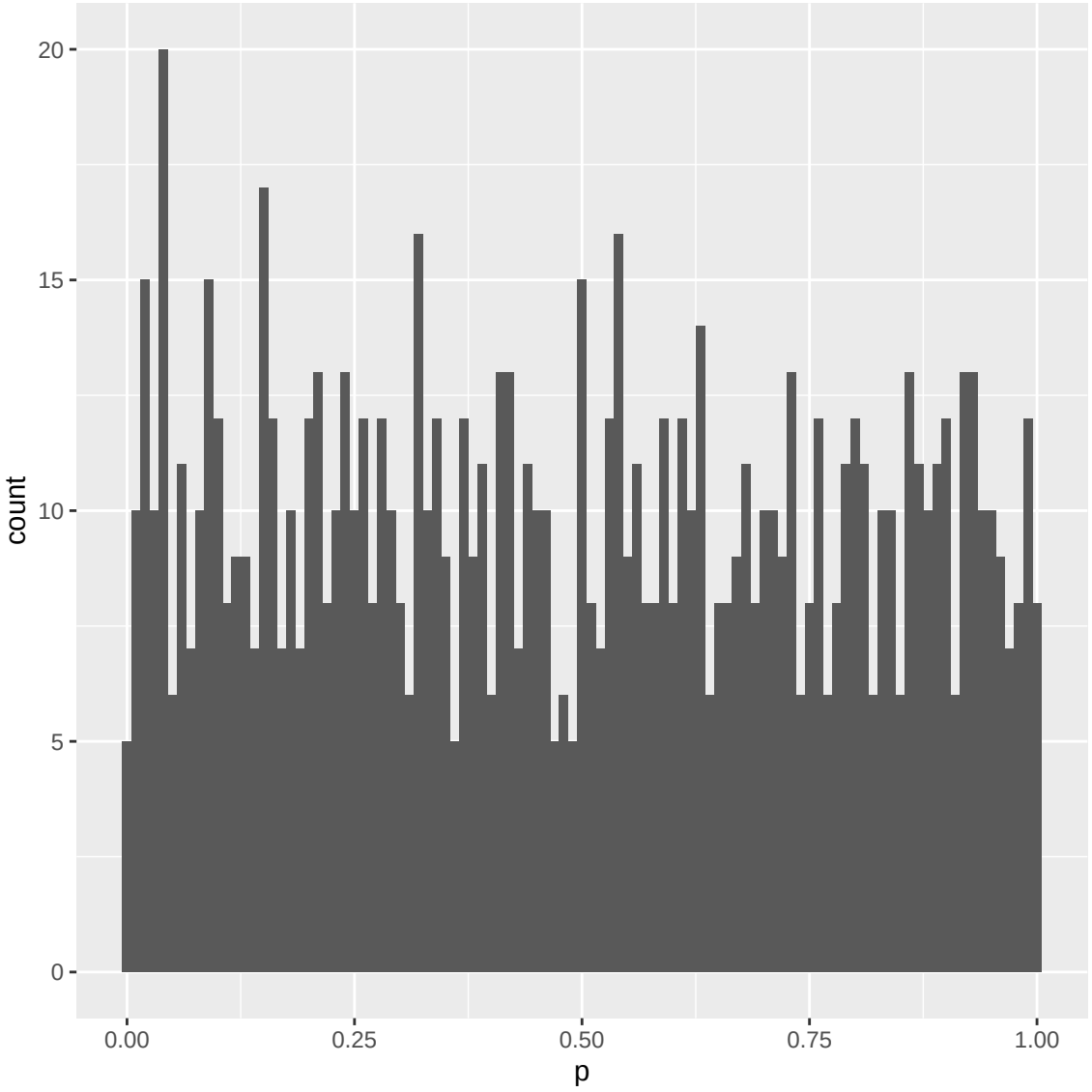
2. Adapt your code to simulate $N = 100, 1000, 10000$ p-values. Generate histograms for each value of N.

Solutions:
We present two solutions, one using the for loop, one using the function and getting fancy.

