Data wrangling II: Appending, joining, and reshaping data

EDH7916 | Spring 2020

Benjamin Skinner

```
## ---------------------------
## libraries
## ---------------------------

library(tidyverse)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

✔ ggplot2 3.2.1     ✔ purrr   0.3.3
✔ tibble  2.1.3     ✔ dplyr   0.8.3
✔ tidyr   1.0.2     ✔ stringr 1.4.0
✔ readr   1.3.1     ✔ forcats 0.4.0

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

```
## ---------------------------
## directory paths
## ---------------------------

## assume we're running this script from the ./scripts subdirectory
dat_dir <- file.path("..", "data")
sch_dir <- file.path(dat_dir, "sch_test")
bys_dir <- file.path(sch_dir, "by_school")
```

Appending data

```
## ---------------------------
## input
## ---------------------------

## data are CSV, so we use read_csv()
df_1 <- read_csv(file.path(bys_dir, "bend_gate_1980.csv"))
```

```
Parsed with column specification:
cols(
  school = col_character(),
  year = col_double(),
  math = col_double(),
  read = col_double(),
  science = col_double()
```

```
)
```

```
df_2 <- read_csv(file.path(bys_dir, "bend_gate_1981.csv"))
```

```
Parsed with column specification:
cols(
  school = col_character(),
  year = col_double(),
  math = col_double(),
  read = col_double(),
  science = col_double()
)
```

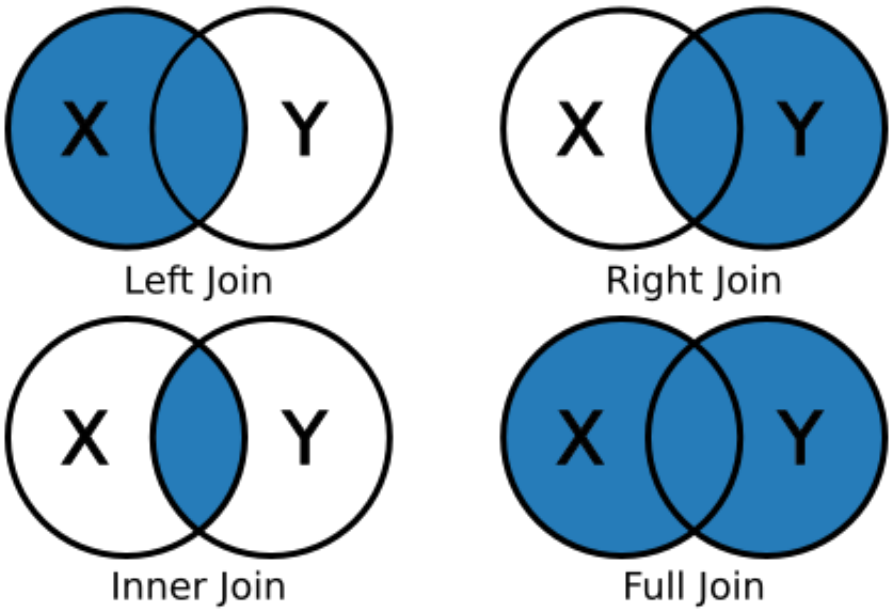
```
df_3 <- read_csv(file.path(bys_dir, "bend_gate_1982.csv"))
```

```
Parsed with column specification:
cols(
  school = col_character(),
  year = col_double(),
  math = col_double(),
  read = col_double(),
  science = col_double()
)
```

Joining data

While one can `merge` using base R, {dplyr} uses the SQL language of joins:

- `left_join(x, y)`: keep all x, drop unmatched y
- `right_join(x, y)`: keep all y, drop unmatched x
- `inner_join(x, y)`: keep only matching
- `full_join(x, y)`: keep everything



Since we want to join a smaller aggregated data frame to the original data frame, we'll use a `left_join()`. The join functions will try to guess the joining variable (and tell you what it picked) if you don't supply one,

but we'll specify one to be clear.

Reshaping data

```
## ---------------------------
## input
## ---------------------------

## data are CSV, so we use read_csv()
df <- read_csv(file.path(sch_dir, "all_schools.csv"))
```

```
Parsed with column specification:
cols(
  school = col_character(),
  year = col_double(),
  math = col_double(),
  read = col_double(),
  science = col_double()
)
```

Reshaping data is a common data wrangling task. Whether going from wide to long format or long to wide, this can be a painful process. Though you can reshape data frames using base R commands, the best way I know to reshape data in R is by using functions in the tidyr library.

To start, the data are wide in **test**. While this setup can be efficient for storage, it's not always the best for analysis or even just browsing. What we want is for the data to be long. Instead of each test having its own column, there should be one column for the test subject (**test**) and another column that gives the value (**score**). It should look like this:

```
# A tibble: 24 x 5
   school        year  math  read science
   <chr>        <dbl> <dbl> <dbl>   <dbl>
 1 Bend Gate     1980   515   281     808
 2 Bend Gate     1981   503   312     814
 3 Bend Gate     1982   514   316     816
 4 Bend Gate     1983   491   276     793
 5 Bend Gate     1984   502   310     788
 6 Bend Gate     1985   488   280     789
 7 East Heights  1980   501   318     782
 8 East Heights  1981   487   323     813
 9 East Heights  1982   496   294     818
10 East Heights  1983   497   306     795
# … with 14 more rows
```

```
## ---------------------------
## input
## ---------------------------

## data are CSV, so we use read_csv()
df <- read_csv(file.path(sch_dir, "all_schools_wide.csv"))
```

```
Parsed with column specification:
cols(
  school = col_character(),
  math_1980 = col_double(),
  read_1980 = col_double(),
```

```
  science_1980 = col_double(),
  math_1981 = col_double(),
  read_1981 = col_double(),
  science_1981 = col_double(),
  math_1982 = col_double(),
  read_1982 = col_double(),
  science_1982 = col_double(),
  math_1983 = col_double(),
  read_1983 = col_double(),
  science_1983 = col_double(),
  math_1984 = col_double(),
  read_1984 = col_double(),
  science_1984 = col_double(),
  math_1985 = col_double(),
  read_1985 = col_double(),
  science_1985 = col_double()
)

# A tibble: 4 x 19
  school math_1980 read_1980 science_1980 math_1981 read_1981 science_1981
  <chr>      <dbl>     <dbl>        <dbl>     <dbl>     <dbl>        <dbl>
1 Bend …       515       281          808       503       312          814
2 East …       501       318          782       487       323          813
3 Niaga…       514       292          787       499       268          762
4 Spott…       498       288          813       494       270          765
# … with 12 more variables: math_1982 <dbl>, read_1982 <dbl>,
#   science_1982 <dbl>, math_1983 <dbl>, read_1983 <dbl>, science_1983 <dbl>,
#   math_1984 <dbl>, read_1984 <dbl>, science_1984 <dbl>, math_1985 <dbl>,
#   read_1985 <dbl>, science_1985 <dbl>
```