Class 5 Data Wrangling with R (Part II)

Dr Wei Miao

UCL School of Management

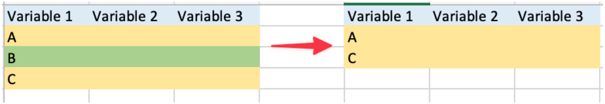
October 18, 2023

Section 1

Data Wrangling

Recap: filter(), arrange(), and mutate()

- dataset %>%``filter(criteria): pick observations by their values



- dataset %>%``arrange(variable): reorder the rows
- dataset %>%``mutate(newvariable = ...): create new variables with functions of existing variables

Pipe Operator

- The pipe has a huge advantage over any other method of processing data in R or Python: It makes data wrangling processes easy to read. If we read %>% as "then", the code will be very easy to interpret as a set of instructions in plain English:

```
mtcars %>%
  filter(cyl>=5) %>%
  mutate(sqrt_wt=sqrt(wt)) %>%
  filter(sqrt_wt>1.5) %>%
  arrange(hp)
## can go on and on
```

1. take the mtcars data, THEN
2. find cars cyl $>=$ 5, THEN
3. mutate a new variable sqrt_wt, THEN
4. find cars sqrt_wt $>$ 1.5 THEN
5. reorder all cars based on hp
6. chain more operations ...

Without Pipe Operator

- As a comparison, without using pipe operators, the previous data cleaning steps need to be done as follows. Overwriting our output dataframe `new_data` in every line is problematic.
  - First, doing this for a procedure with lots of steps isn't efficient and creates unnecessary repetition in the code.
  - Second, this repetition also makes it harder to identify exactly what is changing on each line in some cases.

```
new_data <- filter(mtcars, cyl >=5)
new_data <- mutate(new_data, sqrt_wt=sqrt(wt))
new_data <- filter(new_data,sqrt_wt>1.5)
new_data <- arrange(new_data,hp)
```

Select Variables: `select`

- `select()` can select variables into a smaller dataset.

```
# Select two columns: hp and cyl
mtcars%>%
  select(hp, cyl) %>%
  head()
```

```
Error in mtcars %>% select(hp, cyl) %>% head(): could not find function
```

Aggregation by Groups: `group_by`

- `group_by()` allows us to aggregate data by group and compute statistics for each group

```
# group by cyl
mtcars %>%
  group_by(cyl)
```

- Although nothing seemingly happens to the dataset, internally, the dataset is already grouped based on the specified variable(s).

Aggregation by Groups: `group_by()` + `summarise()`

- `summarise()` creates a new data frame after aggregating data. The final dataset
  - has one row for each pair of grouping variables (for each `cyl` value)
  - contains one column for each grouping variable (`cyl`)
  - contains one column for each new summarised variable (`avg_mp`)

```
# compute the average mpg for each cyl group
mtcars %>%
  group_by(cyl) %>% # group by cyl
  summarise(avg_mp = mean(mpg)) %>% # compute the average mpg
  ungroup()
```

```
Error in mtcars %>% group_by(cyl) %>% summarise(avg_mp = mean(mpg)) %>%
```

Aggregation by Groups: group_by() + summarise()

- We can have multiple group variables for group_by

```
# compute the average mpg for each cyl,vs group
mtcars %>%
  group_by(cyl,vs) %>% # group by cyl
  summarise(avg_mp = mean(mpg)) %>% # compute the average mpg
  ungroup()
```

Aggregation by Groups: `group_by() + mutate()`

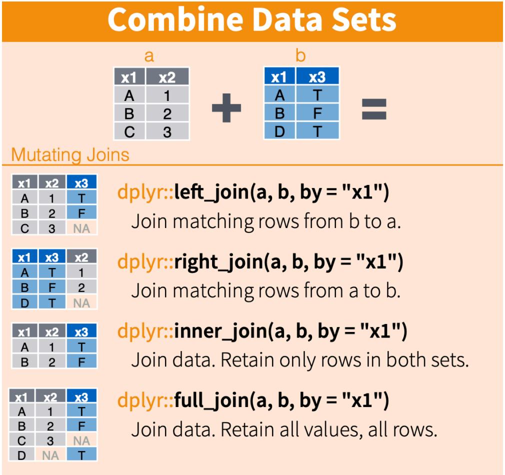
- Try the following code by replacing `summarise()` with `mutate()`, what do you get now?

```
# compute the average mpg for each cyl,vs group
mtcars %>%
  group_by(cyl,vs) %>% # group by cyl
  mutate(avg_mp = mean(mpg)) %>% # compute the average mpg
  ungroup()
```