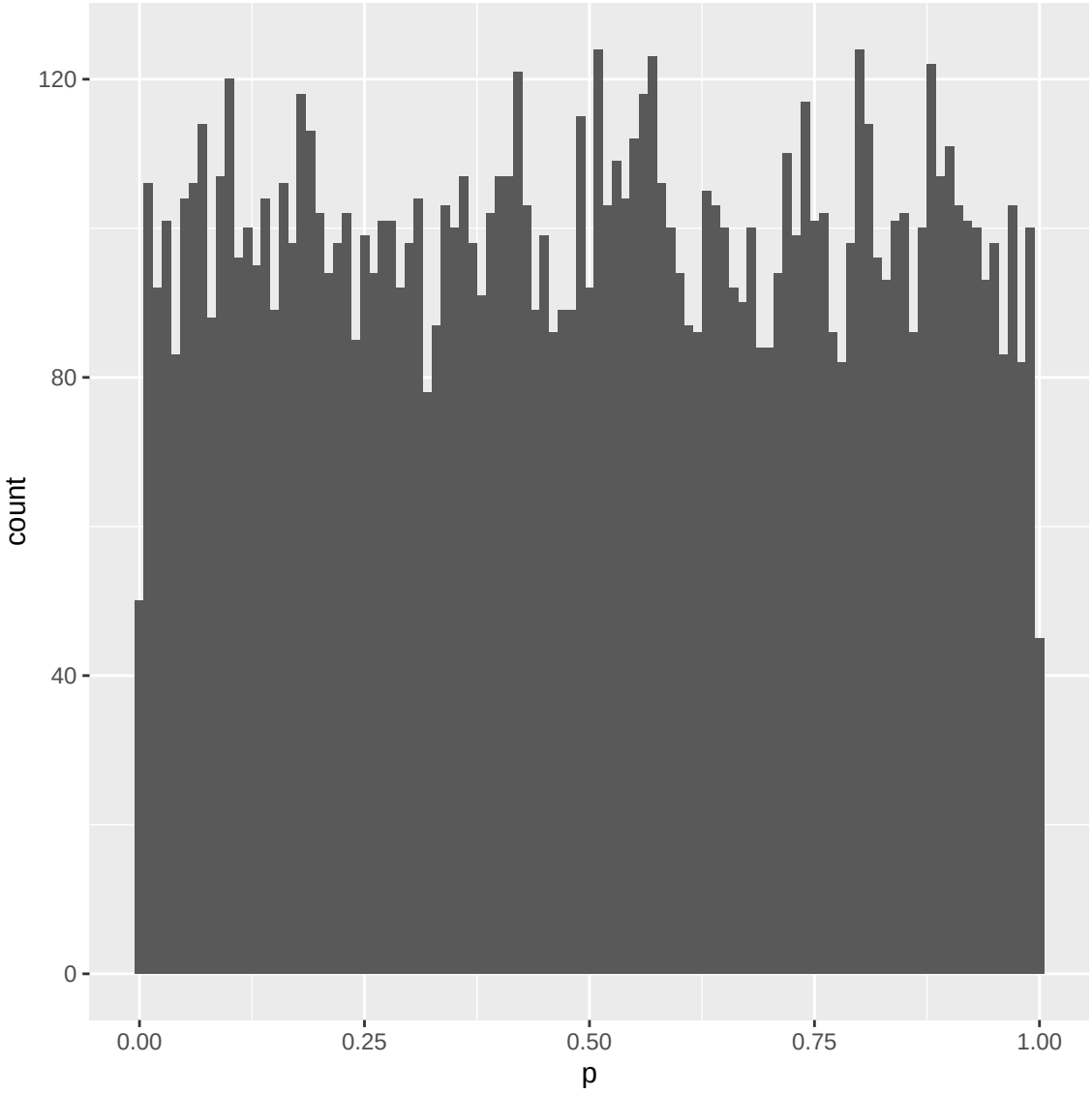
```
# A simple for loop
# Set you parameters
mu <- 0
s <- 1
n <- 100

## N = 100
N <- 100
# initialise P
p <- numeric(N)
# Calculate P
for(i in 1:N){
  x <- rnorm(n = n, mean = mu, sd = s)
