Recitation 2

Drew Rosenberg

What is a Simulation?

Simulations are a way to use your computer like a laboratory. Many statistical principles such as bias and consistency are based theoretically on the idea of repeated sampling. In other words, if we do things over and over again, what would the answer look like?

Survey Example

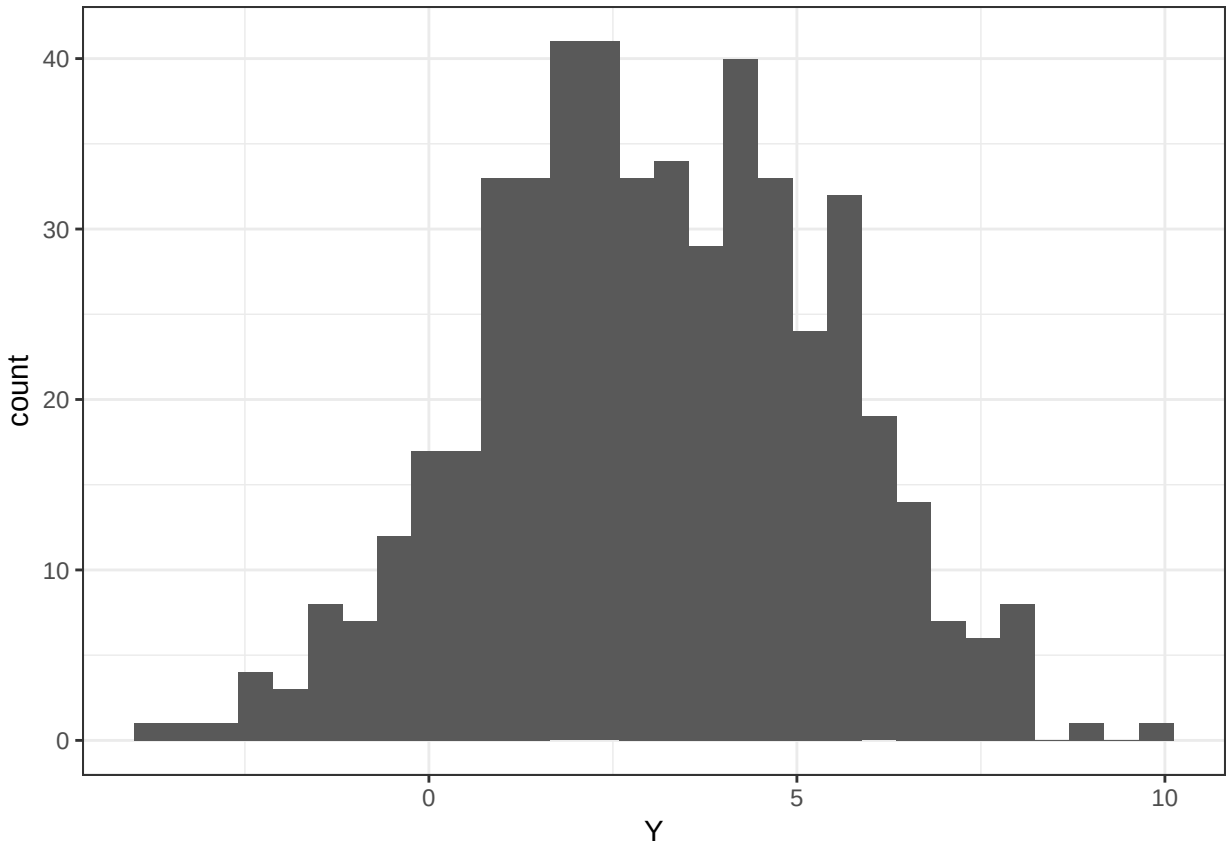
Let's say we were running a survey. We would really like to know how our results would change if we gave the same set of survey questions to a different sample of people. How would our results change? If we do this enough times—draw enough samples of people—we get a distribution of effect sizes. However, we almost always can't do this because we either don't have enough money, or we can't re-run WWII, so we can use our computer!

