Multiple dispatch

INTERMEDIATE JULIA

Anthony Markham
Quantitative Developer

datacamp

Multiple dispatch recap

- Multiple dispatch allows us to run a different method based on the type of argument passed into a function.

```
function add_values(x, y)
    x + y
end
add_values(1, 2)  # 3
```

```
add_values("A", "B")
```

```
ERROR: MethodError: no method matching +(::String, ::String)
```

```
function add_values(x::String, y::String)
    x * y
end

add_values("A", "B")  # AB
```

Anonymous functions

- Unlike typical functions, an anonymous function has no name.
- Used to create quick functions which are discarded after use.

```
(x -> x^2 + 3)(2)
```

```
7
```

Complex anonymous functions

- Anonymous functions can have applications outside of simple function evaluation.
- map() will apply each value in a collection to a function.

```
map(x -> 2*x + x^2 + 1, [1, 2, 3])
```

```
3-element Vector{Int64}:
4 9 16
```

```
map((x, y) -> 2*x + x^2 + 1 + y, [1, 2, 3], [1, 1, 1])
```

```
3-element Vector{Int64}:
5 10 17
```

Filtering DataFrames

- The `filter` function can be used to filter data structures with conditions.
- We can use an anonymous function with `filter` which helps us filter for a certain value.

```
filter!("Date" => n -> n == "21/01/2022", stock_data)
```

```
 Row | Date        Open     High     Low      Close     Adj Close  Volume
     | String15    Float64  Float64  Float64  Float64?  Float64    Int64
<----|-----------------------------------------------------------------------
   1 | 21/01/2022   164.42   166.33    162.3    162.41    161.473  122848900
```

Let's practice!

INTERMEDIATE JULIA

Importing Functions from Python and R

INTERMEDIATE JULIA

Anthony Markham
Quantitative Developer

datacamp

Package overview

- We might want to use packages from Python or R in your Julia programs.
- There are many reasons why we might want to do this:
  - familiarity with a certain package
  - niche package only available for Python or R
  - consistency with other code
- Two Julia packages give this functionality - **PythonCall** and **RCall**.

```
using Pkg
Pkg.add("PythonCall")
Pkg.add("RCall")
```

- Note that the reverse is possible - we can also use Julia code in Python and R.

Importing python packages

- Use `PythonCall` to integrate Python functions into our Julia code.
- Any Python package can be used, as long it is also installed in Python.

```
using PythonCall
```

- To load a package from Python, use `pyimport` .

```
pytime = pyimport("time")
```

- Many advanced functions exist in PythonCall that we will not cover in this course.

Calling python functions

- We can call any function from the base time package.

```
using PythonCall
pytime = pyimport("time")
println(pytime.ctime())
```

```
Sun Jan 22 12:16:28 2023
```

- We can combine multiple functions including Python formatting.

```
println(pytime.strftime("%m-%Y", pytime.localtime()))
```

```
01-2023
```

Importing R libraries

- Use `RCall` to integrate R functions into our Julia code.
- An R installation is required before installing the `RCall` package.
- Any R package can be used, as long as that package is also installed in R.

```
using RCall
```

- To load a package in R, use `@rimport` .

```
@rimport base
@rimport ggplot2
```

- Many advanced functions exist in RCall that we will not cover in this course.

Calling R functions

- We will be calling functions from the base R environment in this example.
- We can provide an alias when importing a package.

```
using RCall
@rimport base as r_base
```

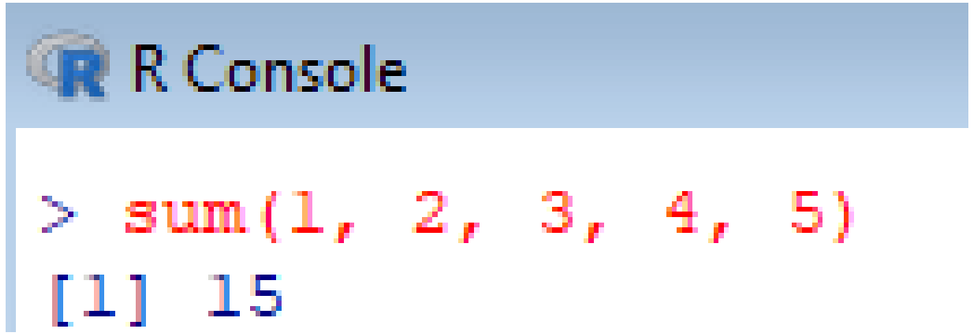
```
r_base.sum([1, 2, 3, 4, 5])
```

```
RObject{IntSxp}
[1] 15
```

R sum output

- We can quickly compare the output from Julia to the output that R gives.

```
RObject{IntSxp}
[1] 15
```

