Stat405

Simulation

Hadley Wickham

1. For loops
2. Hypothesis testing
3. Simulation

For loops

```r
# Common pattern: create object for output,
# then fill with results

cuts <- levels(diamonds$cut)
means <- rep(NA, length(cuts))

for(i in seq_along(cuts)) {
  sub <- diamonds[diamonds$cut == cuts[i], ]
  means[i] <- mean(sub$price)
}

# We will learn more sophisticated ways to do this
# later on, but this is the most explicit
```

```
1:5
seq_len(5)

1:10
seq_len(10)

1:0
seq_len(0)

seq_along(1:10)
1:10 * 2
seq_along(1:10 * 2)
```

Your turn

For each diamond colour, calculate the median price and carat size

```r
colours <- levels(diamonds$color)
n <- length(colours)
mprice <- rep(NA, n)
mcarat <- rep(NA, n)

for(i in seq_len(n)) {
  set <- diamonds[diamonds$color == colours[i], ]
  mprice[i] <- median(set$price)
  mcarat[i] <- median(set$carat)
}

results <- data.frame(colours, mprice, mcarat)
```

Back to slots...

For each row, calculate the prize and save it, then compare calculated prize to actual prize

Question: given a row, how can we extract the slots in the right form for the function?

```r
slots <- read.csv("slots.csv")

i <- 334
slots[i, ]
slots[i, 1:3]
str(slots[i, 1:3])

slots <- read.csv("slots.csv", stringsAsFactors = F)
str(slots[i, 1:3])
as.character(slots[i, 1:3])

calculate_prize(as.character(slots[i, 1:3]))
```

```
# Create space to put the results
slots$check <- NA

# For each row, calculate the prize
for(i in seq_len(nrow(slots))) {
  w <- as.character(slots[i, 1:3])
  slots$check[i] <- calculate_prize(w)
}

# Check with known answers
subset(slots, prize != check)
# Uh oh!
```

```r
# Create space to put the results
slots$check <- NA

# For each row, calculate the prize
for(i in seq_len(nrow(slots))) {
  w <- as.character(slots[i, 1:3])
  slots$check[i] <- calculate_prize(w)
}

# Check with known answers
subset(slots, prize != check)
# Uh oh!
```

What is the problem? Think about the most general case

DD	DD	DD	800
7	7	7	80
BBB	BBB	BBB	40
BB	BB	BB	25
B	B	B	10
C	C	C	10
Any bar	Any bar	Any bar	5
C	C	*	5
C	*	C	5
C	*	*	2
*	C	*	2
*	*	C	2

DD doubles any winning combination. Two DD quadruples. **DD is wild**

Hypothesis testing

Goal

Casino claims that slot machines have prize payout of 92%, but payoff for the 345 we observed is 67%. Is the casino lying?

(House advantage of 8% vs. 33%)

(Big caveat: today we're using a prize calculation function we know to be incorrect)

Q: What does it mean to have prize payout of 92%?

A: If we play the slot machine an infinite number of times, our average prize would be $0.92

Strategy 1

Play the slot machine an infinite number of times. If the average prize is not $0.92, reject the casino's claim.