Data Generating Process and Simulation

When we use statistical models, we assume that our data are generated by a particular data generating process (DGP). A DGP is the mechanism that characterizes the population from which we draw samples. For example, if we are running an OLS regression, we are assuming our data are Gaussian distributed with this functional form: $Y = X\beta + \epsilon$. So, when we simulate many different samples from our hypothetical population, we generate them in this way.

```
# DGP for OLS
library(ggplot2)
n <- 500
X1 <- rnorm(n)
X2 <- rnorm(n)
b0 <- 3
b1 <- -2
b2 <- 0.5
Y <- b0 + b1*X1 + b2*X2 + rnorm(n)
qplot(Y, geom = "histogram") + theme_bw()
```

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



Remember that this is only one sample! We can use this same code to generate as many samples as we want. In fact, we will do this in a little bit in order to see if OLS works.

Central Limit Theorem in Action

But first, let's use simulation to show how the Central Limit Theorem works. We remember what that means, right? Essentially, the CLT says that if you take a bunch of samples from a distribution and sum them up or take their average, then the distribution of those sums will be Gaussian.

Let's pretend everyone in this class (20 people?) has a quarter and is standing on the center line of a football pitch. Then, when I say so, you start flipping your coin. Every time it comes up heads, you take a step toward the left goal and everytime it comes up tails, you step toward the right. In this experiment, everyone flips their coin 16 times, moves each time, then stands still. We can't really say with any certainty where any given person will end up, but we can be super confident what the distribution of final positions will look like.

We can even be agnostic about the size of a given step. Let's assume that a step is between 0 and 1 feet, but need not be the same. So, each person flips 16 times. What does one person look like?

```
sum(runif(16, -1, 1))
```