  z <- (mean(x) - mu)/(s/sqrt(n))
  p[i] <- 2*pnorm(-abs(z))
}

# Plot
ggplot(tibble(p = p), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```



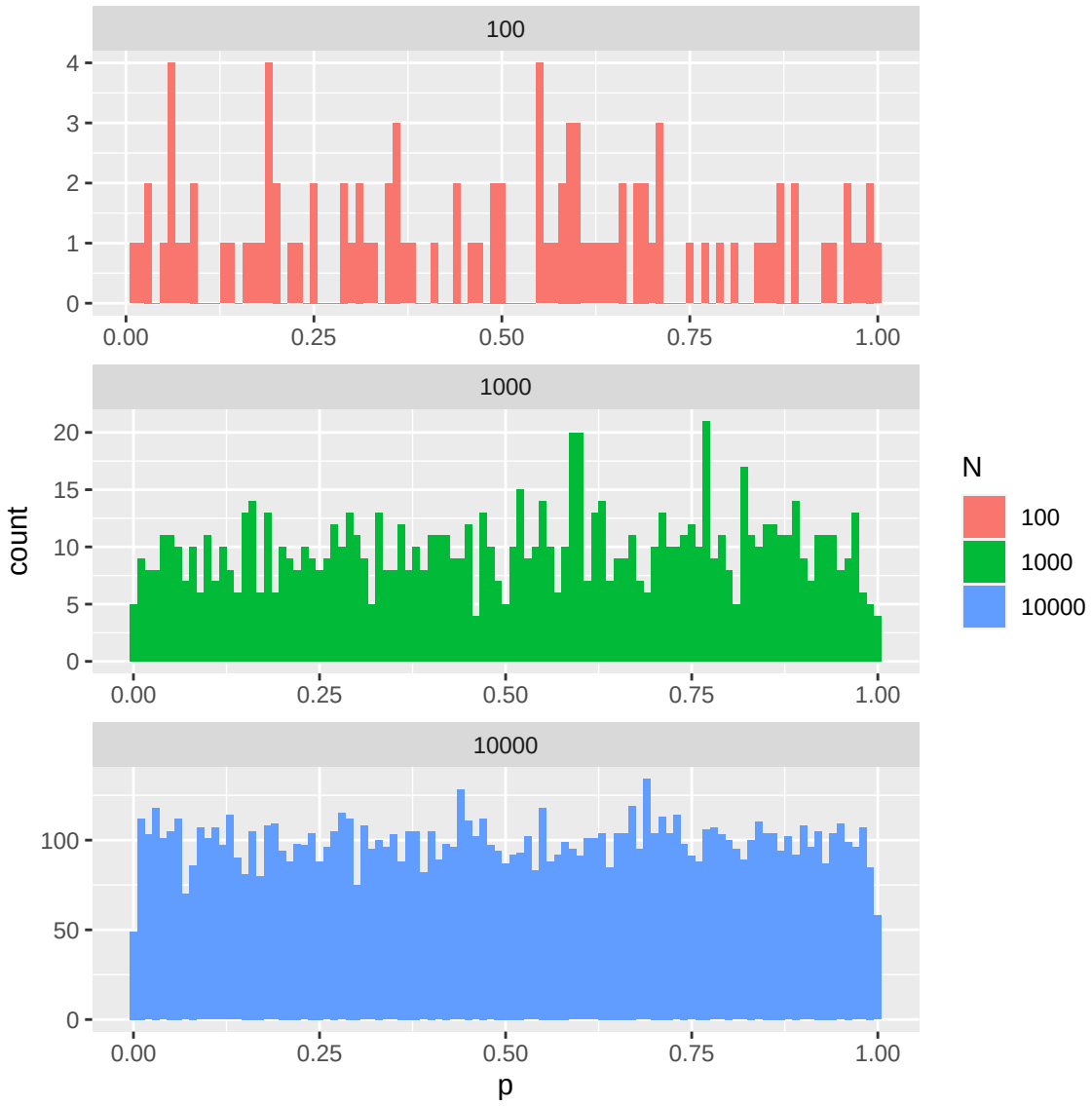

