Joining Data

Some useful insights can be gained when one dataset is analyzed in the context of another dataset. For instance, if weather is expected to have an influence on sales, then it may be worth combining the weather measurements to the point-of-sale data.

Combining two datasets together in this way is typically done with a join. A join always results in a wider dataframe. Sometimes they can be used to create a longer dataframe, but that is less common. There are several types of joins. The one that I tend to use most often is a left join. A left join means that you keep all of the observations in the dataframe on the left. Observations on the right are only kept if they have a matching value in the dataframe on the left.

There are also right joins, inner joins, and full joins. A right join means that you keep all of the observations in the dataframe on the right. Only observations on the left are kept if they have a matching value in the dataframe on the right.

Inner joins only keep observations for which there is a matching value in both dataframes.

Full joins keep all observations in both the right and left dataframes even if there are not matching values.

In this lesson we'll demonstrate each of these joins using the following dplyr functions: left_join, right_join, inner_join, full_join.

Preliminaries

Load the dplyr, magrittr, and lubridate packages.

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## 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(magrittr)
```

```
## Warning: package 'magrittr' was built under R version 4.3.3
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
##
## Attaching package: 'lubridate'
```

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

Make sure that this file and the jan17Items.csv and jan17Weather.csv files are in the same folder and that the working directory is set to that folder.

Read in the jan17Items and jan17Weather data, and make sure to convert the date and time columns to a date and time format.

```
j17i <- read.csv('jan17Items.csv') %>%
  mutate(
    Time = ymd_hms(Time)
  )
j17w <- read.csv('jan17Weather.csv', sep = '\t') %>%
  mutate(
    date = ymd_hms(date)
  )
```

Left Join on Aggregated Data

For a join to work, we have to identify a column on which to match the data. This column is known as the primary key, and the values will only be joined if they values in the primary key column are identical. So one of the data wrangling tasks associated with joins is making sure that the values in the primary key column of both datasets are in the right format.

The obvious option for joining the weather data to the point-of-sale data is to use the Time and date column. Notice that while the Time column in the j17i data and the date column in the j17w data are both the same data type, all of the date values are rounded to the beginning hour, minute, and second of the day. There are probably few if any observations at midnight in our dataset.

Therefore, let's aggregate the point-of-sale data so that there's one observation for each day.

```
j17i %<>% mutate(date = round_date(Time, 'day'))
daily <- j17i %>%
  group_by(date) %>%
  summarise(avgCost = mean(Cost, na.rm = T)
            , maxPrice = max(Price, na.rm = T)
            , transactionQuantity = n_distinct(TransactionNumber)) %>%
  ungroup()
```

Notice that this daily dataframe has 30 observations and four variables. It does not include dates for January 1 and January 2, but does include an observation for February. In contrast, the weather data has an observation for each of the 31 days in January, and no observations for February.

Let's perform a left join with the point-of-sale data on the left and the weather data on the right and see what we get:

```
dailyLeft <- daily %>%
  left_join(j17w, by = 'date')
```

We now have a dailyLeft dataframe with 30 observations and 8 variables. A visual inspection confirms that the weather values are there only for the dates in the j17i dataframe, which is the left one in this case because that's what gets entered into the left_join function first by the pipe symbol. Because there wasn't weather data for February 1, there are NA values for the weather data on that date.

Right Join on Aggregated Data

Now let's keep almost everything the same, except that we'll replace the left_join function with the right_join function:

```
dailyRight <- daily %>%
  right_join(j17w, by = 'date')
```

The dailyRight dataframe has 31 observations because there is weather data for every day in January, and no observations for data in February. Notice that the rows for January 1 and Janary 2 have NA values for the point-of-sale columns, and there is not a value for February 1.

Inner Join on Aggregated Data

Let's compare the left and right join to an inner join.

```
dailyInner <- daily %>%
  inner_join(j17w, by = 'date')
```

The dailyInner dataframe has 29 observations, which correspond to the 29 days in January that are in both the point-of-sale and the weather data.

Full Join on Aggregated Data

Finally, let's evaluate a full join:

```
dailyFull <- daily %>%
  full_join(j17w, by = 'date')
```

The dailyFull dataframe keeps all observations in both dataframes. Naturally, it has the most observations. The 32 observations correspond to the 31 days in January plus the first day of February. Notice that there are NA values in the point of sale columns for January 1 and 2, and NA values in the weather columns for February 1.

Concluding Comments

In conclusion, joins are really easy to do. As you use them more, you'll find other options and other types of joins. I recommend that you think carefully about what the shape of the combined dataframe should be. If the number of rows grows unexpectedly, then it probably means that you have duplicate values in one of the dataframes. You end up learning a lot about your data as you prepare to perform joins.