Mini-lecture

DPSS - 2022 Summer program

Arthur Cheib

August 10, 2022

THE UNIVERSITY OF CHICAGO
HARRIS SCHOOL OF PUBLIC POLICY

Today's agenda

1

Reshaping datasets

- pivot_wider
- pivot_longer

3

Basic control flow

- ifelse statement

2

Conditional columns

- case_when

4

Repeating tasks

- for loop

How can we **reshape** our datasets?

USE CASE

There are two functions in the tidyverse package that can help us with this:

- pivot_wider(): from long to wide format.
- pivot_longer(): from wide to long format.

The pivot_longer and pivot_wider functions

Suppose we have a dataset like this:

Country	year	gdp
United States	2020	22
United States	2021	23
Brazil	2020	2.5
Brazil	2021	3

The pivot_longer and pivot_wider functions

Country	year	gdp
United States	2020	22
United States	2021	23
Brazil	2020	2.5
Brazil	2021	3

What if I wanted the dataset to look like this?

Country	2020	2021

The pivot_longer and pivot_wider functions

Country	year	gdp
United States	2020	22
United States	2021	23
Brazil	2020	2.5
Brazil	2021	3

pivot_wider(names_from = 'year',
values_from = 'gdp')

Country	2020	2021
United States	22	23
Brazil	2.5	3

The pivot_longer and pivot_wider functions

Country	year	gdp
United States	2020	22
United States	2021	23
Brazil	2020	2.5
Brazil	2021	3

Country	2020	2021
United States	22	23
Brazil	2.5	3

What if I wanted to go back to the longer format?

The pivot_longer and pivot_wider functions

Country	year	gdp
United States	2020	22
United States	2021	23
Brazil	2020	2.5
Brazil	2021	3

Country	2020	2021
United States	22	23
Brazil	2.5	3

pivot_longer(cols = c('2020', '2021'),
names_to = 'year',
values_to = 'gdp')

The pivot_longer and pivot_wider functions

Country	year	gdp
United States	2020	22
United States	2021	23
Brazil	2020	2.5
Brazil	2021	3

pivot_wider(names_from = 'year',
values_from = 'gdp')

Country	2020	2021
United States	22	23
Brazil	2.5	3

pivot_longer(cols = c('2020', '2021'),
names_to = 'year',
values_to = 'gdp')

Lets code! <>

RStudio®

The **case_when** function

USE CASE

- When working with datasets, we often want to modify our data based on different conditions (sometimes one condition, sometimes many).

- Normally, we do that using the **mutate** function. Because we want to have a new column with our new values (created based on our conditions)!

The **case_when** function

Country	year	gdp	gdp_classification
United States	2020	22	
United States	2021	23	
Brazil	2020	2.5	
Brazil	2021	3	

Suppose we would like to create a column that indicates if a country is a '**rich**' or a '**development**' one based on their **GDP**

The **case_when** function

You have the **GDP** information

Country	year	gdp	gdp_classification
United States	2020	22	
United States	2021	23	
Brazil	2020	2.5	
Brazil	2021	3	

You want to create a **new column** based on this GDP data

The **case_when** function

You have the **GDP** information

Country	year	gdp	gdp_classification
United States	2020	22	
United States	2021	23	
Brazil	2020	2.5	
Brazil	2021	3	

You want to create a **new column** based on this GDP data

function

the value for R to output if the condition is TRUE

```
case_when(condition ~ output-value)
```

a condition for R to evaluate as TRUE

The **case_when** function

You have the **GDP** information

Country	year	gdp	gdp_classification
United States	2020	22	
United States	2021	23	
Brazil	2020	2.5	
Brazil	2021	3	

You want to create a **new column** based on this GDP data

the value for R to output if the condition is TRUE

```
case_when (gdp > 10 ~ 'rich')
```

a condition for R to evaluate as TRUE

The **case_when** function

Country	year	gdp	gdp_classification
United States	2020	22	
United States	2021	23	
Brazil	2020	2.5	
Brazil	2021	3	

And what about the **output** for every other value that is no higher than 10? Basically, what about everything else?