```
## N = 1000
N <- 1000
# initialise P
p <- numeric(N)
# Calculate P
for(i in 1:N){
  x <- rnorm(n = n, mean = mu, sd = s)
  z <- (mean(x) - mu)/(s/sqrt(n))
  p[i] <- 2*pnorm(-abs(z))
}

# Plot
ggplot(tibble(p = p), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```




```
## N = 10000
N <- 10000
# initialise P
p <- numeric(N)
# Calculate P
for(i in 1:N){
  x <- rnorm(n = n, mean = mu, sd = s)
  z <- (mean(x) - mu)/(s/sqrt(n))
  p[i] <- 2*pnorm(-abs(z))
}

# Plot
ggplot(tibble(p = p), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```



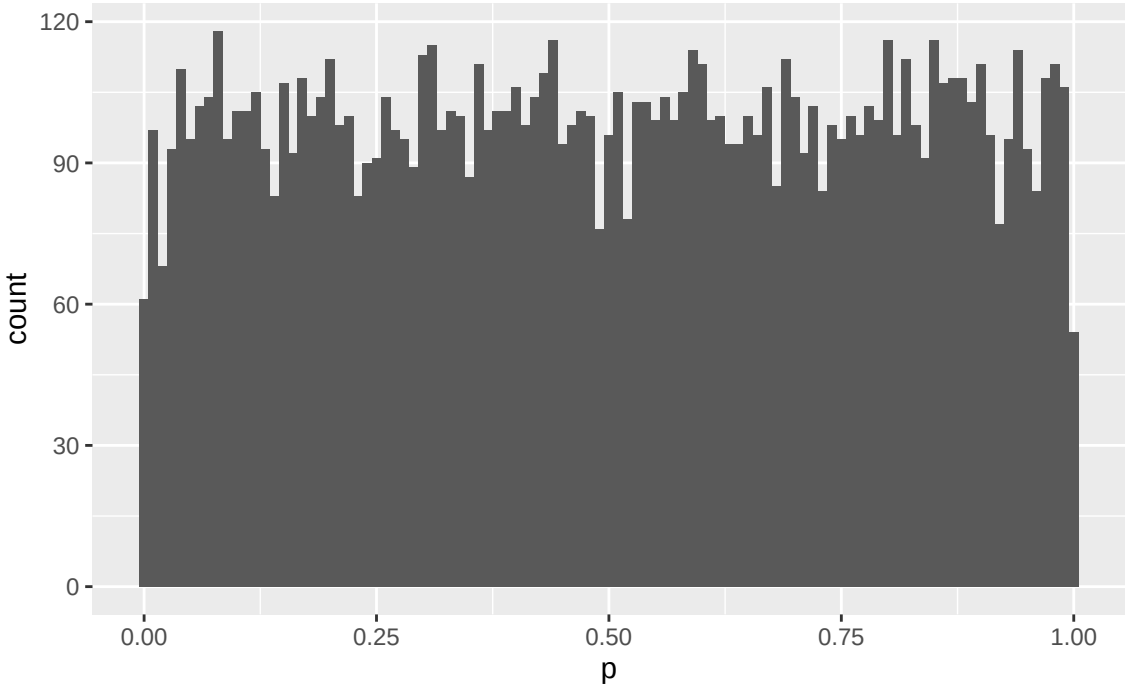

```
# Getting fancy
df <- tibble(
  N = c(100, 1000, 10000)
) %>%
  mutate(p = map(N, generate_p_value),
         N = as.factor(N)) %>%
  unnest(p)

