

Simulation - part 1

STAT 133

Gaston Sanchez

Department of Statistics, UC–Berkeley

`gastonsanchez.com`

`github.com/gastonstat/stat133`

Course web: gastonsanchez.com/stat133

How do we use a computer to
simulate a chance process?

Case Study

Chevalier de Mere

- ▶ Antoine Gombaud (1607 - 1684)
- ▶ *Nom de plume* “Chevalier de Mere”
- ▶ French writer and gambler, but not a nobleman
- ▶ Amateur mathematician



De Mere's games

Game 1

- ▶ One die
- ▶ Four rolls
- ▶ Win: at least one 6

De Mere's games

Game 1

- ▶ One die
- ▶ Four rolls
- ▶ Win: at least one 6

Game 2

- ▶ Two dice
- ▶ 24 rolls
- ▶ Win: at least one double 6

De Mere's games

Game 1

- ▶ One die
- ▶ Four rolls
- ▶ Win: at least one 6

Game 2

- ▶ Two dice
- ▶ 24 rolls
- ▶ Win: at least one double 6

De Mere was making money with game 1, but losing money with game 2. He turned to Blaise Pascal for help.

De Mere's faulty reasoning

Game 1

- ▶ The chance of getting a six in one roll of a die is $\frac{1}{6}$.
- ▶ In four rolls of a die, the chance of getting one six would be $\frac{4}{6} = \frac{2}{3}$.

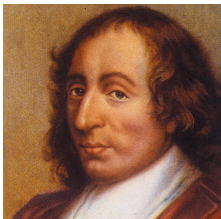
De Mere's faulty reasoning

Game 2

- ▶ The chance of getting a double six in rolling a pair of dice is $\frac{1}{36}$.
- ▶ In 24 rolls of a pair of dice, the chance of getting one double six would be $\frac{24}{36} = \frac{2}{3}$.

Pascal and Fermat

- ▶ De Mere turned into Blaise Pascal
- ▶ Pascal consulted with Pierre de Fermat
- ▶ Beginning of Pascal's and Fermat's famous correspondence (1650's)
- ▶ Origin of combinatorial probability



Blaise Pascal
(1623 - 1662)
French mathematician



Pierre de Fermat
(1601 - 1665)
French lawyer and amateur
mathematician

Letter from Pascal to Fermat

"If one undertakes to throw a six with one die, the advantage of undertaking it in 4 throws is as 671 to 625. If one undertakes to throw a double-six with two dice, there is a disadvantage of undertaking it in 24 throws. And nevertheless 24 is to 36 (which is the number of faces of two dice) as 4 to 6 (which is the number of faces of one die)."

De Mere's Game 1

Probability of no six in four rolls:

$$\frac{5}{6} \times \frac{5}{6} \times \frac{5}{6} \times \frac{5}{6} = \frac{625}{1296} = 0.482253$$

De Mere's Game 1

Probability of no six in four rolls:

$$\frac{5}{6} \times \frac{5}{6} \times \frac{5}{6} \times \frac{5}{6} = \frac{625}{1296} = 0.482253$$

$P(\text{at least one six}) = 1 - \text{Probability of no six in four rolls:}$

$$1 - \frac{625}{1296} = \frac{671}{1296} = 0.517747$$

De Mere's Game 2

Probability of no double six in 24 rolls:

$$P(\text{no double six in 24 rolls}) = \left(\frac{35}{36}\right)^{24} = 0.5086$$

De Mere's Game 2

Probability of no double six in 24 rolls:

$$P(\text{no double six in 24 rolls}) = \left(\frac{35}{36}\right)^{24} = 0.5086$$

$P(\text{at least one double six in 24 rolls}) =$

$$1 - 0.5086 = 0.4914$$

Using a computer to simulate a
chance process

Some Questions

- ▶ Rather than solve the problem analytically, we can simulate 4 rolls of a die and count the number of 6's
- ▶ If we simulate rolling 4 dice many times, then the proportion of times we get 0, 1, 2, 3, or 4 sixes should be close to the chance of that many sixes on any 4 rolls

Some Questions

- ▶ What is the chance of getting one 6 when rolling a die?
- ▶ What is the chance of getting one 6 when rolling two dice?
- ▶ What is the chance of getting at least one 6 when rolling 4 dice?