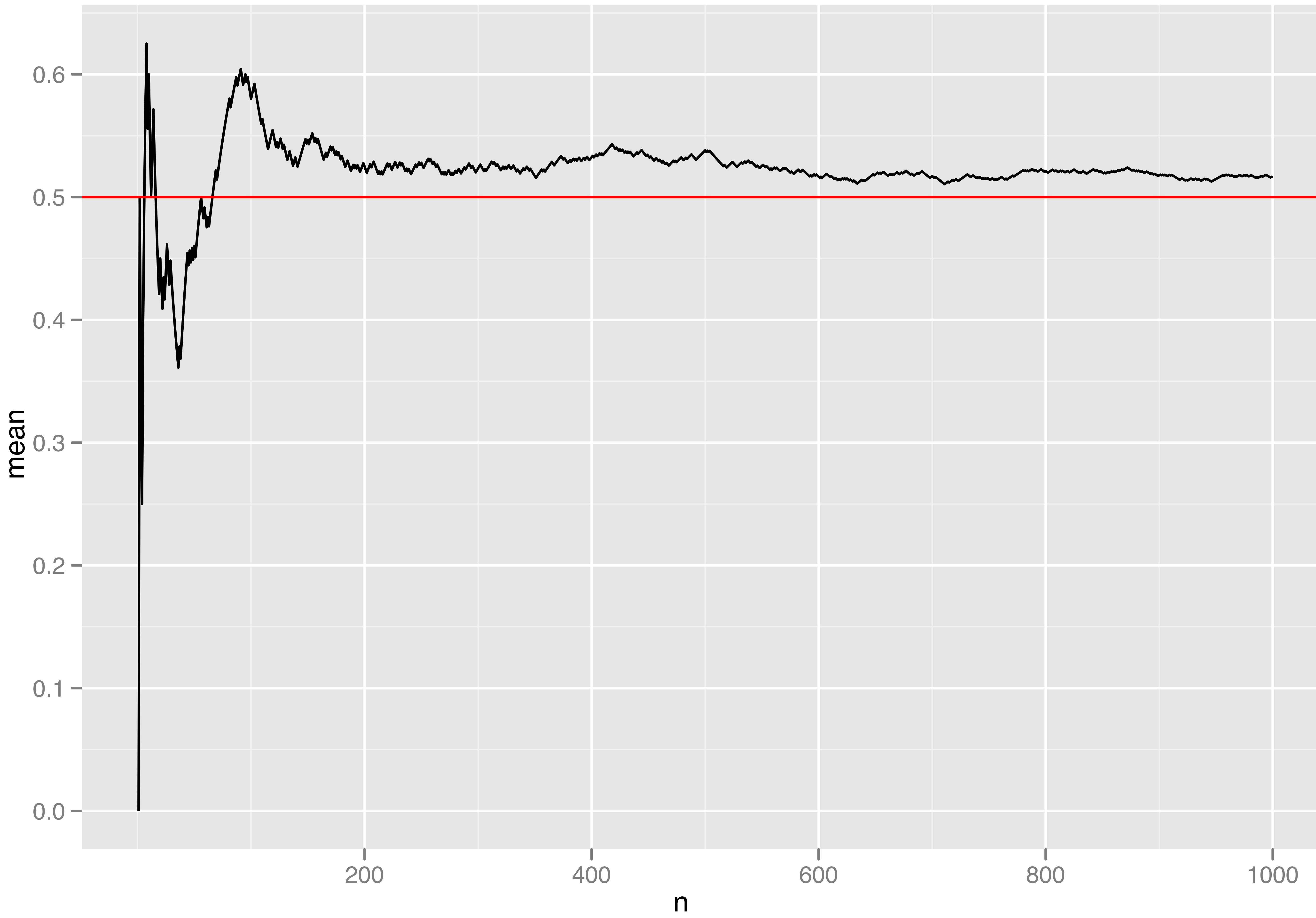
But...

```
# Let’s make a virtual coin flip
# 1 = heads, 0 = tails
coin <- c(0, 1)

# we can flip the coin once
flips <- sample(coin, 1, replace = T)
mean(flips)

# we can flip the coin many times
flips <- sample(coin, 10, replace = T)
mean(flips)
```

