Lecture 1-2: Data wrangling

BTBI30081

統計應用方法

2025/2/19

Data analysis process

1. Data import (tidying data)
2. Data transformation (data manipulation)
3. Data visualization
4. Modeling

Each of these steps need their own tools and software to complete

Bottlenecks in data analysis

- One of the most time-consuming aspects of the data analysis process is "data wrangling" or "data munging"
    - Import, clean and transform messy data into a format that is useful for data visualization and modeling
    - Refer to the first two steps in the data analysis process

Package: tidyverse

- The tidyverse is a collection of R packages designed for data science
- The core tidyverse includes the packages

tibble	simple data frames
readr	read rectangular text data
dplyr	a grammar of data manipulation
tidyr	easily tidy data
ggplot2	grammar of graphics
purrr	functional programming tools

- tibble, readr, tidyr, dplyr in tidyverse are for data wrangling

Data import

- The first step in data analysis is importing the data into the R environment

- The are several function in the base package available for reading data
  - read.table – sep=“” (white space)
  - read.csv – sep=“,” (comma)
  - read.delim – sep=“\t” (Tab)
  - These functions are identical except for the “field separator character” are different.
  - If it does not contain an absolute path, the file name is relative to the current working directory, getwd().

Example

- We took a poll of our students to obtain (self-reported) height and gender. Our task is to describe this list of heights.

Different ways to import data into R

- Option 1: Download file with your browser to your working directory
- Option 2: Read from within R
- Option 3: Download from within R
- RMD_example 01-2.1

Data types

- dat <- read.csv(filename)
  - We make assignments in R: "<-"
  - We put the content of what comes out of read.csv into an **object** "dat"
  - The data type of dat is "data.frame"– one the most widely used data types in R

Tidy data type: tibble

- tibble (or tbl_df) is a modern reimagining of the data.frame, keeping what time has proven to be effective, and throwing out what is not
- In tidyverse, all functions adopt and produce tibble– one of the unifying features of the tidyverse
- Creating tibble: RMD_example 01-2.2

Data import with readr

- We can use the functions in readr package in tidyverse to import data, which will create tibble data type
    - read_csv
    - read_tsc
    - read_delim
- RMD_example 01-2.2