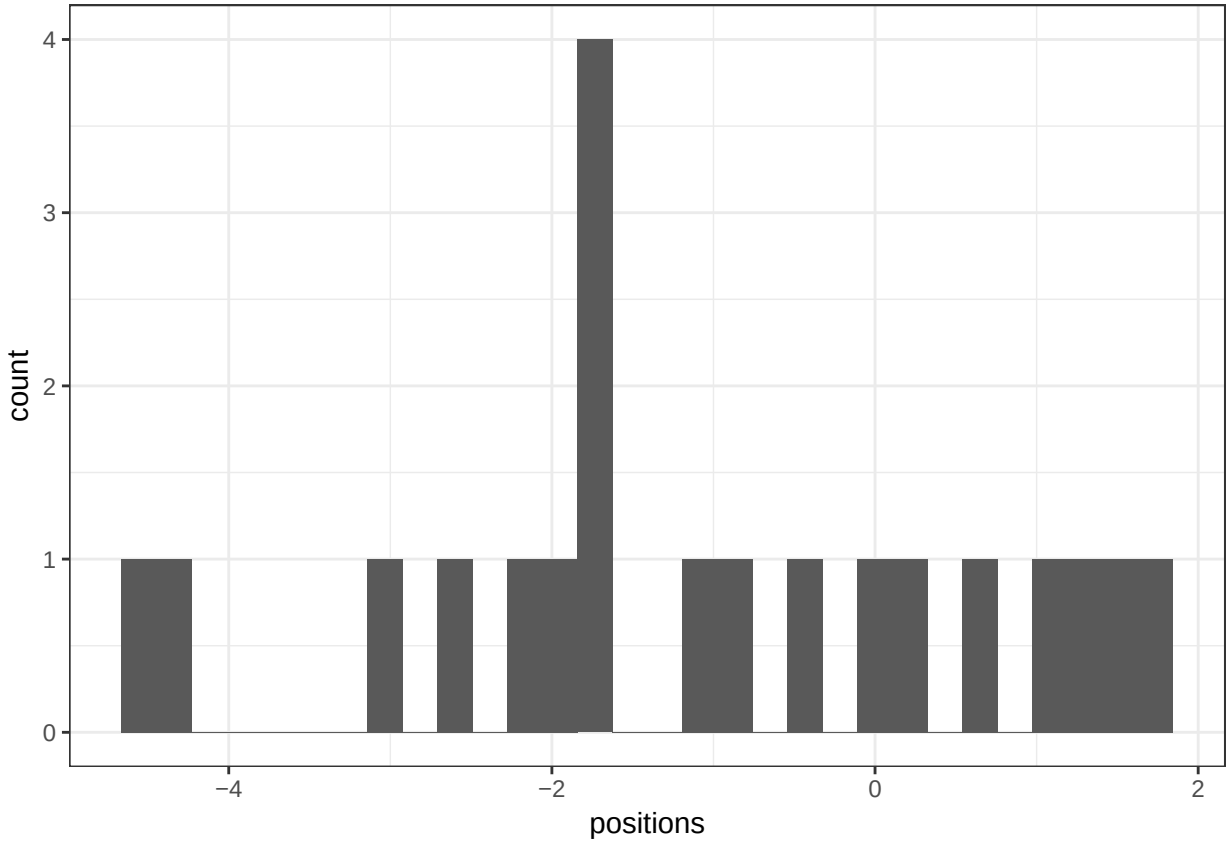
```
## [1] -1.338652
```

So this is the final position of one person. We calculated all the steps and added them up to get the final position. Now, let's simulate this same procedure for 20 people.

```
library(ggplot2)
positions <- replicate(20, sum(runif(16, -1, 1)))
```

```
qplot(positions, geom = "histogram") + theme_bw()
```

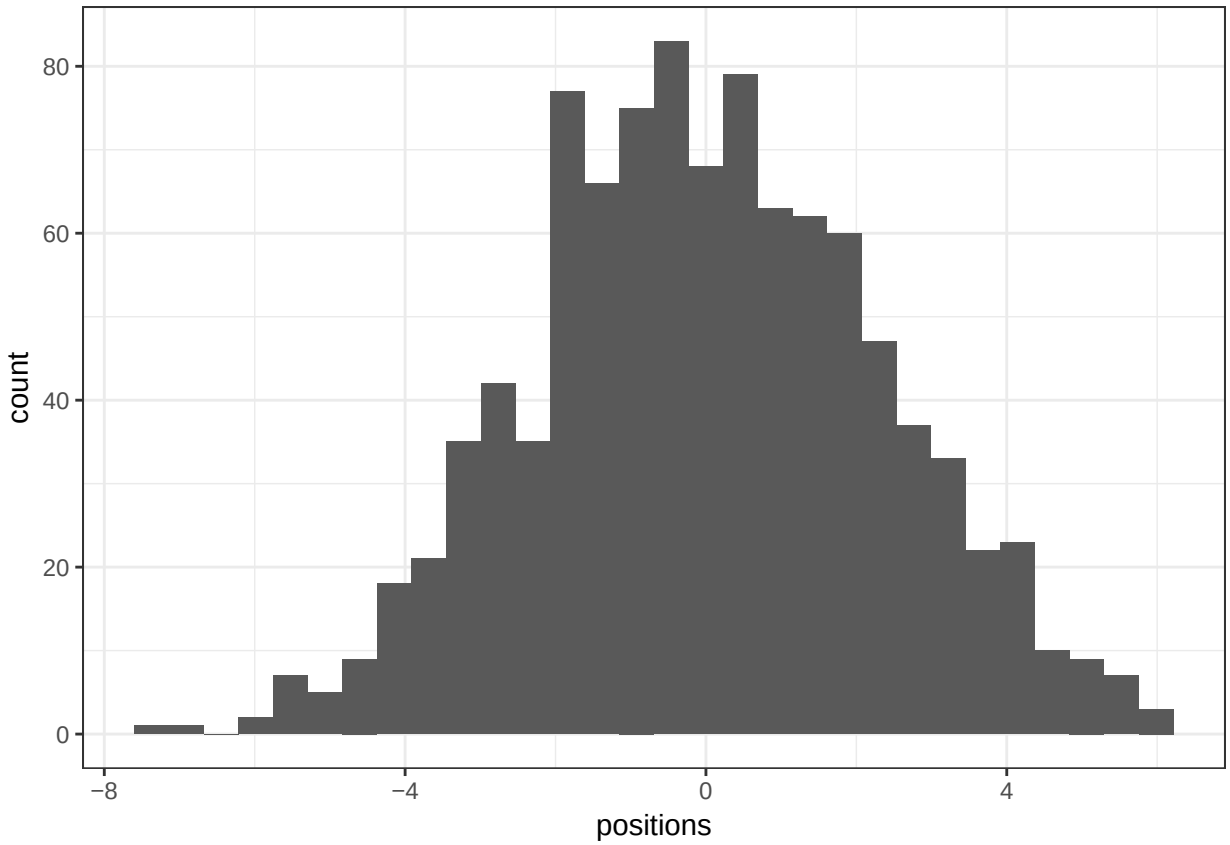
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



That's only one person, so let's crank up the number of simulations a bit.

