Webinar3

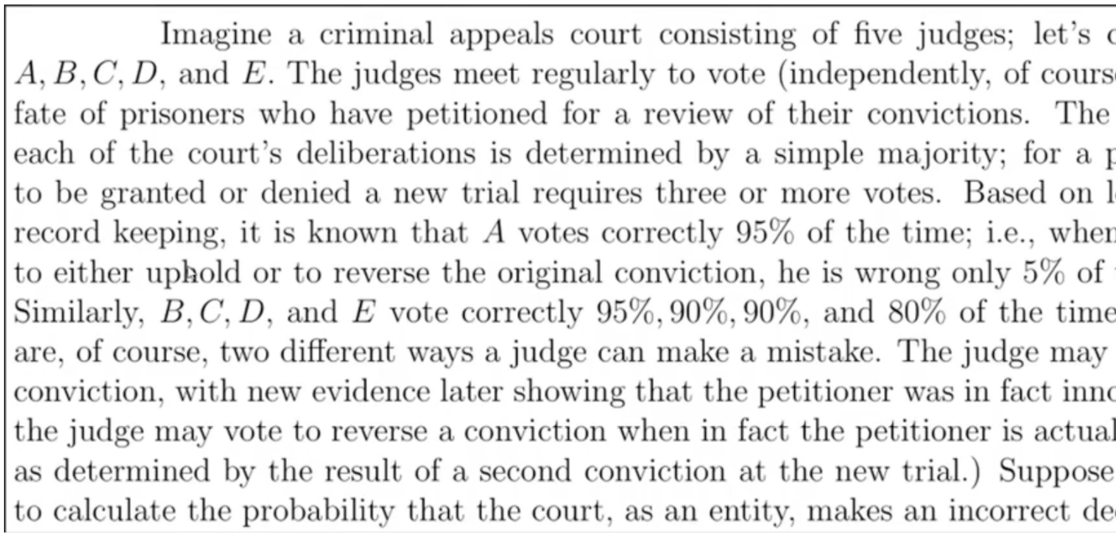
```r
library(ggplot2) # plotting library
library(dplyr)   # data wrangling library
```

Load libraries

```
##
## 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
```



Imagine a criminal appeals court consisting of five judges; let's c
A, B, C, D, and E. The judges meet regularly to vote (independently, of cours
fate of prisoners who have petitioned for a review of their convictions. The
each of the court's deliberations is determined by a simple majority; for a p
to be granted or denied a new trial requires three or more votes. Based on l
record keeping, it is known that A votes correctly 95% of the time; i.e., when
to either uphold or to reverse the original conviction, he is wrong only 5% of
Similarly, B, C, D, and E vote correctly 95%, 90%, 90%, and 80% of the time
are, of course, two different ways a judge can make a mistake. The judge may
conviction, with new evidence later showing that the petitioner was in fact inno
the judge may vote to reverse a conviction when in fact the petitioner is actual
as determined by the result of a second conviction at the new trial.) Suppose
to calculate the probability that the court, as an entity, makes an incorrect de

Court decision problem

```r
s = c('correct', 'incorrect') # 1 correct decision, 0 incorrect decision

# Simulating the decision of judge-A once
sample(s, size = 1, prob = c(0.95, 0.05))
```

```
## [1] "correct"
```

```r
# Simulating the decision of judge-A 100,000 times
nsimulations = 1e5
simulatedData_A = sample(s, size = nsimulations, replace = TRUE, prob = c(0.95, 0.05))
simulatedData_B = sample(s, size = nsimulations, replace = TRUE, prob = c(0.95, 0.05))
simulatedData_C = sample(s, size = nsimulations, replace = TRUE, prob = c(0.9, 0.1))
#print(simulatedData)
```

```
# According to my simulated data, fraction of times judge-A would make a correct decision
mean(simulatedData_A == 'correct')
```

```
## [1] 0.94928
```

```
mean(simulatedData_B == 'correct')
```

```
## [1] 0.94987
```

```
mean(simulatedData_C == 'correct')
```

```
## [1] 0.90067
```

```

# Correct decision probabilities of all judges
judge_A_correct = 0.95
judge_B_correct = 0.95
judge_C_correct = 0.9
judge_D_correct = 0.9
judge_E_correct = 0.8

# Matrix of decision probabilities of all judges
p = matrix(data = c(judge_A_correct, judge_B_correct, judge_C_correct, judge_D_correct, judge_E_correct

# User-defined function to simulated the judges' decision making process once
courtResult = function(){
  result = character(ncol(p))
  for(j in c(1:ncol(p))){
    result[j] = sample(s, size = 1, replace = TRUE, prob = p[, j])
  }
  return(result)
}

# Simulate the judges' decision making process 100,000 times
nsimulations = 1e5
simulatedData = replicate(nsimulations, courtResult())

# User-defined function to check if the court decision is incorrect
checkEvent = function(data){
  return(sum(data == 'incorrect') >= (ncol(p)+1)/2)
}

# Probability that the court makes an incorrect decision
mean(apply(simulatedData, 2, checkEvent))
```