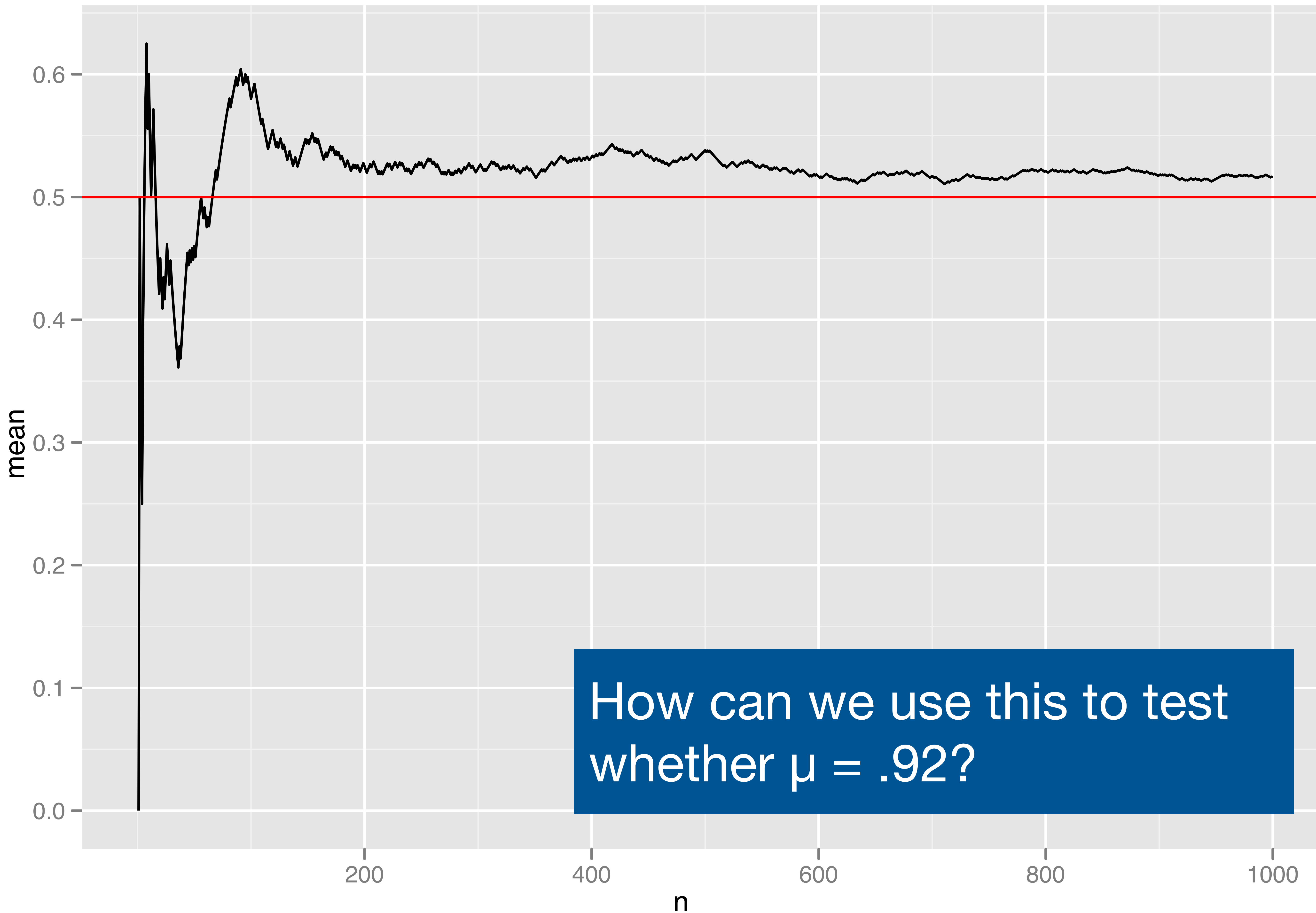
```
# what happens to the proportion of heads as n
# increases?

flips <- sample(coin, 10000, replace = T)
n <- seq_along(flips)
mean <- cumsum(flips) / n
coin_toss <- data.frame(n, flips, mean)

library(ggplot2)
qplot(n, mean, data = coin_toss, geom = "line") +
  geom_hline(yintercept = 0.5)
```

```
# what happens to the proportion of heads as n
# increases?

flips <- sample(coin, 10000, replace = T)
n <- seq_along(flips)
mean <- cumsum(flips) / n
coin_tos                (n, flips, mean)

library(ggplot2)
qplot(n, mean, data = coin_toss, geom = "line") +
  geom_hline(yintercept = 0.5)
```

cumulative sum



mean
0.0
0.1
0.2
0.3
0.4
0.5
0.6
200
400
600
800
1000
n



How can we use this to test whether μ = .92?