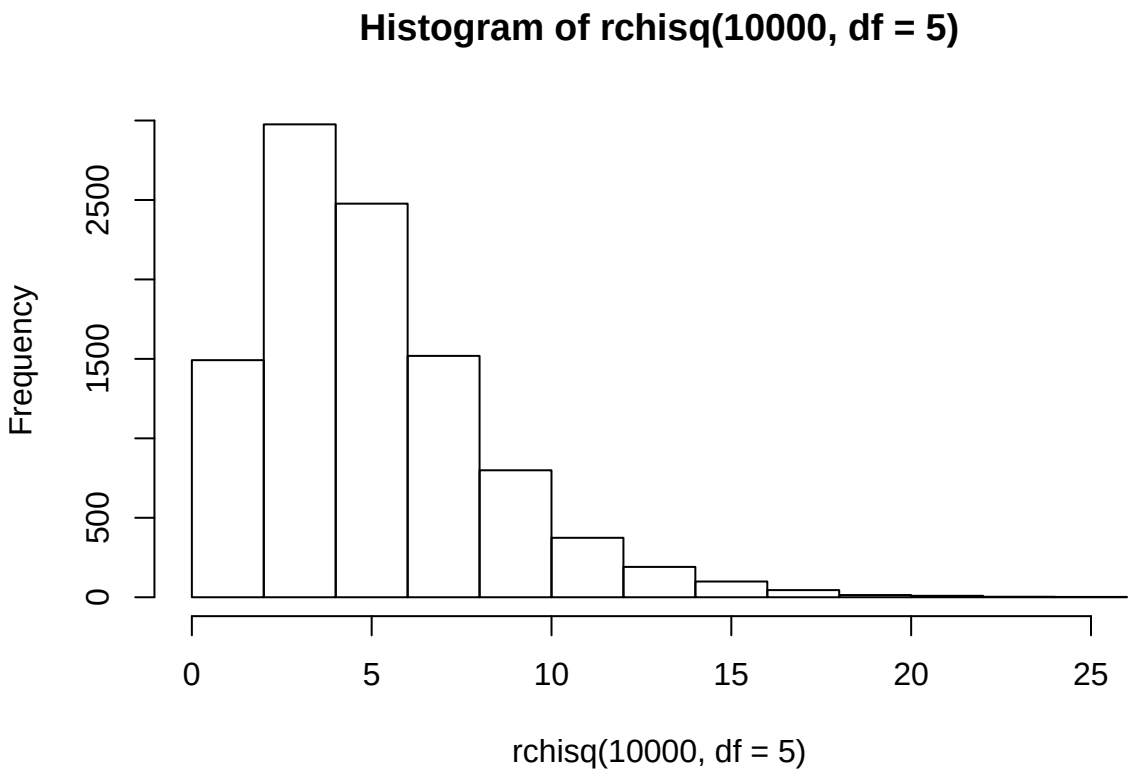
```
library(ggplot2)
positions <- replicate(1000, sum(runif(16, -1, 1)))
qplot(positions, geom = "histogram") + theme_bw()
```

