Class 2:
Intro to R

API-201
Quantitative Analysis and Empirical Methods I

Profs. Goel and Taylor
Harvard Kennedy School

Agenda

1. Introduce core R commands
2. Sampling and simulation in R
3. Double voting redux

Class objectives

- Introduce / review core R commands
- Introduce the conceptual underpinnings of simulation
- Use simulation to understand the birthday paradox and examine claims of double voting

Fundamentals

Syntax	What it does	Example
`<-`	Assigns a value to a variable	`age <- 25`

`c()`	Creates vector of values and/or concatenate multiple vectors	`dogs <- c(‘wolfie’, ‘bella’)`
		`cats <- c(‘blue’, ‘elmo’)`
		`pets <- c(dogs, cats)`

`+ - * /`	Performs arithmetic	`(2+3)*5/4`
`TRUE, FALSE`	Special values to represent true and false statements, which we call *Boolean* values.	`my_var <- c(TRUE, FALSE)`

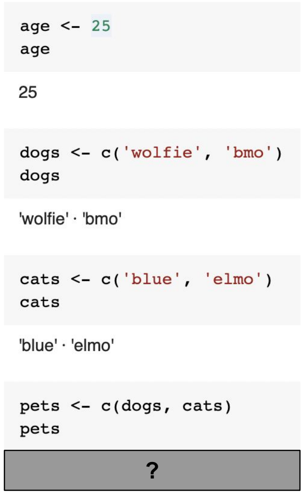
`==, <=, >=`	Compares expressions and return either TRUE or FALSE	`2 == 3`
		`3 >= 1`

`seq(m,n)` *or* `m:n`	Returns a vector of whole numbers from `m` to `n`	`seq(5,10).` Can also use the syntax `m:n`, like `5:10`, to produce the same result

`head()`	Returns first few entries of a vector or data frame	`long_list <- seq(1,100)`
		`head(long_list)`

`length()`	Returns the length of a vector	`long_list <- seq(1,100)`
		`length(long_list)`

Fundamentals



```
age <- 25
age
```

```
25
```

```
dogs <- c('wolfie', 'bmo')
dogs
```

```
'wolfie' · 'bmo'
```

```
cats <- c('blue', 'elmo')
cats
```

```
'blue' · 'elmo'
```

```
pets <- c(dogs, cats)
pets
```

?

Fundamentals

```
(2+3)*5/4
```

6.25

```
my_var <- c(TRUE, FALSE)
my_var
```

TRUE · FALSE

```
2 == 3
```

?

```
3 >= 1
```

?

Fundamentals

```
seq(5,10)
```

?

```
5:10
```

?

```
long_list <- seq(1,100)
head(long_list)
```

?

```
length(long_list)
```

?

Adding and averaging

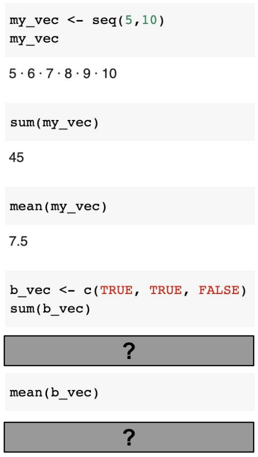
Syntax	What it does	Example

`sum()`	Returns the sum of numbers in a vector. For TRUE / FALSE vectors, `TRUE` values are interpreted as 1 and `FALSE` values are interpreted as 0.	`my_vec <- seq(5,10)`
		`sum(my_vec)`
		
		`b_vec <- c(TRUE, TRUE, FALSE)`
		`sum(b_vec)`

`mean()`	Returns the mean of numbers in a vector. For TRUE / FALSE vectors, `TRUE` values are interpreted as 1 and `FALSE` values are interpreted as 0.	`my_vec <- seq(5,10)`
		`mean(my_vec)`
		
		`b_vec <- c(TRUE, TRUE, FALSE)`
		`mean(b_vec)`

We often use `mean()` to compute the proportion of values in a `TRUE / FALSE` vector that are `TRUE`.