Strategy 2

Play the slot machine a large number of times. If the average prize is “far” from \$0.92, reject the casino’s claim.

Simulation

```
slots <- read.csv("slots.csv", stringsAsFactors = FALSE)

calculate_prize <- function(windows) {
  payoffs <- c("DD" = 800, "7" = 80, "BBB" = 40,
    "BB" = 25, "B" = 10, "C" = 10, "0" = 0)

  same <- length(unique(windows)) == 1
  allbars <- all(windows %in% c("B", "BB", "BBB"))

  if (same) {
    prize <- payoffs[windows[1]]
  } else if (allbars) {
    prize <- 5
  } else {
    cherries <- sum(windows == "C")
    diamonds <- sum(windows == "DD")

    prize <- c(0, 2, 5)[cherries + 1] *
      c(1, 2, 4)[diamonds + 1]
  }
  prize
}
```

Your turn

Write a function that simulates one pull on the slot machine (i.e, it should randomly choose a value from `slots$w1`, a value from `slots$w2`, and a value from `slots$w3` then calculate the prize)

Remember: solve the problem THEN write a function

```r
# Simulate the first window
sample(slots$w1, 1)

# Simulate the second window
sample(slots$w2, 1)

# Simulate the third window
sample(slots$w3, 1)

# What is the implicit assumption here?
# How could we test that assumption?
```

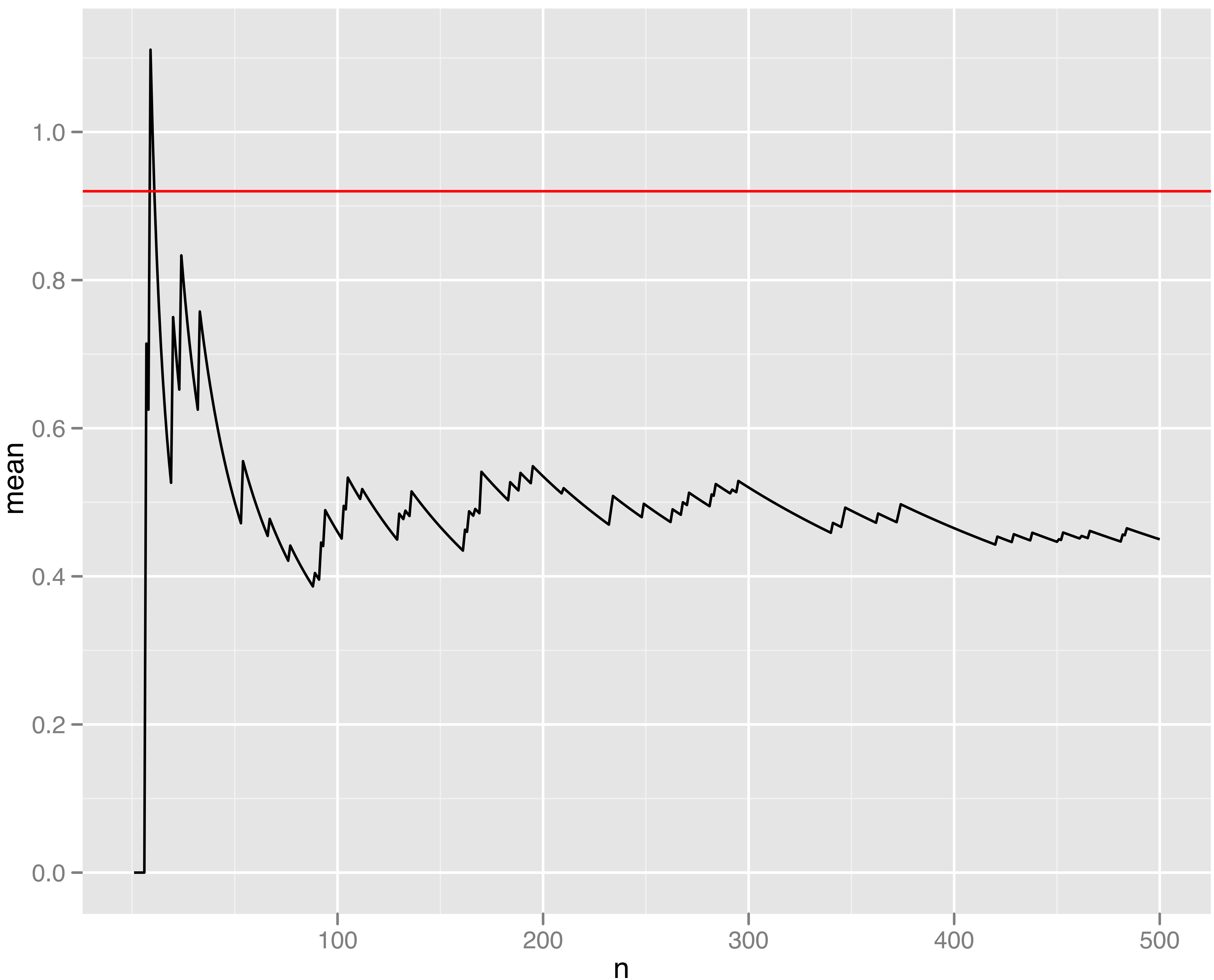
```
play_once <- function() {
  w1 <- sample(slots$w1, 1)
  w2 <- sample(slots$w2, 1)
  w3 <- sample(slots$w3, 1)

  calculate_prize(c(w1, w2, w3))
}
```