df %>%
  ggplot(aes(x = p, fill = N)) +
  geom_histogram(binwidth = 0.01) +
  facet_wrap(~ N, nrow = 3, scale = "free")
```



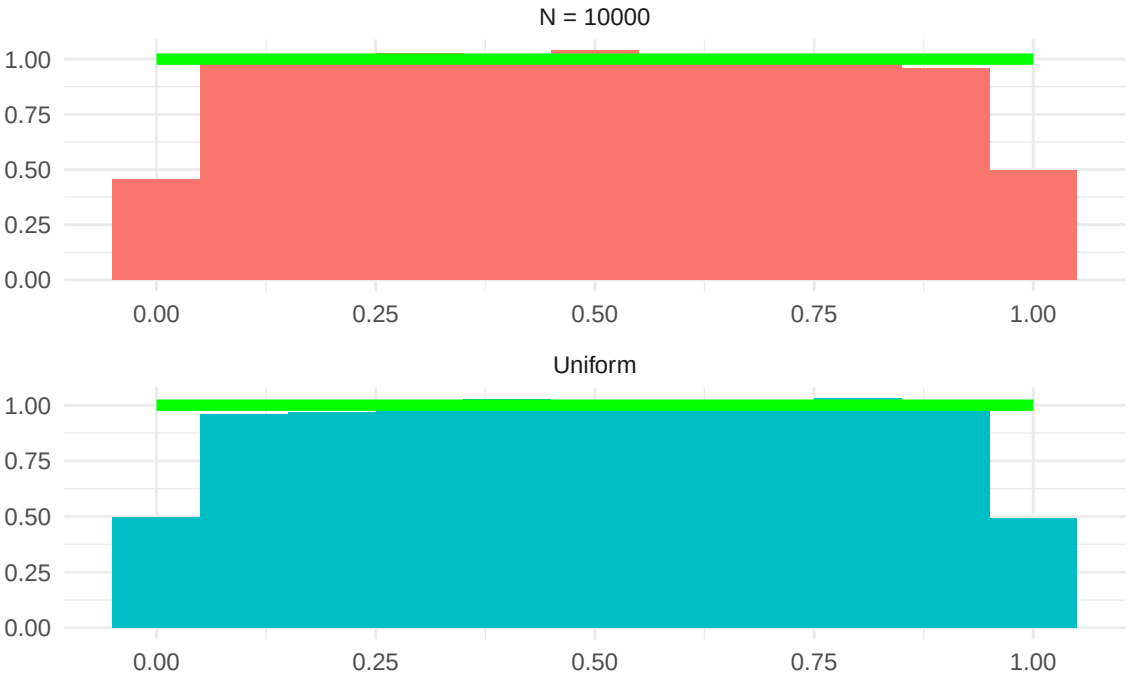
3. Propose a sampling distribution for P.

Solutions:
This is starting to look somewhat uniform on $(0, 1)$. For comparison:

```
ggplot(tibble(p = runif(10000)), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```



What did I get:



The sampling distribution of P is uniform

A powerful theorem

Let X be a continuous random variable with invertible CDF $F(x)$. Then the random variable $Y = F(X)$ is a $U(0, 1)$ random variable.

Why is this useful

1. The CDF of a continuous random variable is strictly monotonic, hence invertible. Thus this applies to many random variables.
2. This allows us to simulate random variables.

A proof.

Observe that

$$\begin{aligned} P(Y \leq y) &= P(F(X) \leq y) \\ &= P(X \leq F^{-1}(y)) \\ &= F(F^{-1}(y)) \\ &= y\,. \end{aligned}$$

A proof

1. CDFs uniquely identify distributions
2. The CDF of $U \sim U(0,1)$ is $F_U(u) = u$.

How does this help us.

Consider the definition of the P-value:

$$P = P(|Z| > Z^*) = 1 - P(|Z| < Z^*)\,.$$

Then $|Z|$ is a random variable, so

$$1 - P = F_{|Z|}(Z^*)\,.$$

How does this help us.

Thus $1 - P \sim U(0,1)$, so $P \sim U(0,1)$.

Your turn

What to do

1. Under the null hypothesis, what is the probability that $P \leq \alpha$? How does this relate to the interpretation of the P-value?

Solutions:
Since $P \sim U(0,1)$, we have

$$P(P \leq \alpha) = \alpha\,.$$

This means that, under the null hypothesis and under infinite replications, the P-value will be less than α approximately $100 \times \alpha\%$ of the time. Relating this to the definition of the P-value, this tells us that under the null hypothesis and under infinite replications, we can expect to see our test statistic or more extreme approximately $100 \times \alpha\%$ of the time.

---

2. How would you use the theorem that if $U = F_Y(y)$, then $U \sim U(0,1)$, to generate random simulations from the distribution Y.