What are the steps?

- ▶ Simulate rolling one die
- ▶ Simulate rolling a pair of dice
- ▶ Simulate rolling four dice
- ▶ Count the number of sixes

Simulating one die



- ▶ Let's start with one die
- ▶ What features?
- ▶ How to create a **die** object?

One die

```
die <- 1:6
```

```
die
```

```
## [1] 1 2 3 4 5 6
```


Your turn

Which option would produce an invalid die:

A

```
die <- 1L:6L
```

B

```
die <- seq(from = 1, to = 6)
```

C

```
die <- c(1, 2, 3, 4, 5, 6)
```

D

```
die <- seq(from = 1, to = 6, by = 6)
```

E

```
die <- seq_len(6)
```

How to simulate the roll of a die?

sample()

Very useful function for selecting from a discrete set (vector) of possibilities.

sample() arguments

- ▶ x
- ▶ size
- ▶ replace
- ▶ prob

Roll a die

```
# use of sample() to simulate the roll of a die  
sample(die, size = 1)  
  
## [1] 2
```

Roll a die

```
# use of sample() to simulate the roll of a die
```

```
sample(die, size = 1)
```

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

```
sample(die, size = 1)
```

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

```
sample(die, size = 1)
```

```
## [1] 5
```

```
sample(die, size = 1)
```

```
## [1] 6
```

Function: Roll one die

Write a function to make it more convenient

```
# function  
rolldie <- function() {  
  
}
```

Function: Roll one die

Write a function to make it more convenient

```
# function  
rolldie <- function() {  
  die <- 1:6  
  sample(die, size = 1)  
}
```

Function: Roll one die

Write a function to make it more convenient

```
# function  
rolldie <- function() {  
  die <- 1:6  
  sample(die, size = 1)  
}
```

```
rolldie()
```

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


Rolling one die

Let's simulate 10 rolls of a die

```
# roll 10 times  
for (i in 1:10) {  
  rolldie()  
}
```

Rolling one die

Let's simulate 10 rolls of a die

```
# roll 10 times  
for (i in 1:10) {  
  rolldie()  
}
```

What's *wrong*?

Why nothing is shown on screen?

Rolling one die

```
# roll 10 times  
for (i in 1:10) {  
  print(rolldie())  
}
```

```
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 6  
## [1] 2  
## [1] 6  
## [1] 6  
## [1] 4  
## [1] 4  
## [1] 1
```

Rolling one die

```
# roll until first 5  
repeat {  
  rol <- rolldie()  
  print(rol)  
  if (rol == 5) break  
}
```

```
## [1] 4  
## [1] 2  
## [1] 6  
## [1] 4  
## [1] 3  
## [1] 6  
## [1] 5
```

Rolling one die

```
# roll until first 5  
rol <- 1
```

```
while (rol != 5) {  
  rol <- rolldie()  
  print(rol)  
}
```

```
## [1] 4  
## [1] 2  
## [1] 6  
## [1] 4  
## [1] 3  
## [1] 6  
## [1] 5
```

Rolling one die

```
# roll 10,000 times  
results <- numeric(10000)  
  
for (i in 1:10000) {  
  results[i] <- rolldie()  
}
```