- A new column is added to the original dataset, the value of which is from the group-by aggregation.

Consolidate Multiple Data Frames

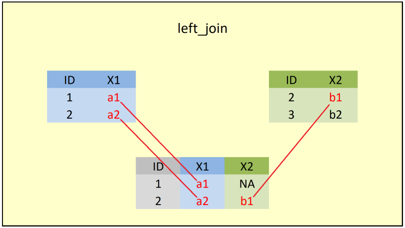
- When consolidating multiple data frames, we usually have 4 types of joining methods



left_join

- `left_join` keeps everything from the left data frame and matches as much as it can from the right data frame.
  - All IDs **in the left data frame** will be retained
  - If a match can be found, value from the right data frame will be filled in
  - If a match cannot be found, a missing value will be filled in

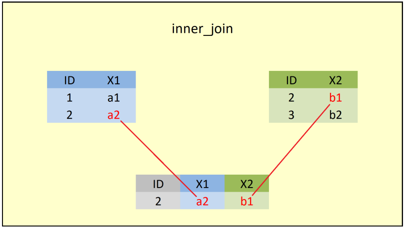
```
# Method 1 without pipe operator
left_join(df_left, df_right, by = 'ID')
# Method 2 with pipe operator
df_left %>%
  left_join(df_right, by = 'ID')
```



inner_join

- inner_join only keeps the observations that appear in both data frames
  - Only common IDs **in both data frames** will be retained
  - If a match can be found, values will be filled in from both data frames

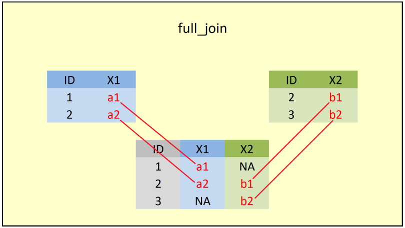
```
# Method 1 without pipe operator
inner_join(df_left, df_right, by = 'ID')
# Method 2 with pipe operator
df_left %>%
  inner_join(df_right, by = 'ID')
# Method 3: order of data frames should not matter. Why?
df_right %>%
  inner_join(df_left, by = 'ID')
```



full_join

- `full_join` keeps all observations from both data frames
  - All IDs **in either data frames** will be retained
  - If a match can be found, values will be filled in from both data frames

```
# Method 1 without pipe operator
full_join(df_left, df_right, by = 'ID')
# Method 2 with pipe operator
df_left %>%
  full_join(df_right, by = 'ID')
# Method 3: order of data frames should not matter. Why?
df_right %>%
  full_join(df_left, by = 'ID')
```



Section 2

Data Cleaning

Variable Types

- Non-metric
  - Categorical (gender, region, brand, religion)
  - Ordinal (Business Week rankings, NCAA rankings)
- Metric
  - Continuous (age, height, sales, rainfall)
- Different types of variables are handled in different ways in statistics
  - Can talk about an average age, but not an average color
  - Some statistical techniques only work with one type of variable
- We need to make sure the variables are of the correct data types. Or we may need to convert them to the correct types.
  - e.g., from character to date time using `lubridate` package

Missing Values

- In R, missing values are represented by the symbol `NA` (i.e., *not available*).
- Most statistical models cannot handle missing values, so we need to deal with them in R.
  - Few missing values: remove them from analysis.
  - Many missing values: need to replace them with appropriate values: mean/median/imputation