Adding and averaging



```
my_vec <- seq(5,10)
my_vec
```

```
5 · 6 · 7 · 8 · 9 · 10
```

```
sum(my_vec)
```

```
45
```

```
mean(my_vec)
```

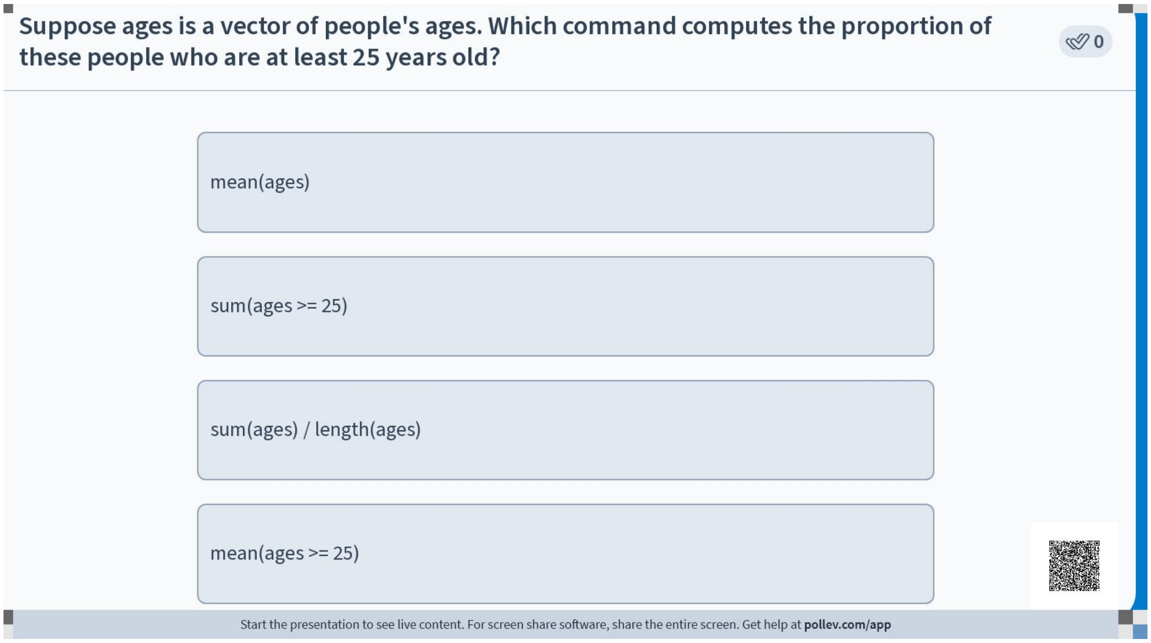
```
7.5
```

```
b_vec <- c(TRUE, TRUE, FALSE)
sum(b_vec)
```

?

```
mean(b_vec)
```

?



Suppose ages is a vector of people's ages. Which command computes the proportion of these people who are at least 25 years old?

- mean(ages)
- sum(ages >= 25)
- sum(ages) / length(ages)
- mean(ages >= 25)

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

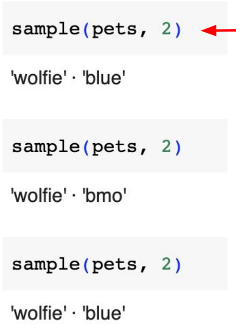
Simulation

Simulation is a powerful tool for understanding complex patterns. It typically proceeds in three steps.

1. **Generate** data based on some underlying process — for example, birthdays, to assess claims of double voting.
2. **Compute** some quantity of interest — for example, number of coincidental birthday matches.
3. **Repeat** steps 1 and 2 many times to understand the *distribution* of outcomes — for example, the *expected* number of birthday matches.

Sampling

Given a vector of *outcomes*, **`sample()`** returns random draws from the outcome vector.



```
sample(pets, 2)
```

'wolfie' · 'blue'

```
sample(pets, 2)
```

'wolfie' · 'bmo'

```
sample(pets, 2)
```

'wolfie' · 'blue'

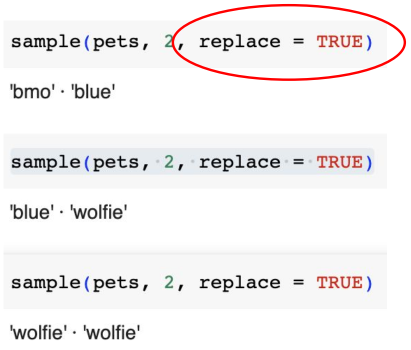
Number of draws

By default, `sample()` will not draw the same outcome twice.

