Live Session 3
Data Wrangling

Babak Zafari

GEORGETOWN UNIVERSITY / McDONOUGH SCHOOL of BUSINESS

Outline

- Goal 1: Explain Data Wrangling Purpose and Techniques
- Goal 2: Apply the *tidyverse* Package
- Goal 3: Organize Data Using *dplyr*
- Goal 4: Apply Piping Functions in R
- Goal 5: Apply Join Functions in R

tidyverse

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

  - *readr* - for reading data in
  - *tibble* - for "tidy" data structures
  - ***dplyr* - for data manipulation**
  - ***ggplot2* - for data visualization**
  - *tidyr* - for cleaning
  - *purr* - functional programming toolkit

dplyr

dplyr aims to provide a function for each basic verb of data manipulation

Rows:

- `filter()` chooses rows based on column values.
- `slice()` chooses rows based on location.
  - Refer to `slice_min()` and `slice_max()` functions for choosing bottom/top *n* values.
- `arrange()` changes the order of the rows.

Columns:

- `select()` changes whether or not a column is included.
- `rename()` changes the name of columns.
- `mutate()` changes the values of columns and creates new columns.

Groups of rows:

- `summarize()` collapses a group into a single row.

pipe operator: %>%

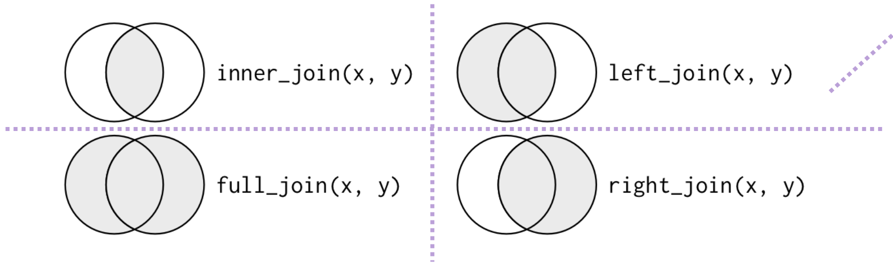
- All of the dplyr functions take a data frame (or tibble) as the first argument.
- Rather than forcing the user to either save intermediate objects or nest functions, dplyr provides the **%>% operator (i.e., pipe operator)**.

- Using pipe operator, a function such as `select(data,variable)` will be written as:
  - `data %>% select(variable),` so the result from one step is then “piped” into the next step

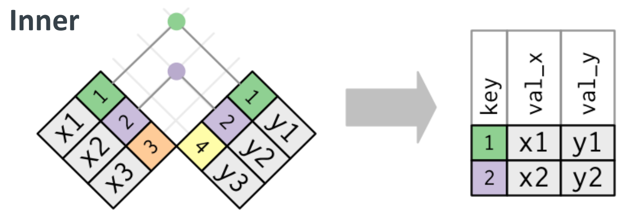
- You can use the pipe to rewrite multiple operations that you can read left-to-right, top-to-bottom (reading the pipe operator as “then”).

Joins

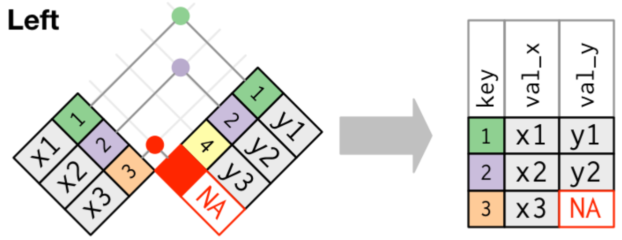
- Read more about joins here: [LINK]



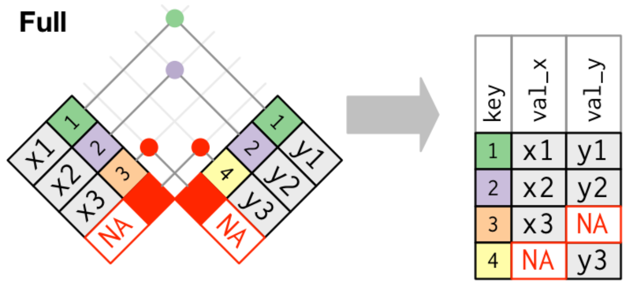
- A **left join** combines two datasets based on a common column (or key). It keeps all the rows from the **left** dataset and matches rows from the right dataset where the keys match.
- If there is no match in the right dataset, the result will still include the row from the left dataset, but with missing (NA) values for the columns from the right dataset.



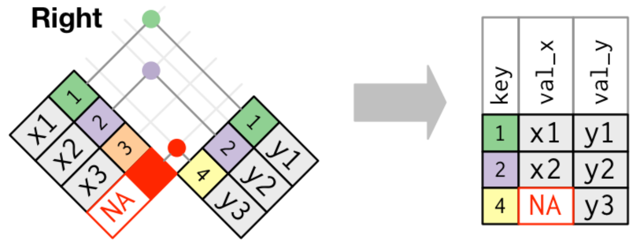
key	val_x	val_y
1	x1	y1
2	x2	y2



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3



key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

https://r4ds.had.co.nz/relational-data.html#understanding-joins

Exercise

In all the exercises, use tidyverse functions, preferably with the piping operator.

Load and save "***AuctionInfo_W3Live.csv***“ as auctionData and "***SellerInfo_W3Live.csv***“ as sellerData.

1. Update auctionData by removing/dropping the OpenPrice variable.

2. Merge the seller ratings from sellerData into auctionData. Make sure correct variables (AuctionID and ID) are used to perform the join.

3. Using pipes, group the auctionData by Category and calculate the average duration and seller rating for each category. Then, display the 3 categories with the shortest average duration.

4. Group the auctionData by both Category and endDay to calculate the average duration for each group. Afterward, modify the code to only display results where the auction's endDay is Friday.

5. Write a function using pipes that takes a category as input and calculates the average and standard deviation of the closing price for each endDay. Test your function on the Books category.