Data manipulation with base functions

- Extracting columns, Quick review of vectors, Coercion
- RMD_example 01-2.3

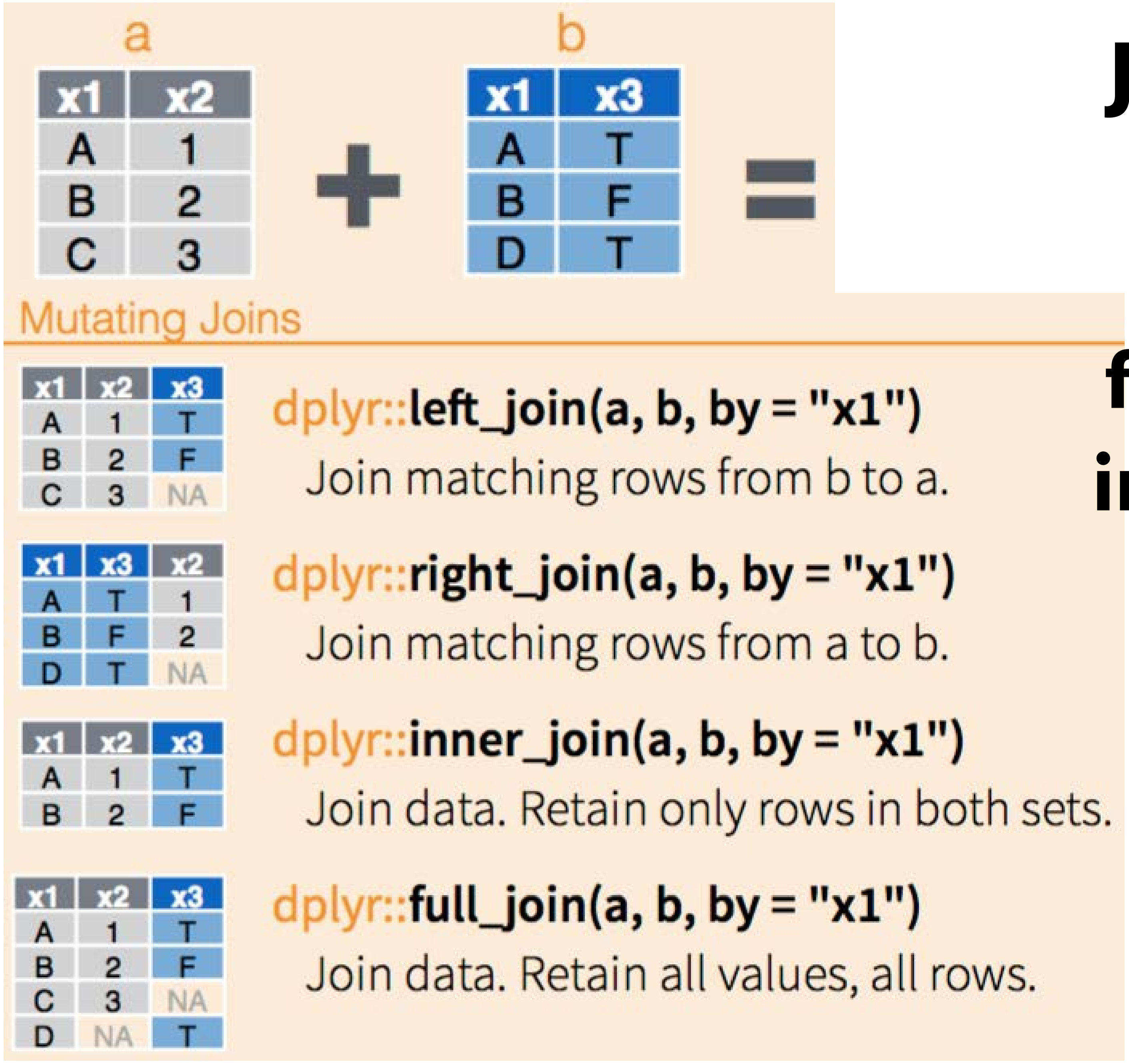
Data manipulation with dplyr

- Important dplyr functions to remember

select()	select columns
mutate()	create new columns
filter()	filter rows
arrange()	arrange or re-order rows
group_by()	grouping operations
summarise()	summarise values

- RMD_example 01-2.4

Joining two data frames in dplyr



a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

dplyr::**left_join(a, b, by = "x1")**

Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

dplyr::**right_join(a, b, by = "x1")**

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

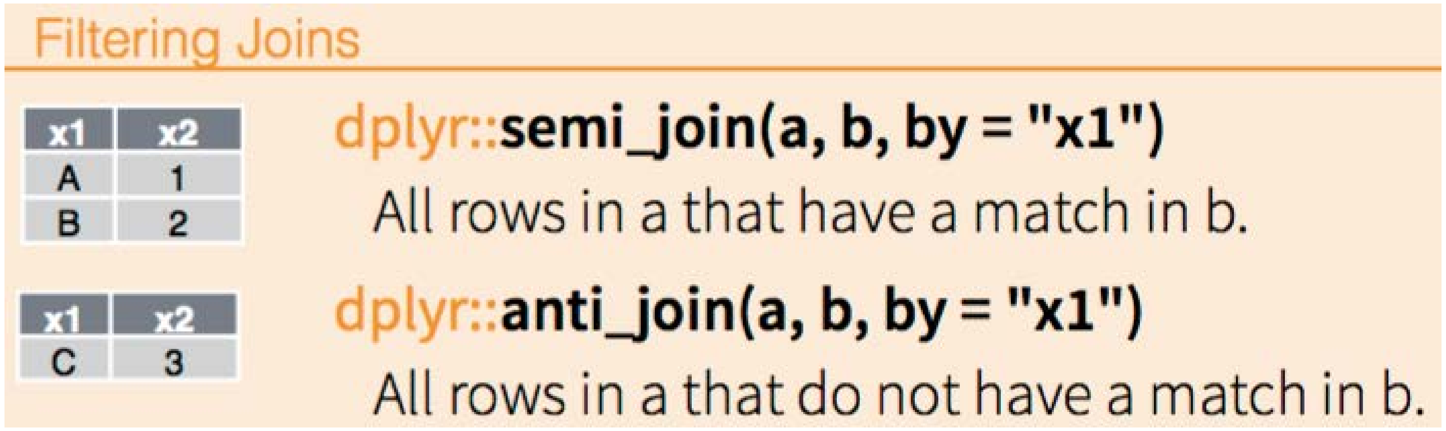
dplyr::**inner_join(a, b, by = "x1")**

Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

dplyr::**full_join(a, b, by = "x1")**

Join data. Retain all values, all rows.