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

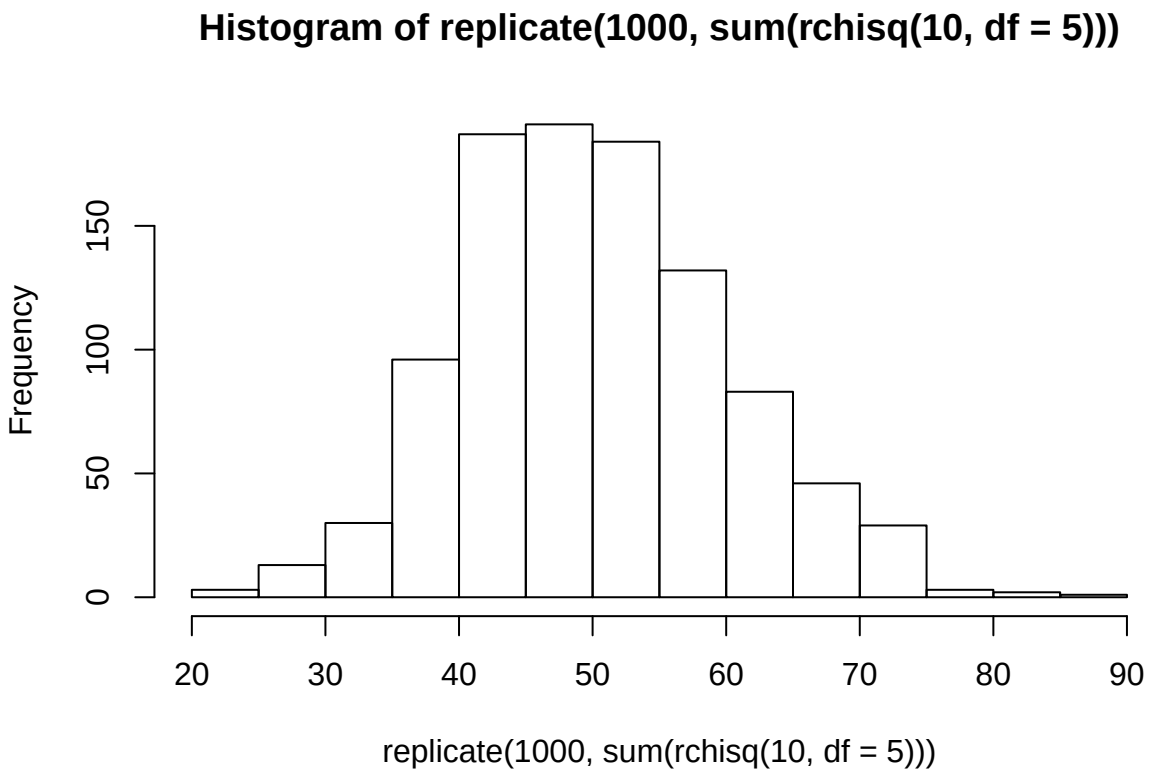


This is pretty cool. Let's do one more example. I am going to sample from the chisq distribution 100 times, add them together, and plot the histogram.

```
hist(rchisq(10000, df = 5))
```




```
hist(replicate(1000, sum(rchisq(10, df = 5))))
```



