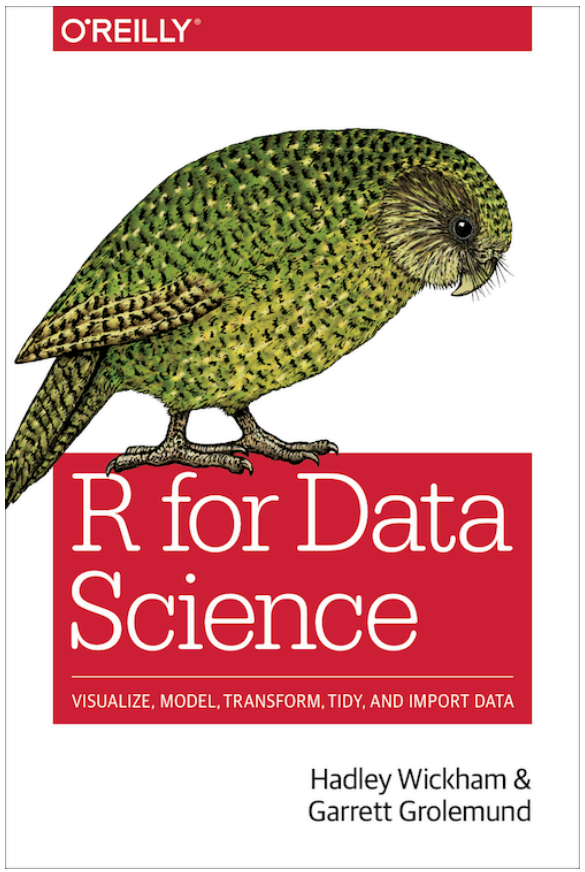
Video 18: introducing tidyverse and tibbles

Stats 102A

Miles Chen, PhD

The tidyverse

Resource: R For Data Science



Portions of this lecture are derived from the book. The book is Free to read: https://r4ds.had.co.nz/

The tidyverse

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures."

Core Packages that get loaded with `library(tidyverse)`:

- `ggplot2` (graphics)
- `tibble` (data frames with some tweaks)
- `tidyr` (making data tidy)
- `dplyr` (data manipulation)
- `readr` (importing data)
- `purrr` (functional programming)
- `stringr` (regular expressions)
- `forcats` (factors and categorical data)

Tibbles

The tibble

The tidyverse works with the "tibble" instead of the traditional `data.frame`. Tibbles are data frames, but they tweak some older behaviors to make life a little easier. R is an old language, and some things that were useful 10 or 20 years ago now get in your way. It's difficult to change base R without breaking existing code, so most innovation occurs in packages. The tibble package provides opinionated data frames that make working in the tidyverse a little easier.

Tibbles vs Data.Frames: Printing

Tibbles have a refined print method that shows only the first 10 rows, and all the columns that fit on screen. This makes it much easier to work with large data. In addition to its name, each column reports its type, a nice feature borrowed from str():

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

Tibbles vs Data.Frames: Printing

```
print(x)
```

```
# A tibble: 1,000 × 5
   a                   b                c      d e
   <dttm>              <date>       <int>  <dbl> <chr>
 1 2024-07-01 17:45:57 2024-07-23       1 0.791  a
 2 2024-07-01 10:00:41 2024-07-14       2 0.468  c
 3 2024-06-30 23:48:37 2024-07-01       3 0.0428 y
 4 2024-07-01 07:13:15 2024-07-10       4 0.352  g
 5 2024-06-30 23:55:02 2024-07-01       5 0.0472 h
 6 2024-07-01 15:56:08 2024-07-21       6 0.715  b
 7 2024-07-01 20:07:55 2024-07-26       7 0.890  q
 8 2024-07-01 10:06:07 2024-07-14       8 0.472  d
 9 2024-07-01 12:54:24 2024-07-17       9 0.588  z
10 2024-07-01 03:31:48 2024-07-05      10 0.198  d
# ℹ 990 more rows
```

Tibbles vs Data Frames: Subsetting

Using single square brackets [ ] on a tibble will *always* return a tibble (unless you explicity use the drop argument).

```
1 tb <- tibble(x = 1:3, y = 3:1)
2 tb[, 1] # subset to the first column. Remains a tibble.
```

```
# A tibble: 3 × 1
      x
  <int>
1     1
2     2
3     3
```

```
1 tb[, 1, drop = TRUE] # with drop = TRUE, it will simplify after sub
```

```
[1] 1 2 3
```