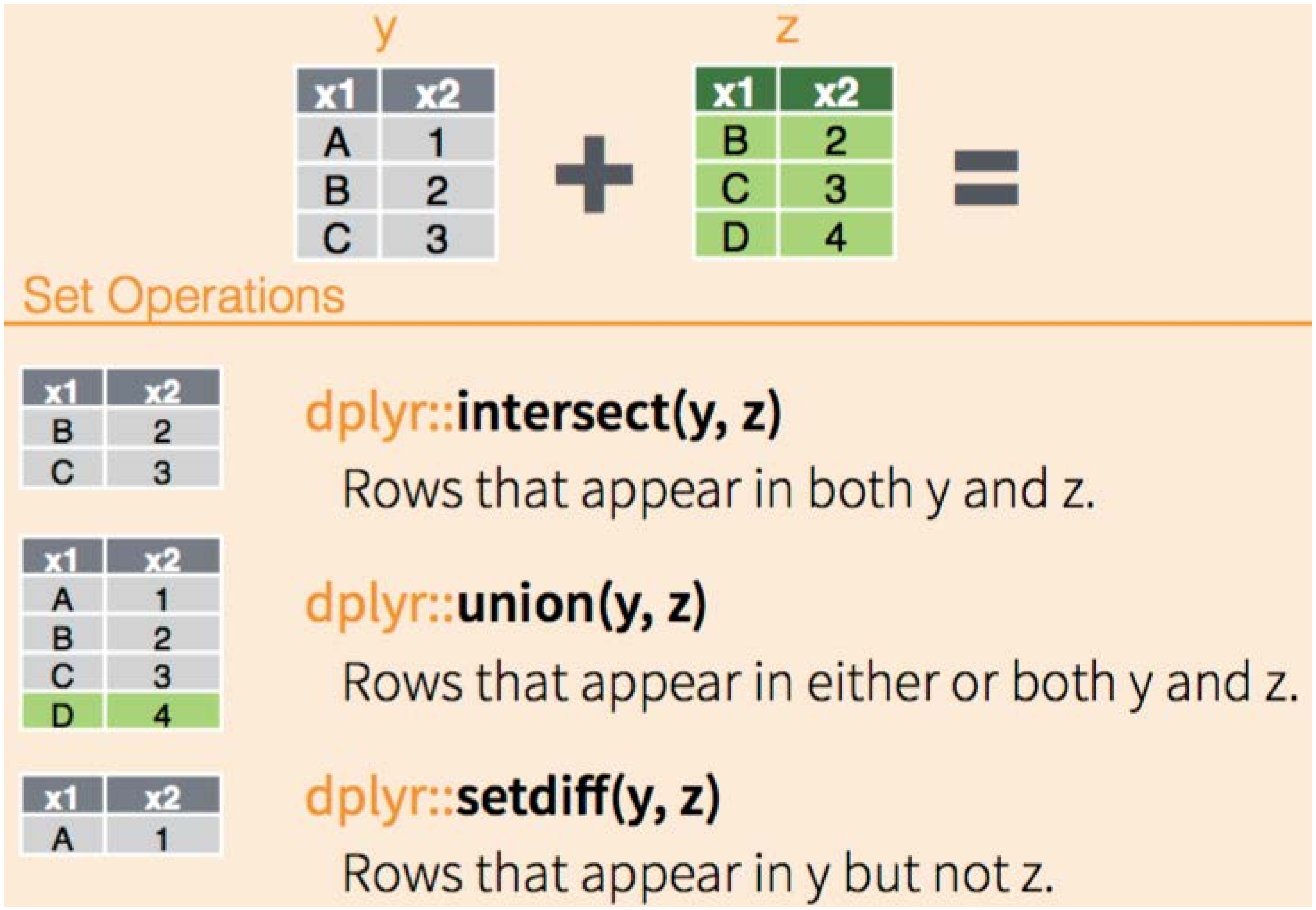
Filtering Joins

x1	x2
A	1
B	2

dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

x1	x2
C	3

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.