Rolling one die

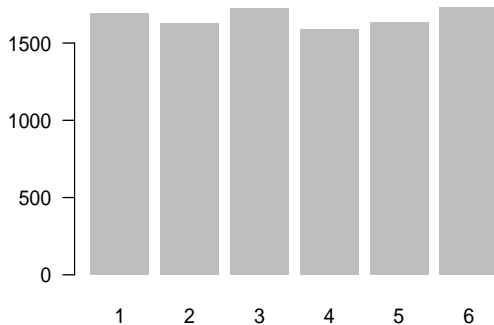
```
# frequencies  
table(results)
```

```
## results  
##      1      2      3      4      5      6  
## 1692 1631 1724 1586 1633 1734
```

```
# relative frequencies  
table(results) / 10000
```

```
## results  
##      1      2      3      4      5      6  
## 0.1692 0.1631 0.1724 0.1586 0.1633 0.1734
```

Distribution of results



How to simulate a **loaded** die?

Loaded Die

Changing argument **prob** to create a loaded die

```
probs <- c(1/21, 2/21, 3/21, 4/21, 5/21, 6/21)

# function
loaded <- function() {
  die <- 1:6
  sample(die, size = 1, prob = probs)
}
```

Loaded Die

Changing argument **prob** to create a loaded die

```
probs <- c(1/21, 2/21, 3/21, 4/21, 5/21, 6/21)
```

```
# function
```

```
loaded <- function() {  
  die <- 1:6  
  sample(die, size = 1, prob = probs)  
}
```

```
loaded()
```

```
## [1] 6
```

Rolling a loaded die

```
# roll 10 times  
for (i in 1:10) {  
  print(loaded())  
}
```

```
## [1] 6  
## [1] 5  
## [1] 4  
## [1] 2  
## [1] 6  
## [1] 2  
## [1] 2  
## [1] 4  
## [1] 4  
## [1] 6
```

Rolling a loaded die

```
# roll 10,000 times
results <- numeric(10000)

for (i in 1:10000) {
  results[i] <- loaded()
}
```

Rolling a loaded die

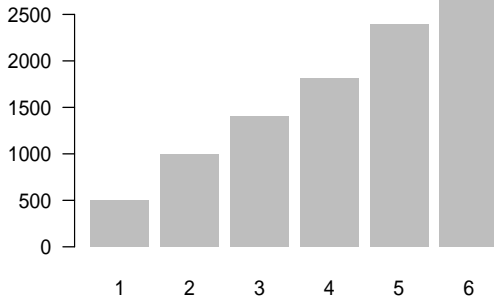
```
# frequencies  
table(results)
```

```
## results  
##      1      2      3      4      5      6  
## 505 1000 1405 1817 2391 2882
```

```
# relative frequencies  
table(results) / 10000
```

```
## results  
##      1      2      3      4      5      6  
## 0.0505 0.1000 0.1405 0.1817 0.2391 0.2882
```

Loaded Distribution



Simulating rolling a pair of dice



Roll a pair of dice

```
# 1st die  
sample(die, size = 1)
```

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

```
# 2nd die  
sample(die, size = 1)
```

```
## [1] 5
```

Function: roll a pair of dice

```
# pair of dice function  
roll2 <- function() {  
  
  
  
  
  
  
  
  
  
}
```

Function: roll a pair of dice

```
# pair of dice function
roll2 <- function() {
  die <- 1:6
  rol1 <- sample(die, size = 1)
  rol2 <- sample(die, size = 1)
  c(rol1, rol2)
}
```

Various rolls

```
roll2()
```

```
## [1] 5 1
```

```
roll2()
```

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

```
roll2()
```

```
## [1] 6 6
```

Function

```
# pair of dice function
roll2 <- function() {
  die <- 1:6
  rol1 <- sample(die, size = 1)
  rol2 <- sample(die, size = 1) # repeated command!
  c(rol1, rol2)
}
```

Roll a pair of dice

```
die <- 1:6  
  
# avoid repetition with one call of 'sample()'  
sample(die, size = 2)  
  
## [1] 6 1
```

Roll a pair dice

```
for (i in 1:15) {  
  print(sample(die, size = 2))  
}
```

```
## [1] 1 3  
## [1] 1 4  
## [1] 5 4  
## [1] 4 3  
## [1] 3 1  
## [1] 2 6  
## [1] 4 6  
## [1] 3 4  
## [1] 4 6  
## [1] 6 3  
## [1] 2 5  
## [1] 2 5  
## [1] 2 3  
## [1] 3 2  
## [1] 2 5
```

Can you spot a problem?