This is called sampling **without replacement.**

Sampling

Given a vector of *outcomes*, **`sample()`** returns random draws from the outcome vector.



```
sample(pets, 2, replace = TRUE)
```

'bmo' · 'blue'

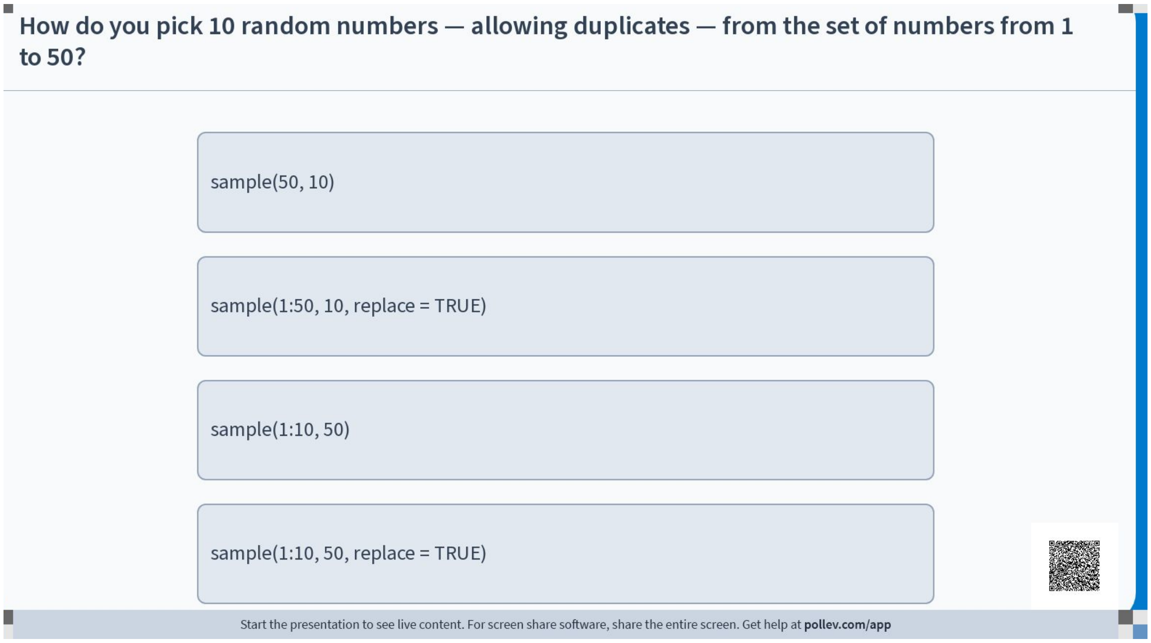
```
sample(pets, 2, replace = TRUE)
```

'blue' · 'wolfie'

```
sample(pets, 2, replace = TRUE)
```

'wolfie' · 'wolfie'

We can sample **with replacement** by setting `replace = TRUE`



How do you pick 10 random numbers — allowing duplicates — from the set of numbers from 1 to 50?

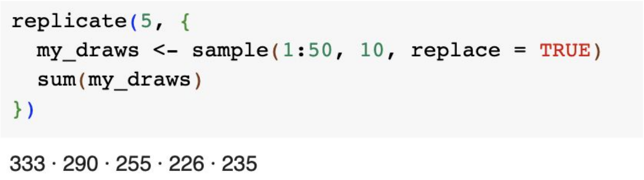
- sample(50, 10)
- sample(1:50, 10, replace = TRUE)
- sample(1:10, 50)
- sample(1:10, 50, replace = TRUE)

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

`replicate()`

Replicate lets us repeat the same line of code multiple times, returning the results of each iteration as a vector.

Number of replications



```
replicate(5, {
  my_draws <- sample(1:50, 10, replace = TRUE)
  sum(my_draws)
})
```

```
333 · 290 · 255 · 226 · 235
```

Key ideas

Much of our use of R can be grouped into three areas:

- **Fundamentals**. Arithmetic, storing values, creating and inspecting vectors.
- **Simulation**. Generating random data through sampling and replication
- **Data manipulation**. The five verbs of data manipulation plus the grouping operation. [Next two lectures.]

We use about **30 R commands** in this course.
[And we've already introduced many of them!]

Proficiency with this alphabet allows you to carry out complex data analysis.

Putting it all together

Now we'll use what we learned to help understand the birthday paradox and assess claims of double voting.

`https://bit.ly/API201-R`