Programming for Data Science

Tibbles in R

Marco Beccuti

Università degli Studi di Torino

Dipartimento di Informatica

November 2021

Tibbles in R

- Tibbles are a new implementation for data frames;
- They tweak some older behaviors to make life a little easier;
- They are implemented in the *tibble* package, part of the *tidyverse* package.

  $>$ *library*(*tidyverse*)

Creating Tibbles

- You can convert an older data frames into a tibbles one using *as_tibble* function:

```
> as_tibble(iris)
```

```
#> # A tibble: 150 × 5
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#>           <dbl>       <dbl>        <dbl>       <dbl>  <fctr>
#> 1           5.1         3.5          1.4         0.2  setosa
#> 2           4.9         3.0          1.4         0.2  setosa
#> 3           4.7         3.2          1.3         0.2  setosa
#> 4           4.6         3.1          1.5         0.2  setosa
#> 5           5.0         3.6          1.4         0.2  setosa
#> 6           5.4         3.9          1.7         0.4  setosa
#> # ... with 144 more rows
```

Creating Tibbles

- You can create a new tibble from individual vectors with *tibble()*;

  $> tibble(x = 1:5, y = 1, z = x^2 + y)$

```
#> # A tibble: 5 × 3
#>       x     y     z
#>   <int> <dbl> <dbl>
#> 1     1     1     2
#> 2     2     1     5
#> 3     3     1    10
#> 4     4     1    17
#> 5     5     1    26
```

- *tibble()* will automatically recycle inputs of length 1.

Creating Tibbles

- Another way to create a tibble is with *tribble()*;
- Using *tribble()* column headings are defined by formulas, and entries are separated by commas.

$> tribble(\sim x, \sim y, \sim z, "a", 2, 3.6, "b", 1, 8.5)$

```
#> # A tibble: 2 × 3
#>       x     y     z
#>   <chr> <dbl> <dbl>
#> 1     a     2   3.6
#> 2     b     1   8.5
```

Tibbles VS data.frame

- There are two main differences in the usage of a *tibble* versus a classic *data.frame*: printing and subsetting.
- A refined print method shows only the first 10 rows, and all the columns that fit on screen.

```
> MyTibble = tibble(
                    a = lubridate :: now() + runif(1e3) * 86400,
                    b = lubridate :: today() + runif(1e3) * 30,
                    c = 1 : 1e3,
                    d = runif(1e3),
                    e = sample(letters, 1e3, replace = TRUE)
                  )
```

Tibbles VS data.frame

- Standard visualization of a tibble:

```
#> # A tibble: 1,000 × 5
#>                     a          b     c     d     e
#>                 <dttm>     <date> <int> <dbl> <chr>
#> 1 2016-10-10 17:14:14 2016-10-17     1 0.368     h
#> 2 2016-10-11 11:19:24 2016-10-22     2 0.612     n
#> 3 2016-10-11 05:43:03 2016-11-01     3 0.415     l
#> 4 2016-10-10 19:04:20 2016-10-31     4 0.212     x
#> 5 2016-10-10 15:28:37 2016-10-28     5 0.733     a
#> 6 2016-10-11 02:29:34 2016-10-24     6 0.460     v
#> # ... with 994 more rows
```

- you can explicitly *print()* the data frame controlling the number of rows and the width of the display:

  $> print(MyTibble, n = 10, width = Inf)$

- Using Rstudio you can exploit *view()*:

  $> view(MyTibble)$

Subsetting

- To pull out a single variable (column) of a tibble $ and [[]] operators can be used.

```
> df <- tibble(x = runif(5), y = rnorm(5))

> df$x
[1] 0.434 0.395 0.548 0.762 0.254

> df[["x"]]
[1] 0.434 0.395 0.548 0.762 0.254

> df[[1]]
[1] 0.434 0.395 0.548 0.762 0.254
```

- You can access elements of a tibble as data.frame.

```
> df[1, c(1, 3)]
[1] 0.434 0.876
```

Interacting with Older Code

- Some older functions do not work with tibbles;
- We can use *as.data.frame()* to turn a *tibble* back to a *data.frame*:

```
> class(as.data.frame(tb))
[1] "data.frame"
```