Roll a pair of dice

```
die <- 1:6  
  
# sample with replacement  
sample(die, size = 2, replace = TRUE)  
  
## [1] 4 1
```

Roll a pair of dice

```
for (i in 1:15) {  
  print(sample(die, size = 2, replace = TRUE))  
}
```

```
## [1] 6 6  
## [1] 6 5  
## [1] 5 4  
## [1] 3 6  
## [1] 1 2  
## [1] 1 5  
## [1] 1 2  
## [1] 3 1  
## [1] 6 1  
## [1] 1 2  
## [1] 1 6  
## [1] 3 2  
## [1] 3 1  
## [1] 2 2  
## [1] 1 1
```

Roll a pair of dice

```
# rewrite roll2()  
roll2 <- function() {  
  die <- 1:6  
  sample(die, size = 2, replace = TRUE)  
}
```

Roll a pair of dice

```
# rewrite roll2()  
roll2 <- function() {  
  die <- 1:6  
  sample(die, size = 2, replace = TRUE)  
}
```

```
roll2()
```

```
## [1] 5 4
```

Rolling a die any number of
times

Rolling several dice

More general function to roll a die any number of times

```
roll <- function(times = 1) {  
  die <- 1:6  
  sample(die, size = times, replace = TRUE)  
}
```

Rolling several dice

More general function to roll a die any number of times

```
roll <- function(times = 1) {  
  die <- 1:6  
  sample(die, size = times, replace = TRUE)  
}
```

```
roll()
```

```
## [1] 4
```

Rolling several dice

```
# default (one roll)
```

```
roll()
```

```
## [1] 4
```

```
# two rolls
```

```
roll(2)
```

```
## [1] 1 5
```

```
# 4 rolls
```

```
roll(4)
```

```
## [1] 6 4 6 5
```


De Mere's Game 1

De Mere's Game 1

Game 1

- ▶ One die
- ▶ Four rolls
- ▶ Win: at least one 6

De Mere's Game 1

```
# play 100 times
results <- matrix(0, nrow = 100, ncol = 4)

for (i in 1:100) {
  results[i, ] <- roll(times = 4)
}

head(results)
```

| ## | [,1] | [,2] | [,3] | [,4] |
|---------|------|------|------|------|
| ## [1,] | 2 | 5 | 4 | 2 |
| ## [2,] | 6 | 6 | 1 | 6 |
| ## [3,] | 3 | 4 | 4 | 2 |
| ## [4,] | 5 | 2 | 3 | 6 |
| ## [5,] | 6 | 2 | 3 | 1 |
| ## [6,] | 4 | 3 | 6 | 1 |

De Mere's Game 1

```
counts <- 0

for (i in 1:100) {
  if (any(results[i, ] == 6))
    counts <- counts + 1
}

# proportion of wins
counts / 100

## [1] 0.56
```

De Mere's Game 1

```
sixes <- apply(results, 1, function(x) sum(x == 6))  
  
table(sixes)  
  
## sizes  
##  0  1  2  3  
## 44 41 13  2
```

De Mere's Game 1

```
sixes <- apply(results, 1, function(x) sum(x == 6))
```

```
table(sixes)
```

```
## sixes
```

```
## 0 1 2 3
```

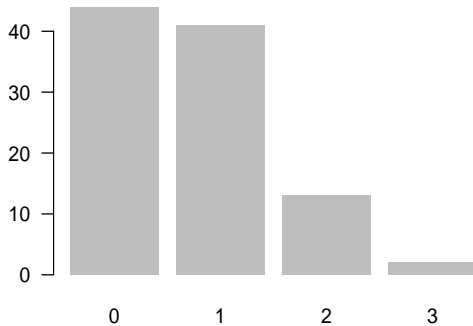
```
## 44 41 13 2
```

```
# rolls with at least one six
```

```
sum(table(sixes)[-1])
```

```
## [1] 56
```