The **case_when** function

Country	year	gdp	gdp_classification
United States	2020	22	
United States	2021	23	
Brazil	2020	2.5	
Brazil	2021	3	

And what about the **output** for everything else?

```
case_when (gdp > 10 ~ 'rich',
           TRUE ~ 'development')
```

this it the formula for **everything else**

The case_when function

Final result!

Country	year	gdp	gdp_classification
United States	2020	22	rich
United States	2021	23	rich
Brazil	2020	2.5	development
Brazil	2021	3	development

```
mutate (gdp_classification = case_when (gdp > 10 ~ 'rich',
                                        TRUE ~ 'development')
```

Lets code! <>

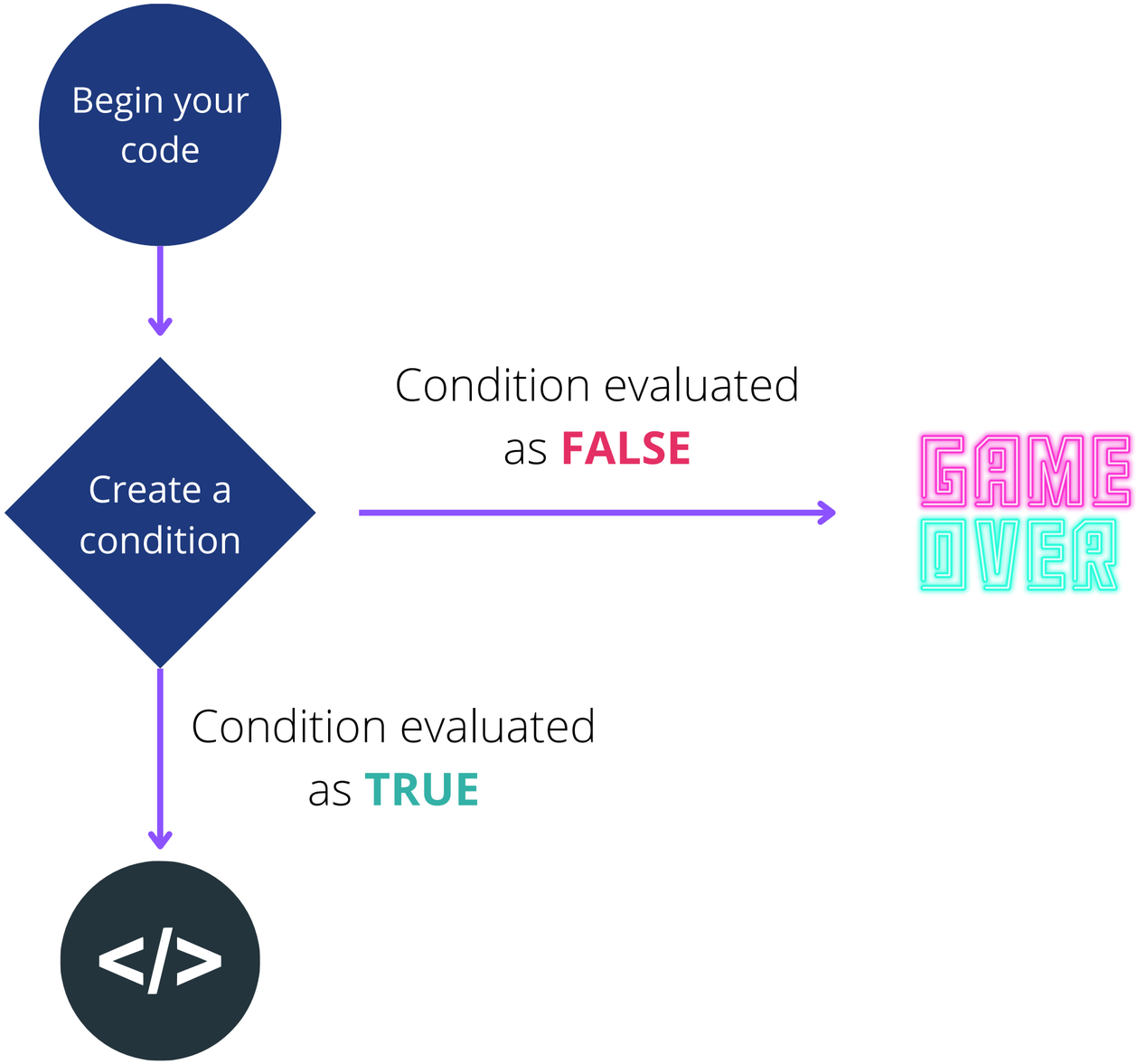
RStudio®

The if-else statement

USE CASE

- Know as a control statement, the **if** and **else** statement evaluates a condition and then indicates R what is next in case the condition is **TRUE** or **FALSE**

The if-else statement



The if-else statement

What is the basic structure of an **if-else** statement?

The condition we want R to check for us

```
if (condition) {

  body

}
```

The task we want R to perform in case the condition is TRUE

The if-else statement

Basic structure added of the the **else** statement?

```
if (condition) {

  body

} else {

  body II

}
```

The task we want R to perform in case the condition is FALSE

Lets code! <>

RStudio®

For loop in R

USE CASE

- A loop is a control statement that allows multiple executions of a set of lines of code. The word ‘looping’ means cycling or iterating
- Normally, we want to use a **for loop** when we need R to perform a task multiple times.
- **Golden rule:** *"if you repeated the same line 3x, you might need a **for loop**"*

For loop in R

Task a

Needs to be done 9x

Option 1

Task a	Task a	Task a
Task a	Task a	Task a
Task a	Task a	Task a

Option 2

for each time I want to perform the task:	peform task a

For loop in R

Task a

Option 1

Task a	Task a	Task a
Task a	Task a	Task a
Task a	Task a	Task a

Option 2