Tibbles vs Data Frames: Subsetting

In contrast, with a data frame, an operation like `df[ , 1]` will simplify and return a vector.

```
df <- data.frame(x = 1:3, y = 3:1)
df[, 1]
```

```
[1] 1 2 3
```

To extract the column as a vector from a tibble, you can use double square brackets `[[]]` or the dollar sign `$` as you do with data frames.

```
tb[[1]]
```

```
[1] 1 2 3
```

```
tb$x
```

```
[1] 1 2 3
```

Tibble creation

Creating a tibble is easy. You can create a tibble the same way you do a data frame, specifying the name of a column and the values that go in the column.

```
tib <- tibble(
  a = sample(5),
  b = letters[sample(5)],
  c = rnorm(5)
)
```

You can also take an existing data frame and feed it into a tibble.

```
mtcars_tib <- tibble(mtcars)
```

Tibble creation

Tibbles can also be created row-wise so that a person reading your code can easily see the values contained in the tibble without needing to print the tibble. This is achieved with the function `tribble`

```r
tib_r <- tribble(
  ~colA, ~colB,
  "a",    1,
  "b",    2,
  "c",    3
)
print(tib_r)
```

```
# A tibble: 3 × 2
  colA   colB
  <chr> <dbl>
1 a         1
2 b         2
3 c         3
```

Significant figures in tibbles

In order to fit as many columns as possible, tibbles have a tendency to display only one or two places after a decimal point.

If you prefer to have more digits shown, you can use the following option

```
1 tibble(col_a = 9.8765432, col_b = 1.1234567)
```

```
# A tibble: 1 × 2
  col_a col_b
  <dbl> <dbl>
1  9.88  1.12
```

```
1 options(pillar.sigfig = 5) # will display 5 significant figures
2 tibble(col_a = 9.8765432, col_b = 1.1234567)
```

```
# A tibble: 1 × 2
   col_a  col_b
   <dbl>  <dbl>
1 9.8765 1.1235
```

Printing more rows

```
1 mtcars_tib <- tibble(mtcars)
2 mtcars_tib
```

```
# A tibble: 32 × 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21        6 160     110  3.9  2.62  16.46     0     1     4     4
 2  21        6 160     110  3.9  2.875 17.02     0     1     4     4
 3  22.8      4 108      93  3.85 2.32  18.61     1     1     4     1
 4  21.4      6 258     110  3.08 3.215 19.44     1     0     3     1
 5  18.7      8 360     175  3.15 3.44  17.02     0     0     3     2
 6  18.1      6 225     105  2.76 3.46  20.22     1     0     3     1
 7  14.3      8 360     245  3.21 3.57  15.84     0     0     3     4
 8  24.4      4 146.7    62  3.69 3.19  20        1     0     4     2
 9  22.8      4 140.8    95  3.92 3.15  22.9      1     0     4     2
10  19.2      6 167.6   123  3.92 3.44  18.3      1     0     4     4
# ℹ 22 more rows
```

Printing more rows

```
1 print(mtcars_tib, n = 32)
```

```
# A tibble: 32 × 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21        6 160     110  3.9  2.62  16.46     0     1     4     4
 2  21        6 160     110  3.9  2.875 17.02     0     1     4     4
 3  22.8      4 108      93  3.85 2.32  18.61     1     1     4     1
 4  21.4      6 258     110  3.08 3.215 19.44     1     0     3     1
 5  18.7      8 360     175  3.15 3.44  17.02     0     0     3     2
 6  18.1      6 225     105  2.76 3.46  20.22     1     0     3     1
 7  14.3      8 360     245  3.21 3.57  15.84     0     0     3     4
 8  24.4      4 146.7    62  3.69 3.19  20        1     0     4     2
 9  22.8      4 140.8    95  3.92 3.15  22.9      1     0     4     2
10  19.2      6 167.6   123  3.92 3.44  18.3      1     0     4     4
11  17.8      6 167.6   123  3.92 3.44  18.9      1     0     4     4
12  16.4      8 275.8   180  3.07 4.07  17.4      0     0     3     3
13  17.3      8 275.8   180  3.07 3.73  17.6      0     0     3     3
14  15.2      8 275.8   180  3.07 3.78  18        0     0     3     3
15  10.4      8 472     205  2.93 5.25  17.98     0     0     3     4
16  10.4      8 460     215  3    5.424 17.82     0     0     3     4
17  14.7      8 440     230  3.23 5.345 17.42     0     0     3     4
18  32.4      4  78.7    66  4.08 2.2   19.47     1     1     4     1
19  30.4      4  75.7    52  4.93 1.615 18.52     1     1     4     2
```