De Mere's Game 1



De Mere's Game 1

Considerations

- ▶ How would you make the code more flexible?
- ▶ What type of “parameters”?
- ▶ Avoid Repetition

De Mere's Game 1

```
games <- 10000
results <- matrix(0, nrow = games, ncol = 4)

for (i in 1:games) {
  results[i, ] <- roll(times = 4)
}

counts <- 0
for (i in 1:games) {
  if (any(results[i, ] == 6))
    counts <- counts + 1
}

counts / games  # proportion of wins

## [1] 0.5204
```

De Mere's Game 1

```
sixes <- apply(results, 1, function(x) sum(x == 6))
```

```
table(sixes)
```

```
## sizes
```

```
##      0      1      2      3      4
```

```
## 4796 3858 1171  166    9
```

De Mere's Game 1

```
sixes <- apply(results, 1, function(x) sum(x == 6))
```

```
table(sixes)
```

```
## sixes
```

```
##      0      1      2      3      4
```

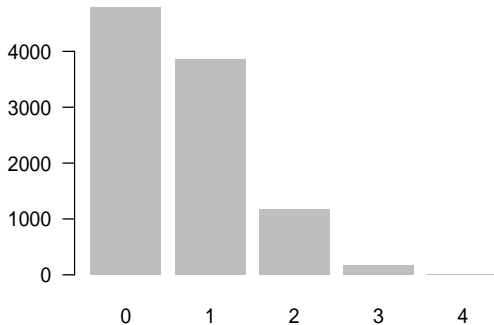
```
## 4796 3858 1171  166      9
```

```
# rolls with at least one six
```

```
sum(table(sixes)[-1])
```

```
## [1] 5204
```

De Mere's Game 1



De Mere's Game 2

De Mere's Game 2

Game 2

- ▶ Two dice
- ▶ 24 rolls
- ▶ Win: at least one double 6

De Mere's Game 2

```
roll2 <- function(times = 1) {  
  dice2 <- unlist(lapply(1:6, function(x) x + 1:6))  
  sample(dice2, size = times, replace = TRUE)  
}
```

```
roll2()
```

```
## [1] 10
```

```
roll2(24)
```

```
## [1] 8 4 4 3 8 6 7 6 9 9 5 4 5 7 3 11 9 11 10
```

De Mere's Game 2

Game 2

- ▶ It's better if we sum the points of rolling two dice
- ▶ Possible outcomes: $\{2, 3, 4, \dots, 10, 11, 12\}$
- ▶ Double six is equivalent to 12 points

De Mere's Game 2

```
games <- 10000
results <- matrix(0, nrow = games, ncol = 24)

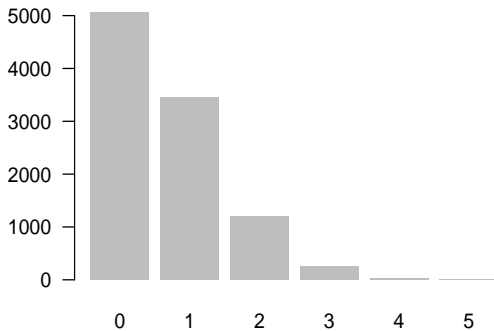
for (i in 1:games) {
  results[i, ] <- roll2(24)
}

doublesix <- apply(results, 1, function(x) sum(x == 12))

sum(table(doublesix)[-1])

## [1] 4930
```

De Mere's Game 2



De Mere's Game 2

```
counts <- 0
for (i in 1:games) {
  if (any(results[i, ] == 12))
    counts <- counts + 1
}

counts / games  # proportion of wins

## [1] 0.493
```