Simulated the 5 judges decision making using a modular code

```
## [1] 0.00715
```

```

# Correct decision probabilities of all judges
judge_A_correct = 0.95
judge_B_correct = 0.95
judge_C_correct = 0.9
judge_D_correct = 0.9
judge_E_correct = 0.8
```

```r
# Matrix of decision probabilities of all judges
p = matrix(data = c(judge_A_correct, judge_B_correct, judge_C_correct, judge_D_correct, judge_E_correct

# User-defined function to simulated the judges' decision making process once
courtResult = function(){
  result = character(ncol(p))
  for(j in c(1:ncol(p))){
    result[j] = sample(s, size = 1, replace = TRUE, prob = p[, j])
  }
  return(result)
}

# Simulate the judges' decision making process 100,000 times
nsimulations = 1e5
simulatedData = replicate(nsimulations, courtResult())

# User-defined function to check if the court decision is incorrect
checkEvent = function(data){
  return(sum(data == 'incorrect') >= (ncol(p)+1)/2)
}

# Probability that the court makes an incorrect decision
mean(apply(simulatedData, 2, checkEvent))
```

Simulated the 5 judges decision making using a modular code

```
## [1] 0.00693
```

```r
# Correct decision probabilities of all judges
judge_A_correct = 0.95
judge_B_correct = 0.95
judge_C_correct = 0.9
judge_D_correct = 0.9
judge_E_correct = 0.8

# Matrix of decision probabilities of all judges
p = matrix(data = c(judge_A_correct, judge_B_correct, judge_C_correct, judge_D_correct, judge_E_correct

# User-defined function to simulated the judges' decision making process once
courtResult = function(){
  result = character(ncol(p))
  for(j in c(1:ncol(p))){
    result[j] = sample(s, size = 1, replace = TRUE, prob = p[, j])
  }
  return(result)
}

# Simulate the judges' decision making process 100,000 times
nsimulations = 1e5
simulatedData = replicate(nsimulations, courtResult())

# User-defined function to check if the court decision is incorrect
checkEvent = function(data){
  return(sum(data == 'incorrect') >= (ncol(p)+1)/2)
}

# Probability that the court makes an incorrect decision
mean(apply(simulatedData, 2, checkEvent))
```

Simulated the 5 judges decision making using a modular code

```
## [1] 0.00738
```

```r

# Correct decision probabilities of all judges
judge_A_correct = 0.95
judge_B_correct = 0.95
judge_C_correct = 0.9
judge_D_correct = 0.9
judge_E_correct = 0.8

# Matrix of decision probabilities of all judges
p = matrix(data = c(judge_A_correct, judge_B_correct, judge_C_correct, judge_D_correct, judge_E_correct

# User-defined function to simulated the judges' decision making process once
courtResult = function(){
  result = character(ncol(p))
  for(j in c(1:ncol(p))){
    result[j] = sample(s, size = 1, replace = TRUE, prob = p[, j])
  }
  result[5] = result[1]
  return(result)
}

# Simulate the judges' decision making process 100,000 times
nsimulations = 1e5
simulatedData = replicate(nsimulations, courtResult())

# User-defined function to check if the court decision is incorrect
checkEvent = function(data){
  return(sum(data == 'incorrect') >= (ncol(p)+1)/2)
}

# Probability that the court makes an incorrect decision
mean(apply(simulatedData, 2, checkEvent))
```

Simulated the 5 judges decision making when judge-E replicates what judge-A does

```
## [1] 0.01177
```