# Class 5 Data Wrangling with R (Part II)

Dr Wei Miao

UCL School of Management

October 18, 2023

Section 1

# Data Wrangling Part II

# Data Wrangling Part II

- Select rows (`filter`)
- Sort rows (`arrange`)
- Select columns (`select`)
- Generate new columns (`mutate`)
- **Group aggregation (`group_by`): compute statistics for each group**
- **Merge datasets (`join`): combine datasets from different sources**

# Aggregation by Groups: group_by

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

```
# group by marital status
data_demo %>%
  group_by(Marital_Status)
```

- Internally, the dataset is already grouped based on the specified variable(s).

# Aggregation by Groups: `group_by() %>% summarise()`

- After aggregating data, we can use `summarise()` to compute group-specific statistics for us.
  - Similar to `mutate()` in generating new variables
  - Different from `mutate()` in that the new variable is computed based on groups.

```
# compute the average income for each marital status group
data_demo %>%
  group_by(Marital_Status) %>%
  summarise(avg_income = mean(Income,na.rm = T)) %>%
  ungroup()
```

| Marital_Status | avg_income |
|---|---|
| Alone | 43789.00 |
| Divorced | 53300.63 |
| Married | 51776.36 |
| Single | 51091.16 |
| Together | 52609.59 |
| Widow | 56158.90 |

- What if you replace `summarise()` with `mutate()`?

# Aggregation by Groups: `group_by()` Multiple Groups

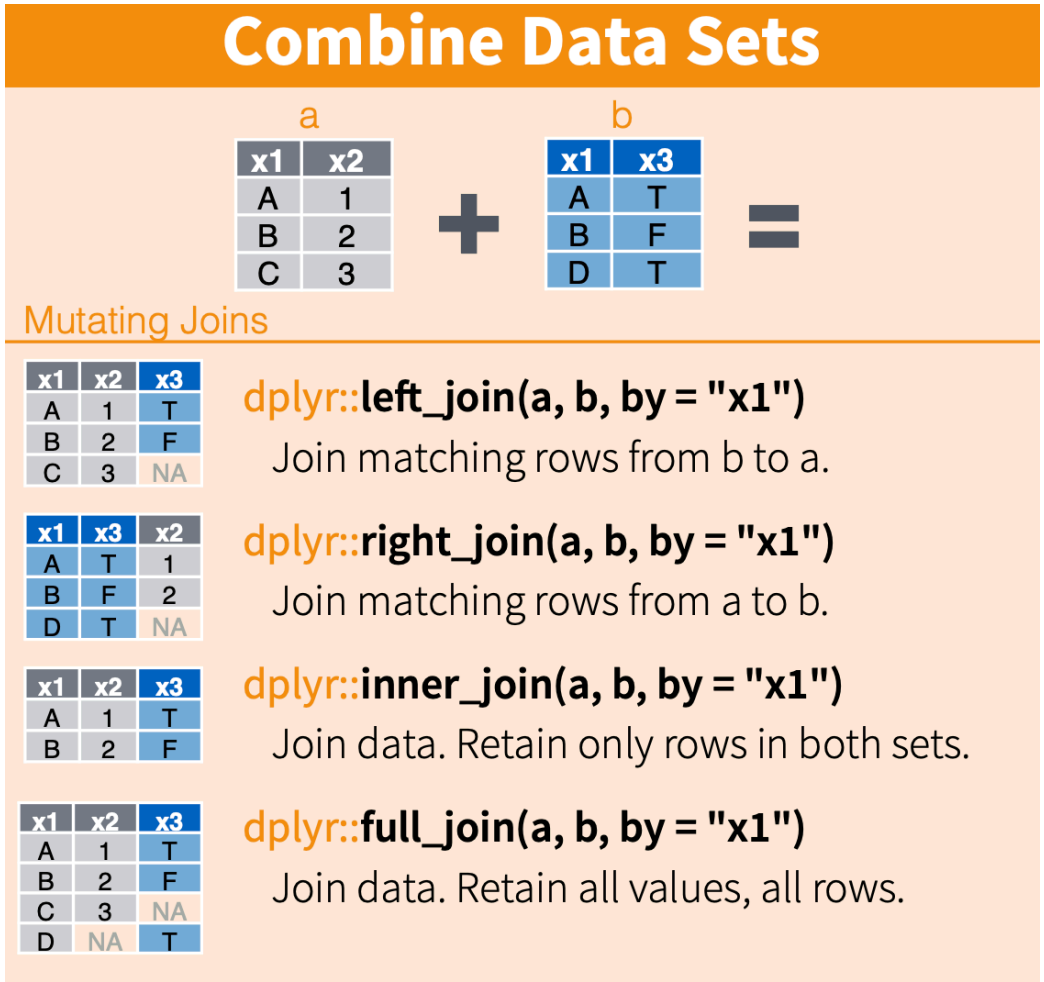
- We can have multiple group variables for `group_by` , such as computing average income for each marital status, education combination

```
# compute the average income for each marital, education group
data_demo %>%
  group_by(Marital_Status,Education) %>%
  summarise(avg_income = mean(Income,na.rm = T)) %>%
  ungroup() %>%
  head(5)
```