Section 3

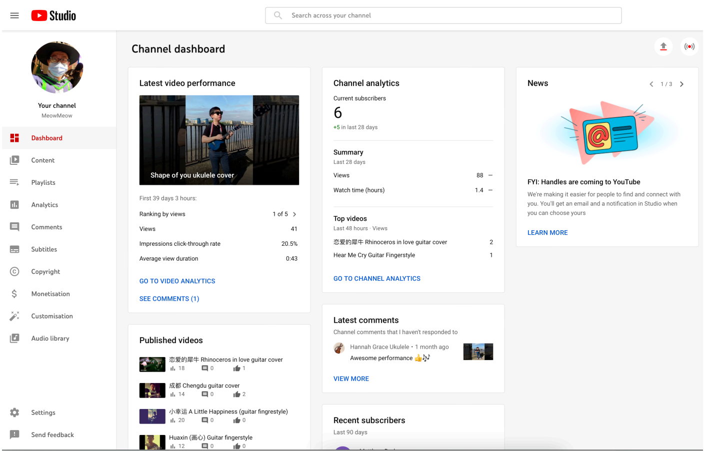
Descriptive Analytics

Two Major Tasks of Descriptive Analytics

1. **Describe data depending on your business purposes**
   - "How much do our customers spend each month on average?"
   - "What percentage of our customers are unprofitable?"
   - "What is the difference between the retention rates of men and women?"
2. **Make statistical inferences from data**
   - "Based on our sample, does the difference between the spendings of men and women indicate that men and women respond differently in the customer base at large?"
   - "Based on our sample, can we conclude that customers who sign up for online banking are more profitable than customers who do not?"
   - "Based on our test mailing, can we conclude that ad-copy A works better than ad-copy B?"

Descriptive Analytics

- You can think of descriptive analytics as **creating a dashboard** to display the key information you would like to know for your business.



Summary Statistics

- In descriptive analytics, **summary statistics** are used to summarize a set of observations, in order to communicate the largest amount of information as simply as possible.
- There are two main types of summary statistics used in evaluation: **measures of central tendency** and **measures of dispersion**.
  - Measures of central tendency provide different versions of the average, including the mean, the median, 25 percentile, 75 percentile, the mode, etc.
  - Measures of dispersion provide information about how much variation there is in the data, including the range and the standard deviation.
- It's good to include summary statistics table in your dissertation before any statistical analysis!
  - Commonly reported summary statistics include *mean*, *standard deviation*, *number of observations*, *min*, *25 percentile*, *median*, *75 percentile*, and *max*.
  - Then describe the distribution of the variable, dispersion of the variable, etc.

Summary Statistics with R

- In R, a nice package to report summary statistics is `modelsummary`. `datasummary_skim()` is a shortcut to conduct basic summary statistics
- For more features, refer to the package tutorial here, especially `datasummary()` function.
  - `datasummary_skim()` is a special case of more general `datasummary()`, which outputs a pre-determined set of summary statistics

```
pacman::p_load(modelsummary)
mtcars %>%
  datasummary_skim()
```

Correlation Matrix

- Correlation matrix helps us understand the co-movement of any two variables in the data
- `datasummary_correlation()` reports the pairwise correlation coefficient
- In general, in a statistical model, variables of high correlation should not be included together, which leads to instability

```
mtcars %>%
  datasummary_correlation()
```

Section 4

Preliminary Customer Analysis

Preliminary Customer Analysis

- Spend 30 min-ish to work on the case study in a group.
- There are 7 questions in total. At the end of the discussion, each group selects a group leader to answer one question.
- To show your codes, join the Zoom link under “Module Overview” on Moodle.
- The group that did not answer correctly needs to do a performance next week!

After-Class Exercise

- What percent of customers are single? Try alternative ways to do the calculation.
- Is the average total spending by responders and non-responders statistically different? Answer this question using a t-test.
- Is income and total spending correlated?
- Are PhDs more likely to respond to marketing offers than Graduation? Use a statistical test to answer the question. Is the result what you expected?
- What would be the other useful descriptive analytics you would like to know for Tesco?