for each time I want to perform the task:	peform task a

For loop in R

What is the basic structure of a **for loop**?

The element
or
the smallest element

The object
or
the length of an object
or
how many times you want the task to be performed

```
for (variable in object) {

  body

}
```

the task we want R to repeatedly perform

For loop in R

The most common version you will find: the **'i'**

```
for (i in my_vec) {

  print(i)

}
```

For loop in R

Suppose the **task** was:

a) *print - manually - every value that exists inside the vector named 'my_vec'*

```
## Vector (samples of Harris courses)
Line 1 ← my_vec <- c('Program Evaluation', 'Advanced Statistics', 'Analytica Politics',
            'Microeconomics', 'Cost & Benefit Analysis', 'Data and Programming in R',
            'Data Visualization', 'The Art of Political campaining')

## How it would be done manually?
Line 2 ← print(my_vec[1])
## [1] "Program Evaluation"
Line 3 ← print(my_vec[2])
## [1] "Advanced Statistics"
Line 4 ← print(my_vec[3])
## [1] "Analytica Politics"
Line 5 ← print(my_vec[4])
## [1] "Microeconomics"
Line 6 ← print(my_vec[5])
## [1] "Cost & Benefit Analysis"
Line 7 ← print(my_vec[6])
## [1] "Data and Programming in R"
Line 8 ← print(my_vec[7])
## [1] "Data Visualization"
Line 9 ← print(my_vec[8])
## [1] "The Art of Political campaining"
```

For loop in R

Suppose the **task** was:

b) *how to do it using a* ***for loop?***

```
## How we can do it using a 'for loop'?
for (course in my_vec) {

  print(course)

}
## [1] "Program Evaluation"
## [1] "Advanced Statistics"
## [1] "Analytica Politics"
## [1] "Microeconomics"
## [1] "Cost & Benefit Analysis"
## [1] "Data and Programming in R"
## [1] "Data Visualization"
## [1] "The Art of Political campaining"
```

Line 2

Line 3

For loop in R

Ok, but...

Why is R understanding the mere word **course** as being every element inside the vector?

Lets code! <>

RStudio®

Thank you!

Mini-lecture

DPSS - 2022 Summer program