| Marital_Status | Education | avg_income |
|---|---|---|
| Alone | Graduation | 34176 |
| Alone | Master | 61331 |
| Alone | PhD | 35860 |
| Divorced | 2n Cycle | 49345 |
| Divorced | Basic | 9548 |

# Consolidate Multiple Data Frames

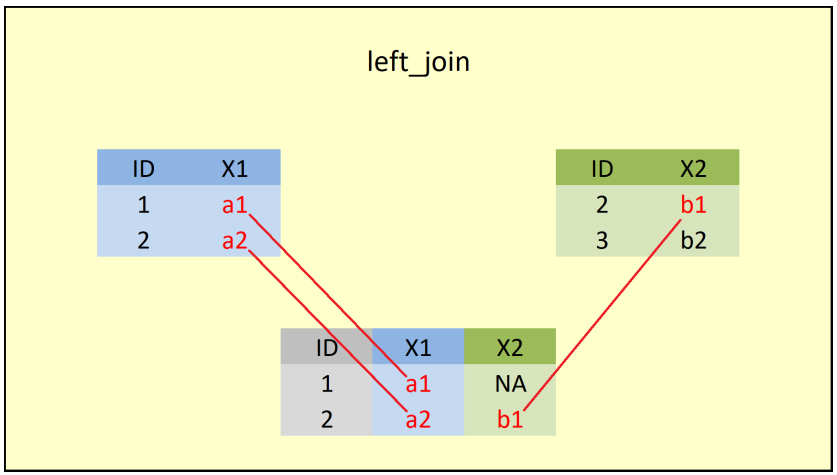
- When consolidating multiple data frames, we have 4 types of joining methods.
- `left_join()` handles most data join situations, which we will focus on today.



# left_join()

- `left_join` keeps everything from the **left data frame** and matches as much as it can from the right data frame based on the chosen IDs.
  - 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 returned

```
1 df_left %>%
2   left_join(df_right, by = c('ID' = 'ID') )
```