Your turn

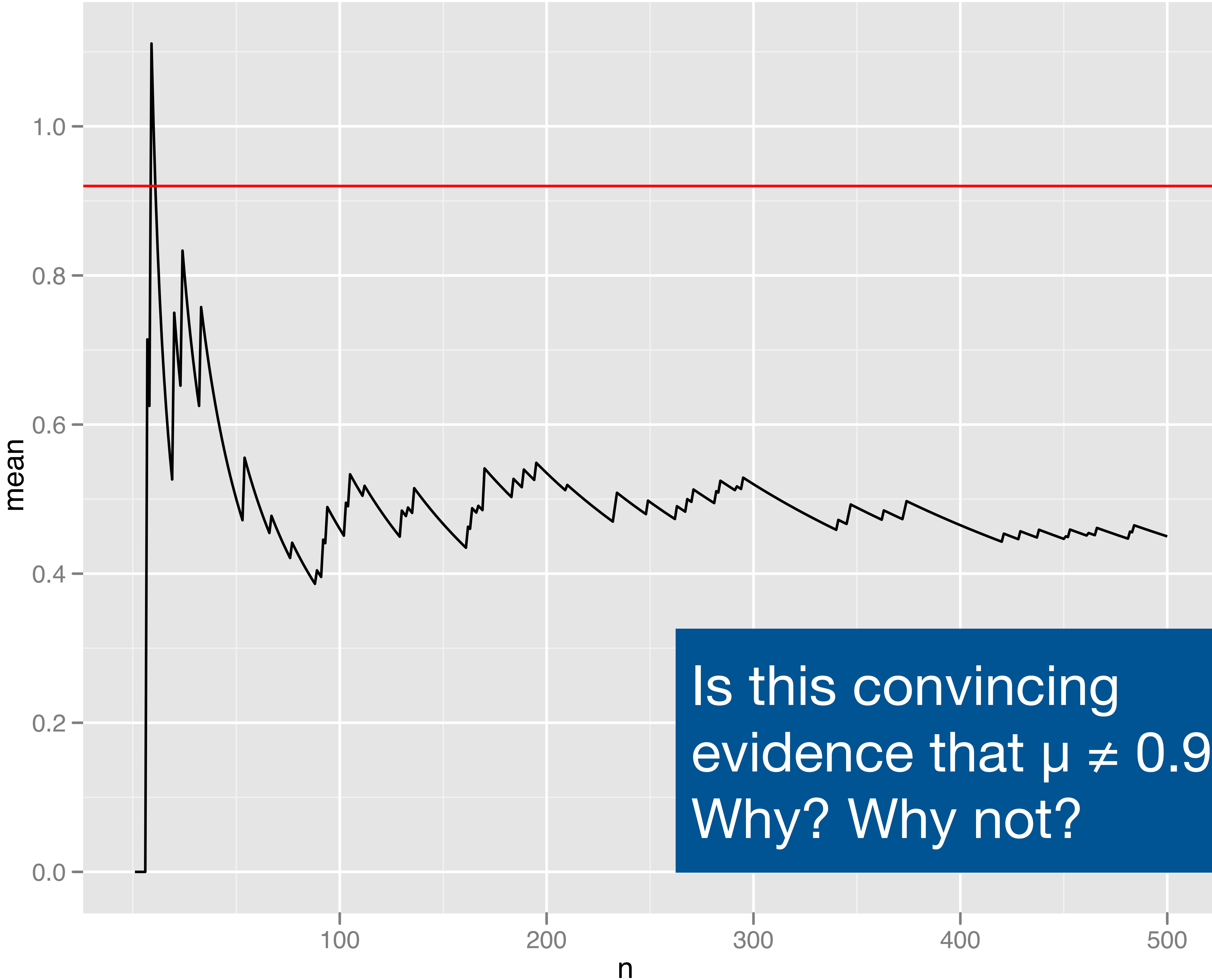
But we need to play the slot machine many times. Create a new function that plays `n` times and return `n` prizes. Call it `play_n`

```
play_n <- function(n) {
  prizes <- rep(NA, n)
  for(i in seq_len(n)) {
    prizes[i] <- play_once()
  }
  prizes
}
```

```
# Now we can see what happens to the mean prize as
# n increases

games <- data.frame(prizes = play_n(500))
games <- mutate(games,
  n = seq_along(prizes),
  avg = cumsum(prizes) / n)

qplot(n, avg, data = games, geom = "line") +
  geom_hline(yintercept = 0.92, color = "red")
```



