pandas (R)osetta: Tidyverse R

Intro to R and Tidyverse for Pandas users and vice versa.

Lisa Malins

Contents

This is a demonstration of basic data wrangling operations in Tidyverse R.

Sister notebooks demonstrate the exact same operations in base R and the Pandas library in Python, just like the Rosetta Stone.

I/O

Create dataframe from scratch

```r
suppressPackageStartupMessages(library("dplyr"))
tbl <- tibble(
  letter = c("a", "b", "c", "d", "e"),
  number = c(1:5),
  fruit = c("apple", "banana", "coconut", "date", "elderberry"),
  vegetable = c("arugula", "beet", "carrot", "daikon", "eggplant"),
  name = c("Alice", "Bob", "Carol", "Dan", "Eve")
)
tbl
```

```
## # A tibble: 5 x 5
##   letter number fruit      vegetable name
##   <chr>   <int> <chr>      <chr>     <chr>
## 1 a           1 apple      arugula   Alice
## 2 b           2 banana     beet      Bob
## 3 c           3 coconut    carrot    Carol
## 4 d           4 date       daikon    Dan
## 5 e           5 elderberry eggplant  Eve
```

Write

```r
library("readr")
write_csv(tbl, "data/tidy_letters.csv")
write_tsv(tbl, "data/tidy_letters.tsv")
```

Read

```r
library("readr")
read_csv("data/tidy_letters.csv")
```

```
## Rows: 5 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): letter, fruit, vegetable, name
## dbl (1): number
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 5 x 5
##   letter number fruit      vegetable name
##   <chr>   <dbl> <chr>      <chr>     <chr>
## 1 a           1 apple      arugula   Alice
## 2 b           2 banana     beet      Bob
## 3 c           3 coconut    carrot    Carol
## 4 d           4 date       daikon    Dan
## 5 e           5 elderberry eggplant  Eve
```

```r
read_tsv("data/tidy_letters.tsv")
```

```
## Rows: 5 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (4): letter, fruit, vegetable, name
## dbl (1): number
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 5 x 5
##   letter number fruit      vegetable name
##   <chr>   <dbl> <chr>      <chr>     <chr>
## 1 a           1 apple      arugula   Alice
## 2 b           2 banana     beet      Bob
## 3 c           3 coconut    carrot    Carol
## 4 d           4 date       daikon    Dan
## 5 e           5 elderberry eggplant  Eve
```

Accessing data

Note for Pandas users:

- Both R and Pandas have the same convention of `[row, column]` for retrieving a cell from a dataframe.
- However, if only one number is specified with no comma, in pandas a row is returned, but in R a column is returned.
- Also, remember that **R has 1-based indexing** while Python has 0-based indexing.
- Thus,
  - In pandas: `df.iloc[1]` returns the second row
  - In R: `df[1]` returns the first column

Select cell

To **select a single cell**, **use double brackets** with the row number, followed by the column number or column name.

```
# Get "banana" cell value
tbl[[2, 3]]
tbl[[2, "fruit"]]
```

```
## [1] "banana"
## [1] "banana"
```

Cells can also be selected with single brackets. **Tibbles** (the Tidyverse version of dataframes) actually have different behavior for single brackets than dataframes in base R.

- When using single brackets to select a single cell of a dataframe, the value itself is returned.
- When using single brackets to select a single cell of a tibble, a 1x1 tibble is returned. The behavior of tibbles is more consistent with R behavior generally, because single brackets usually return an object of the same type.

```
# Get "banana" cell as 1x1 tibble
tbl[2, 3]
tbl[2, "fruit"]
```

```
## # A tibble: 1 x 1
##   fruit
##   <chr>
## 1 banana
## # A tibble: 1 x 1
##   fruit
##   <chr>
## 1 banana
```

The bracket notation is the most common method of selecting a single cell, even in Tidyverse workflows.

That being said, there are `dplyr` functions that do the same thing, and can be useful for chaining when a single cell is the end goal, but other `dplyr` operations need to be done first.

- `select` returns one or more columns (as a dataframe)

- `pull` returns a single column as a vector
- `slice` returns a row

```r
# Get "banana" cell value
tbl %>% slice(2) %>% pull("fruit")
tbl %>% slice(2) %>% pull(3)
```

```
## [1] "banana"
## [1] "banana"
```

```r
# Get "banana" cell as 1x1 tibble
tbl %>% select("fruit") %>% slice(2)
tbl %>% select(3) %>% slice(2)
```

```
## # A tibble: 1 x 1
##   fruit
##   <chr>
## 1 banana
## # A tibble: 1 x 1
##   fruit
##   <chr>
## 1 banana
```