y

x1	x2
A	1
B	2
C	3

+

z

x1	x2
B	2
C	3
D	4

=

Set Operations

x1	x2
B	2
C	3

dplyr::**intersect(y, z)**

Rows that appear in both y and z.

x1	x2
A	1
B	2
C	3
D	4

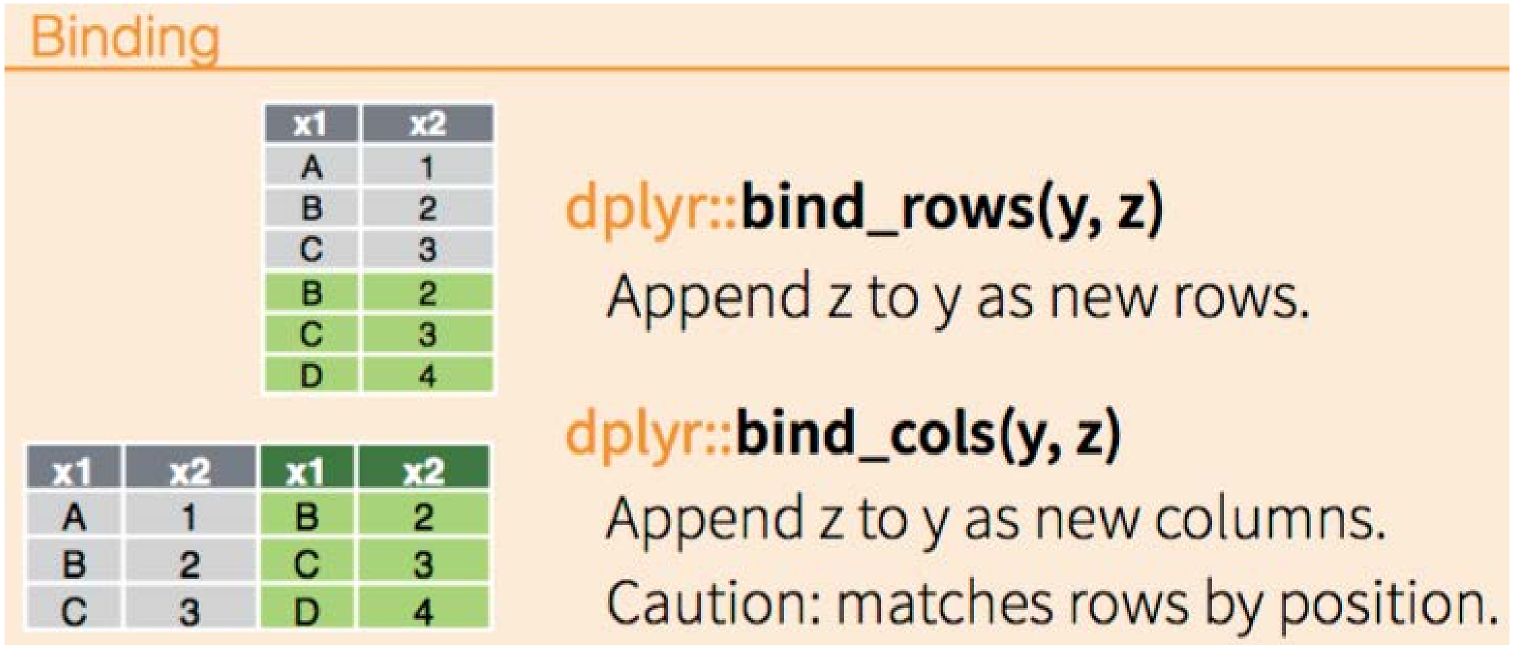
dplyr::**union(y, z)**

Rows that appear in either or both y and z.

x1	x2
A	1

dplyr::**setdiff(y, z)**

Rows that appear in y but not z.



Binding

x1	x2
A	1
B	2
C	3
B	2
C	3
D	4

dplyr::bind_rows(y, z)
Append z to y as new rows.

x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4

dplyr::bind_cols(y, z)
Append z to y as new columns.
Caution: matches rows by position.

More data transformation with dplyr

- https://rstudio.github.io/cheatsheets/html/data-transformation.html