Birthday Problem

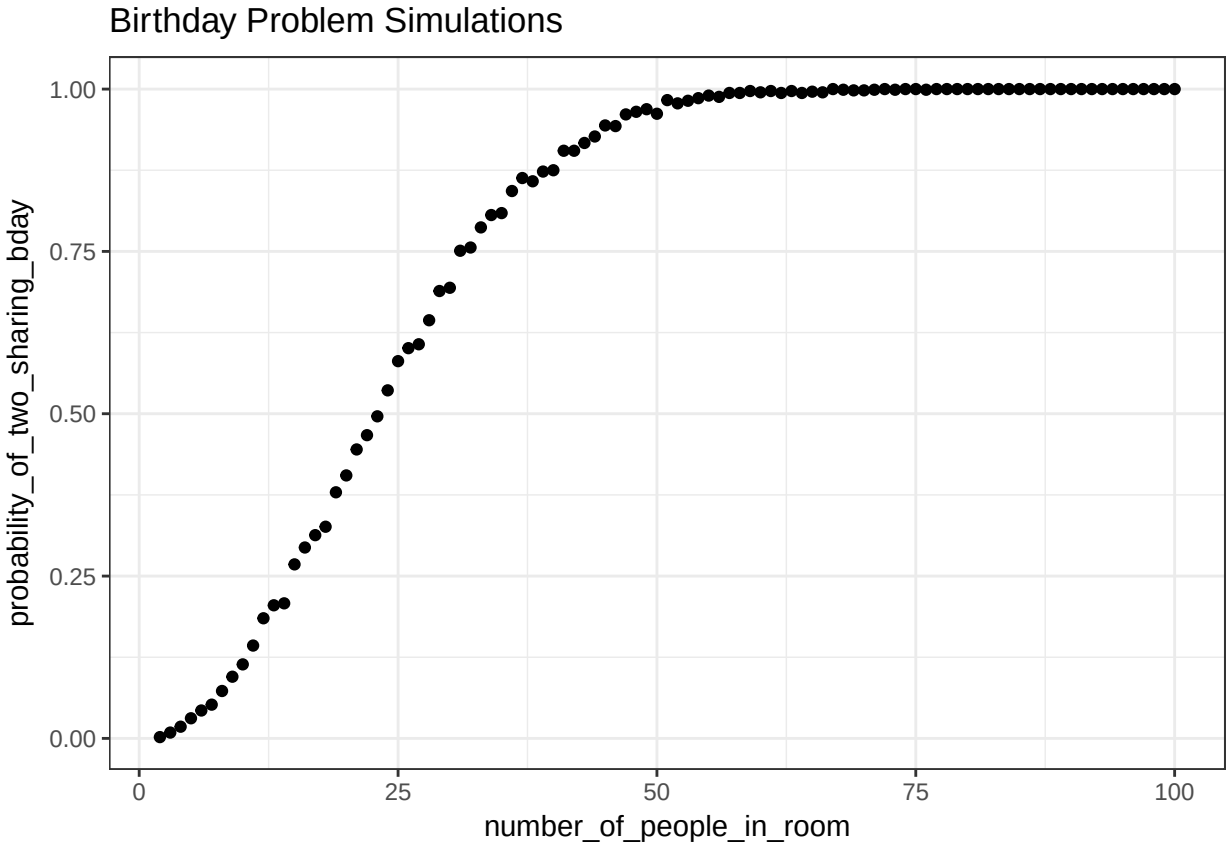
As we have seen, we can use simulation to find out the probability that any two people in a room of 24 have the same birthday. However, I'm going to show you a slick way of doing this. In addition, I'll figure out how this probability changes with the size of the room.

```
library(ggplot2)
library(data.table)
birthday_problem <- function(npeople,
  seed = sample.int(.Machine$integer.max, 1))
{
  days <- seq(1, 365, 1)
  room <- sample(days, npeople, replace = TRUE)
  ifelse(length(unique(room)) < npeople, 1, 0)
}
#birthday_problem(npeople = 24)

simulate_birthday_problem <- function(nsims, npeople)
{
  sum(replicate(nsims, birthday_problem(npeople = npeople)))/nsims
}
simulate_birthday_problem(nsims = 1000, npeople = 24)
```

```
## [1] 0.565
```

```
bday_probs <- sapply(seq(2, 100, 1), simulate_birthday_problem, nsims = 1000)
plot_data <- data.table(number_of_people_in_room = seq(2, 100, 1),
  probability_of_two_sharing_bday = bday_probs)
ggplot(plot_data, aes(x = number_of_people_in_room,
  y = probability_of_two_sharing_bday)) +
  geom_point() + theme_bw() +
  ggtitle("Birthday Problem Simulations")
```



As our final trick, let's use simulations to show that OLS works. We'll build on the work we did above. First, we will write down the DGP for our data: it's normally distributed with some set parameter values (if this doesn't make perfect sense, don't worry! We'll come back to this often this semester). Second, we will draw a sample, and write down the coefficients. Third, we will do this a lot of times and show that OLS gives us the proper answers.

OLS works!

Now let's verify that OLS gives us the right number in repeated samples!

```
test_ols <- function(n, b0, b1, b2, seed =
  sample(.Machine$integer.max, 1))
{
  x1 <- rnorm(n)
  x2 <- rnorm(n)
  y <- b0 + b1 * x1 + b2 * x2 + rnorm(n, 0, 1)
  coef(lm(y ~ x1 + x2))
```

```
}
test_ols(n = 100, b0 = 3, b1 = -2, b2 = 0.5)
```

```
## (Intercept)          x1          x2
##   2.7480324  -2.0108769   0.5161914
```

```
simulations <- t(replicate(1000, test_ols(n = 1000, b0 = 3, b1 = -2, b2 = 0.5)))
plot_data <- melt(simulations)
real_answer_data <- data.frame(v1 = c(3, -2, 0.5),
  Var2 = levels(plot_data$Var2))
ggplot(plot_data, aes(value)) + geom_histogram() +
  facet_grid(~ Var2, scales = "free") +
  geom_vline(aes(xintercept = v1), data = real_answer_data) + theme_bw()
```

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