# Caveats for doing `left_join()`

- We can do 1:1, or M:1 left_joins
- **Never** do 1:M or M:M left_joins

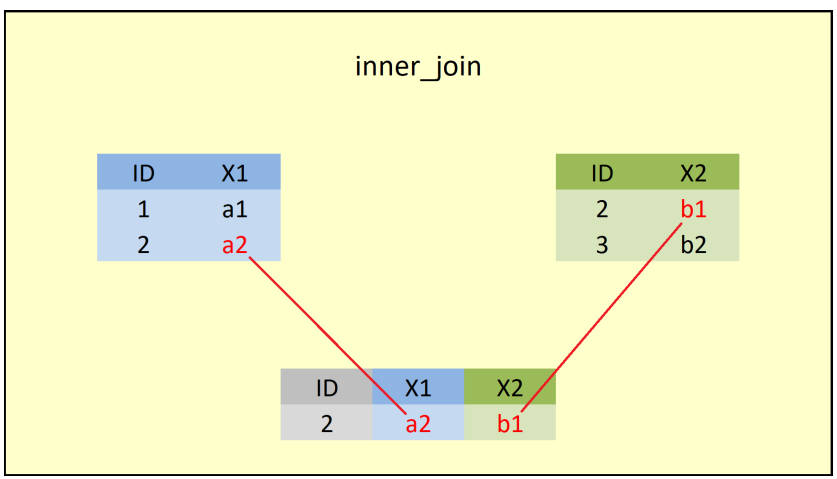
| demographics: 1 | | |
|---|---|---|
| ID | age | income |
| 1 | 15 | 3000 |
| 2 | 18 | 1000 |

| Transaction data: M | | |
|---|---|---|
| ID | transaction | revenue |
| 1 | 1 | 12 |
| 1 | 2 | 2 |
| 2 | 1 | 10 |
| 2 | 2 | 5 |
| 2 | 3 | 10 |

# inner_join() (optional)

- 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

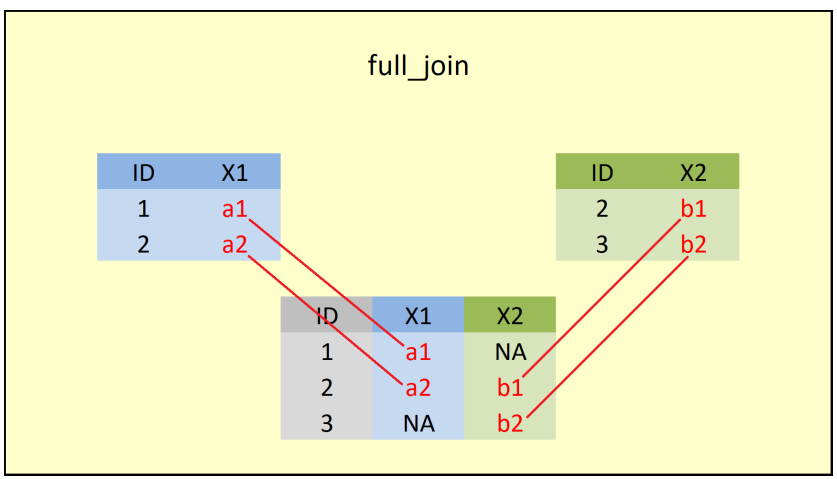
```
# Method 1 without pipe operator
inner_join(df_left, df_right, by = 'ID')
# Method 2 with pipe operator
df_left %>%
  inner_join(df_right, by = 'ID')
# Method 3: order of data frames should not matter. Why?
df_right %>%
  inner_join(df_left, by = 'ID')
```



# full_join() (optional)

- 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

# 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

# Outliers

- Sometimes, due to data collection errors, we may have abnormal observations in the data, such as unusually large and small values
- Winsorization is a common way to deal with outliers
  - Remove top 1% and bottom 1% observations

Section 3

# Descriptive Analytics

# Two Major Tasks of 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.

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 spending 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?"

# Summary Statistics

- **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**: mean, the median, 25 percentile, 75 percentile, the mode, etc.
  - **measures of dispersion:** range and standard deviation.
- It's important to include summary statistics table in your dissertation before any statistical analysis!

# 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

```
pacman::p_load(modelsummary)
## Summary statistics for numeric variables
data_demo %>%
  datasummary_skim(type = "numeric")

## Summary statistics for categorical variables
data_demo %>%
  datasummary_skim(type = "categorical")
```

# Case Study: Preliminary Customer Analysis

- Let's solve the preliminary customer analysis case together in class!