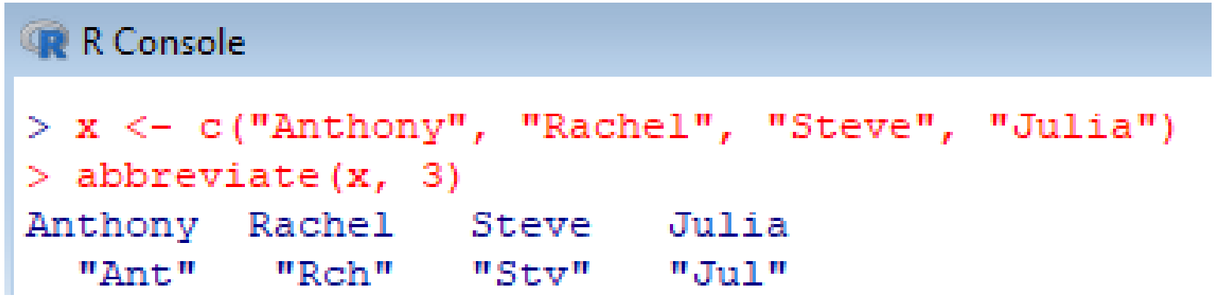


Calling further R functions

- We will now try the abbreviate function, also in the base package.

```
r_base.abbreviate(["Anthony", "Rachel", "Steve", "Julia"], 3)
```

```
RObject{StrSxp}
Anthony  Rachel   Steve   Julia
  "Ant"   "Rch"   "Stv"   "Jul"
```



Let's practice!

INTERMEDIATE JULIA

Cleaning Data

INTERMEDIATE JULIA

Anthony Markham
Quantitative Developer

Show column information

- Column names are the first things that you will look at in a dataset.
- Generally speaking, column names should be legible and concise.

```
println(first(stock_data))
```

```
DataFrameRow
 Row | Date        Open     High     Low      Close    Adj Close  Volume
     | String15    Float64  Float64  Float64  Float64  Float64    Int64
<----|-----------------------------------------------------------------
   1 | 21/01/2022   164.42   166.33    162.3   162.41    161.473  122848900
```

Rename a column

- We might want to rename a column if we feel a different label is more appropriate.

```
 Row | Date        Open     High     Low      Close    Adj Close  Volume
     | String15    Float64  Float64  Float64  Float64  Float64    Int64
<----|-----------------------------------------------------------------
   1 | 21/01/2022   164.42   166.33    162.3   162.41    161.473  122848900
```

- In this case, `Adj Close` has a space, which we want to avoid.

```
rename!(stock_data, Dict(:Adj Close => :Adj_Close))
```

```
 Row | Date        Open     High     Low      Close    Adj_Close  Volume
     | String15    Float64  Float64  Float64  Float64  Float64    Int64
<----|-----------------------------------------------------------------
   1 | 21/01/2022   164.42   166.33    162.3   162.41    161.473  122848900
```

Describe and find missing data

- Missing values are a common source of inconsistency in data.
- There can be various reasons for missing data:
  - measurement errors
  - transcription errors
  - intentionally missing
- The describe method can be used to quickly find missing data in a DataFrame.

```
println(describe(stock_data))
```

Describe missing data

- describe() gives us a general overview of our entire DataFrame.

```
7×7 DataFrame
 Row | variable    mean        min         median     max         nmissing  eltype
     | Symbol      Union       Any          Union     Any         Int64     DataType
<-------------------------------------------------------------------------------------
   1 | Date                    1/02/2022              9/12/2022           0  String15
   2 | Open        152.613     126.01      151.19     178.55              0  Float64
   3 | High        154.721     127.77      153.72     179.61              0  Float64
   4 | Low         150.529     124.17      149.34     176.7               0  Float64
   5 | Close       152.698     125.02      151.21     178.96              4  Union
   6 | Adj_Close   152.296     125.02      151.07     178.154             0  Float64
   7 | Volume      8.70851e7   35195900    8.22912e7  182602000           0  Int64
```

Remove missing data

- We can drop the missing rows using `dropmissing` .

```
println(nrow(stock_data))
```

```
252
```

```
dropmissing!(stock_data, :"Close")
```

- We can confirm that we dropped 4 rows using nrow() again.

```
println(nrow(stock_data))
```

```
248
```

Replace missing data

- Simply removing rows with missing values is often not an acceptable approach.
- Depending on the data, we can replace missing values with a substitute.

```
replace!(stock_data[!, "Close"], missing => 130)
println(stock_data[[202, 227, 235, 240], :])
```

```
 Row | Date        Open     High     Low      Close     Adj Close  Volume
     | String15    Float64  Float64  Float64  Float64?  Float64    Int64
<----|-------------------------------------------------------------------
   1 | 8/11/2022    140.41   141.43   137.49    130       139.5    89908500
   2 | 14/12/2022   145.35   146.66   141.16    130       143.21   82291200
   3 | 25/12/2022   130.92   132.42   129.64    130       131.86   63814900
   4 | 3/01/2023    130.28   130.9    124.17    130       125.07  112117500
```

- Choosing an arbitrary value to replace missing values with is rarely correct.

Let's practice!

INTERMEDIATE JULIA

Congratulations!

INTERMEDIATE JULIA

Anthony Markham
Quantitative Developer

Julia is growing!

Course recap

Loops and Ranges

- For loops
- While loops
- Ranges

Data Structures

- Tuples
- Dictionaries
- Structs

Advanced Functions

- Execution time measurement
- Function argument types
- Writing your own functions

DataFrame Operations

- Anonymous functions
- DataFrame filtering/cleaning
- Using Python/R in Julia

What's next?

- Keep practicing! Like anything, programming in Julia takes practice.
- Learn a new package, e.g:
  - Plots.jl
  - DataFrames.jl
  - Distributions.jl
- DataCamp courses on Julia in the future will include:
  - Data Visualization in Julia
  - Data Manipulation in Julia

Thank you!

INTERMEDIATE JULIA