Solutions:
It is relatively easy to generate a random number between 0 and 1, almost every program language under the sun can do it. So if u is a random observation from $U(0,1)$, then

$$F_Y^{-1}(u) = y$$

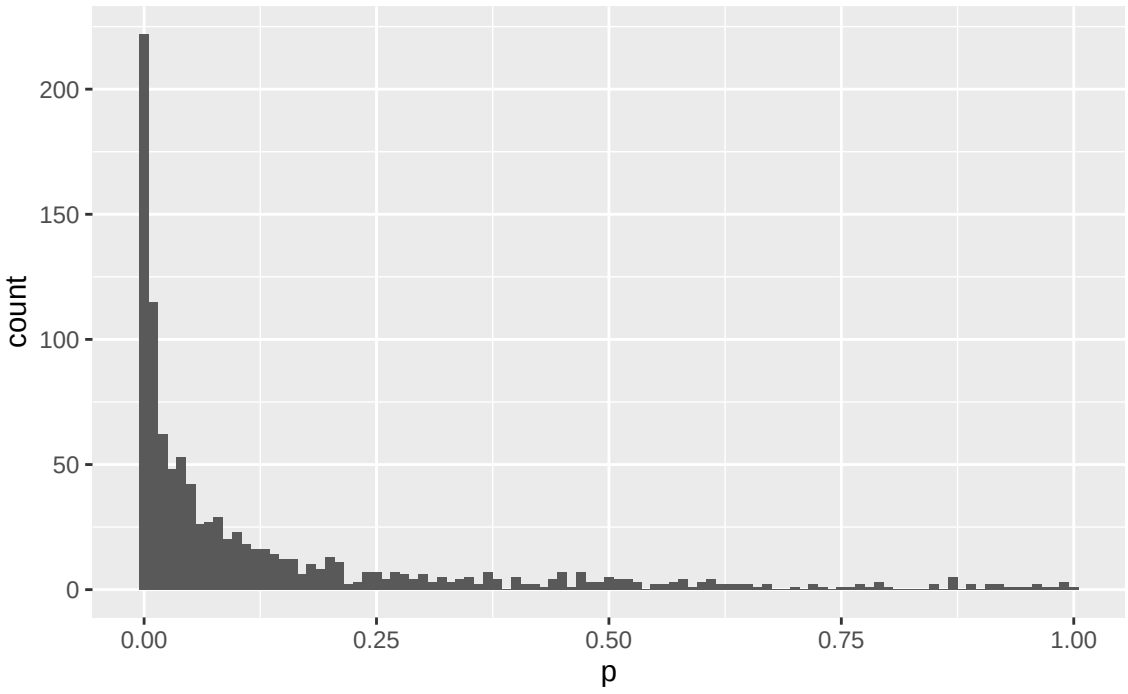
is a random observation from Y. This is known as the inverse-CDF transform method, and you will investigate this in Mathematical Statistics III next year.

---

3. How does the distribution of the P-value change if the null hypothesis is false?

Solutions:
You will find that the distribution will become more heavily distributed around 0 under the alternative hypothesis.

```
# Define a function for easy repeatability
# I have taken the vectorised code, and turned it into a function
# I have set default values for mu, s, and n so all we need to give it is the number of simulations
# defined be N
generate_p_value_alt <- function(N, mu = 0, s = 1, n = 100, alt = NULL){
  if(is.null(alt)){
    p <- N %>%
      rerun(rnorm(n = n, mean = mu, sd = s)) %>%
      map_dbl(function(x){
        z <- (mean(x) - mu)/(s/sqrt(n))
        p <- 2*pnorm(-abs(z))
        return(p)
      })
    return(p)
  }else{
    p <- N %>%
      rerun(rnorm(n = n, mean = alt, sd = s)) %>%
      map_dbl(function(x){
        z <- (mean(x) - mu)/(s/sqrt(n))
        p <- 2*pnorm(-abs(z))
        return(p)
      })
    return(p)
  }
}

p <- generate_p_value_alt(1000, alt = 0.2)

# Plot
ggplot(tibble(p = p), aes(x = p)) +
  geom_histogram(binwidth = 0.01)
```


200
150
100
50
0
count
0.00
0.25
0.50
0.75
1.00
p