Data Manipulation with Tidyverse, Part II

Andie Creel

1 Picking up from Data Manipulation, Part I

In the last lecture, we used the following function from the `dplyr` package

- `mutate()`
- `if_else()`
- `filter()`
- `select()`
- `group_by()`
- `summarise()`

Today, we are going to learn about `join()` functions in the `dplyr` package, as well as `pivot()` functions in the `tidyr` package

- `left_join()`
- `pivot_longer()`
- `pivot_wider()`

First things first, let's load our packages

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

2 join() functions

You may need data from multiple data sets to be in one data set. We do this with `join()` functions.

The most common is `left_join()`, but you may also see `right_join()`, `inner_join()`, and `full_join()`.

A `left_join()` adds variables from a second dataset to your original dataset. This is what I almost exclusively use.

Let's create our employees dataset from Part I of our Data Manipulation Lectures

```
# ------------------------------------------------------------------------
# make a data frame of employees
# ------------------------------------------------------------------------
myEmps <- data.frame(
  id = 1:5,
  name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  department = c("IT", "Market", "HR", "Market", "IT"),
  salary = c(45000, 55000, 40000, 60000, 50000)
)

myEmps
```

```
##   id    name department salary
## 1  1   Alice         IT  45000
## 2  2     Bob     Market  55000
## 3  3 Charlie         HR  40000
## 4  4   David     Market  60000
## 5  5     Eva         IT  50000
```

Now, let's create a second data set that we want to combine with our employee data set

```
# ------------------------------------------------------------------------
# create second dataset
# ------------------------------------------------------------------------
myDepts <- data.frame(
  department = c("IT", "Market", "HR"),
  location = c("Building A", "Building B", "Building C"),
  boss = c("John Doe", "Jane Smith", "Mike Brown")
)

# ------------------------------------------------------------------------
# left_join
# ------------------------------------------------------------------------
myEmps_join<- left_join(x = myEmps, y = myDepts, by = "department")
myEmps_join
```

```
##   id    name department salary   location       boss
## 1  1   Alice         IT  45000 Building A   John Doe
## 2  2     Bob     Market  55000 Building B Jane Smith
## 3  3 Charlie         HR  40000 Building C Mike Brown
## 4  4   David     Market  60000 Building B Jane Smith
## 5  5     Eva         IT  50000 Building A   John Doe
```

3 Manipulating and cleaning with tidyr

Just like `dplyr`, `tidyr` has loads of very useful functions. There are two that I think are the most important to be aware of

- `pivot_longer()`
- `pivot_wider()`

4 pivot_longer()

Sometimes you'll want to pivot your data longer. Most models I work will require data to be formatted so that they are "long" by variables because that's usually the format best for regression models.

Ex. You want to have month be a variable in a regression (so you control for the month), but instead there are 12 columns for the month. You can pivot those columns into one column called `month`. You'd say you're data is "long by month" after you've done this formatting.

```
# -------------------------------------------------------------------------
# Create data set
# -------------------------------------------------------------------------
myTemps <- data.frame(
  location = c("Forest", "Desert"),
  Jan = c(5, 20),
  Feb = c(6, 22),
  Mar = c(10, 25)
)
myTemps
```

```
##   location Jan Feb Mar
## 1   Forest   5   6  10
## 2   Desert  20  22  25
```

```
# -------------------------------------------------------------------------
# pivot longer
# -------------------------------------------------------------------------
myTemps_long <- myTemps %>%
  pivot_longer(
  cols = Jan:Mar, # the columns we want to pivot
  names_to = "Month", # the new variable where the names of columns will be assigned
  values_to = "Temperature" # the new variable where the values will be assigned
)
myTemps_long
```

```
## # A tibble: 6 x 3
##   location Month Temperature
##   <chr>    <chr>       <dbl>
## 1 Forest   Jan             5
## 2 Forest   Feb             6
## 3 Forest   Mar            10
## 4 Desert   Jan            20
## 5 Desert   Feb            22
## 6 Desert   Mar            25
```

5 pivot_wider()

I usually pivot data frames from a long format to a wide format when the model I want to aggregate the data i.e., I want the data to be *less* granular. Usually you want data to me more grandular, but there are cases where one data set is less granular than another, and so you need them to match in order to merge them together.

Ex. You have one data set that is long by day. You want to combine it with another data set that isn't long by day and instead is only long by station and week. Therefore you want to get the total weekly rainfall, instead of having it as daily rainfall.

```
# -------------------------------------------------------------------------
# Create long data
# -------------------------------------------------------------------------
myRain <- data.frame(
  station = c("StationA", "StationA", "StationA",
              "StationB", "StationB", "StationB"),
  day = c("Monday", "Tuesday", "Wednesday",
          "Monday", "Tuesday", "Wednesday"),
  rainfall_mm = c(5, 10, 3,
                  0, 0, 12)
)

# -------------------------------------------------------------------------
# pivot so it's wide by station
# -------------------------------------------------------------------------

myRain_wide <- myRain %>%
  pivot_wider(names_from = day, values_from = rainfall_mm)

# -------------------------------------------------------------------------
# get weekly total rainfall
# -------------------------------------------------------------------------

myRain_weekly <- myRain_wide %>%
  mutate(weekly_rain = Monday + Tuesday + Wednesday) %>%
  select(-Monday, -Tuesday, -Wednesday)

myRain_weekly
```

```
## # A tibble: 2 x 2
##   station  weekly_rain
##   <chr>          <dbl>
## 1 StationA          18
## 2 StationB          12
```

New things introduced

- we used a "-" sign with the `select()` command to drop columns rather than selecting them