1.0
0.8
0.6
0.4
0.2
0.0
mean
100
200
300
400
500
n



Is this convincing evidence that μ ≠ 0.92? Why? Why not?

Questions

Is 500 pulls enough?

What do other realisations look like?

How can we do this more quickly?

```
# Current function is pretty slow
system.time(play_n(5000))

# I wrote a vectorised version - instead of
# using explicit for loops, use R functions that
# work with vectors.  This is usually much much
# faster
source("payoff-v.r")
system.time(play_many(5000))
```

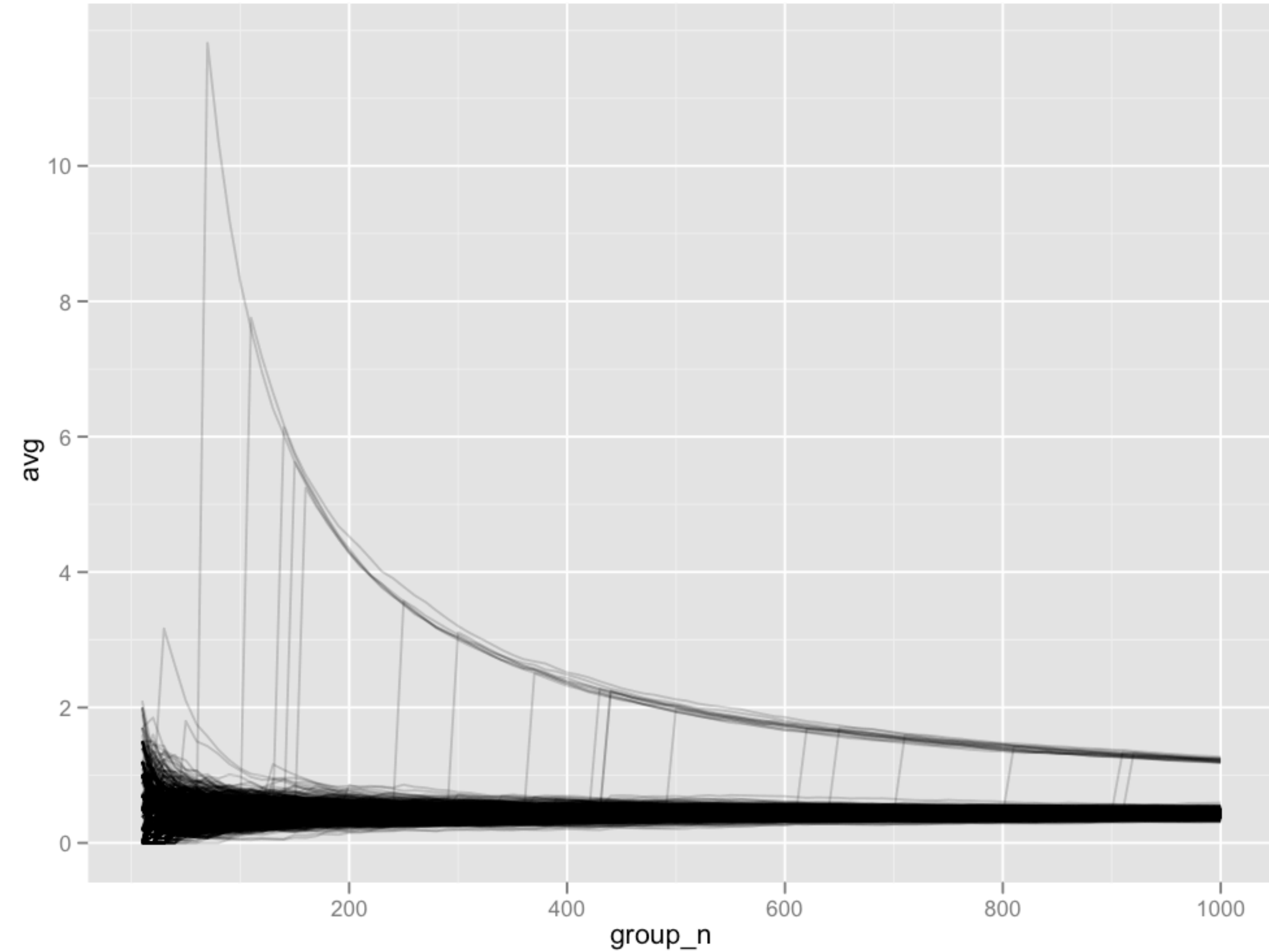
```
# What happens if we play more games?

games <- data.frame(prizes = play_many(10^6))
games <- mutate(games,
  n = seq_along(prizes),
  avg = cumsum(prizes) / n)

every1000 <- subset(games, n %% 1000 == 0)
qplot(n, avg, data = every1000, geom = "line")
qplot(n, avg, data = subset(every1000, n > 10000),
  geom = "line")

# Still seems to be quite a lot of variation even
# after 1,000,000 pulls
```

%%	remainder
%/%	integer division

```
seq_len(100) %% 5
seq_len(100) %/% 5

seq_len(100) %% 10
seq_len(100) %/% 10

seq_len(100) %% 11
seq_len(100) %/% 11
```

```
# How can we characterise the amount of variation?
# We could do multiple runs and look at the
# distribution at multiple points

# Turn our million pulls into 1,000 sessions of
# 1,000 pulls

many <- mutate(games,
  group = (n - 1) %/% 1000 + 1,
  group_n = (n - 1) %% 1000 + 1)

# How do we calculate the average?  Just looking
# at the cumulative sum will no longer work
```

```
# New function: ave
# ave takes the first argument, divides it into
# pieces according to the second argument, applies
# FUN to each piece, and joins them back together

many$avg <- ave(many$prize, many$group,
 FUN = cumsum) / many$group_n

every10 <- subset(many, group_n %% 10 == 0)
qplot(group_n, avg, data = every10, geom = "line",
  group = group, alpha = I(1/5))
```



avg
10
8
6
4
2
0
200
400
600
800
1000
group_n

```
# Could just look at the distribution at pull
# 1000

final <- subset(many, group_n == 1000)
qplot(avg, data = final, binwidth = 0.01)

# What do you think the average payoff is?

# This basic technique is called bootstrapping.
```