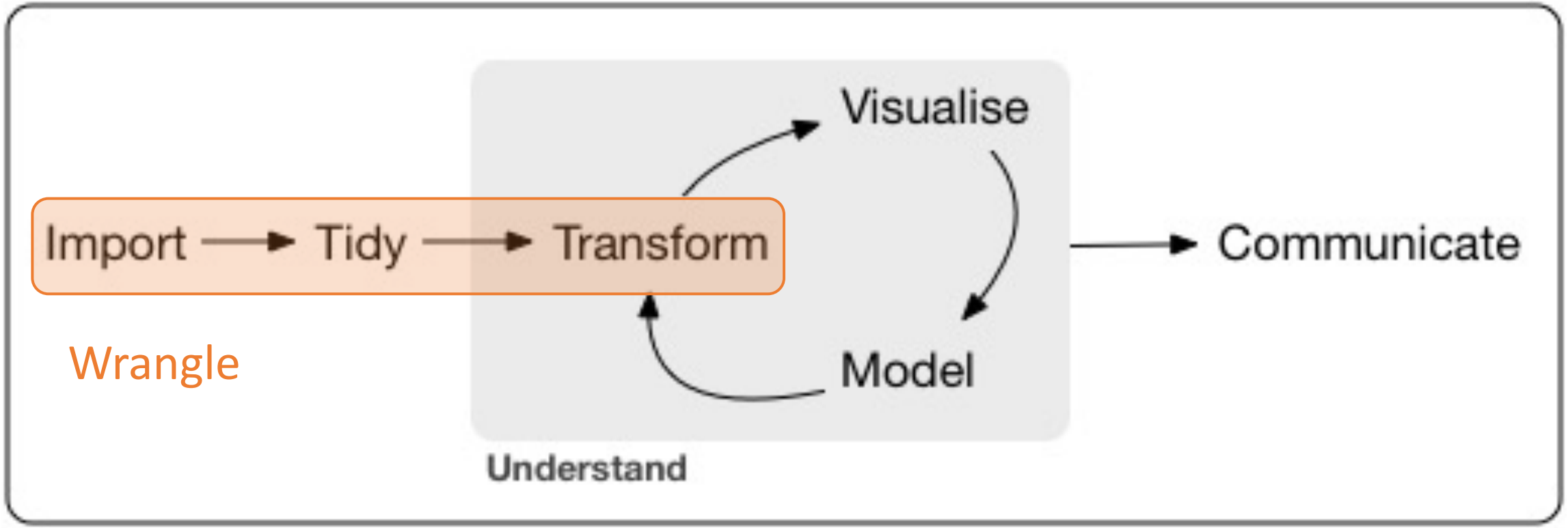
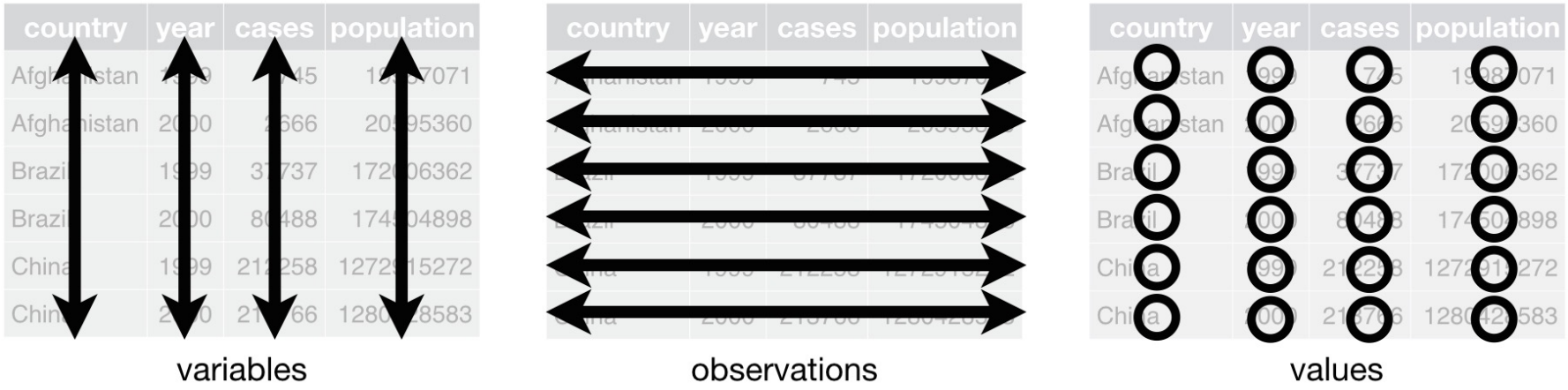
Week 5: Data wrangling (II)


Visualise
Import
Tidy
Transform
Communicate
Model
Wrangle
Understand

What makes a dataset tidy?

- Each variable must have its own column.
- Each observation must have its own row.
- Each value must have its own cell.



dplyr, ggplot2, and all the other packages in the tidyverse are designed to work with tidy data.

Let's get set

- Create an R project for this session and name it “session_5”
- Download the script file (.R) and data file (.CSV) and place them in the folder “session_5”
- Open the script file
- Load the tidyverse package
- Import the data file and name it “sinai_covid”

The glimpse function

- When you need a quick, compact summary of the data use the glimpse function.

Core functions

Operate on the variables (i.e. the columns)

Operate on the observations (i.e. the rows)

Function	Utility	Package
%>%	“pipe” (pass) data from one function to the next	**magrittr**
clean_names()	standardize the syntax of column names	**janitor**
rename()	rename columns	**dplyr**
select()	selects a subset of variables to retain and (optionally) renames them in the process	**dplyr**
mutate()	create, transform, and re-define columns	**dplyr**
filter()	keep certain rows	**dplyr**
arrange()	sort rows	**dplyr**

Working with variables

clean_names() - Automatic cleaning

- The function clean_names() from the package **janitor** standardizes column names. It converts all names to consist of only underscores, numbers, and letters

rename() - Manual name cleaning

- Re-naming columns manually is often necessary
- Re-naming is performed using the rename() function from the dplyr package.
- rename() uses the style NEW = OLD (the new column name is given before the old column name)

Select()

- Use select() from **dplyr** to select, specify the order, and remove columns

Mutate()

- Use the function mutate() to **add a new column**, or to modify an existing one.
- The syntax is: mutate(new_column_name = value or transformation)

Working with observations

Subset observations with filter()

- Use the function filter() to **subset observations** in a data frame or tibble. This is often done when we want to limit an analysis to a subset of observations.

Reording observations with arrange()

- Use the function arrange() to **reorder the rows** of an object. This is sometimes used when we want to inspect a dataset to look for associations among the different variables.

Case Study

- The tidyr::who dataset contains tuberculosis (TB) cases broken down by year, country, age, gender, and diagnosis method.
- This is a very typical real-life example dataset.
- The best place to start is to gather the columns that are not variables.

Can you identify them?

Case Study

1. new_sp_m014 - new_rel_f65 are counts of new TB cases recorded by group.
2. We need to gather all the columns (group). We know the cells represent the count of cases, so we'll use the variable cases.
3. There are a lot of missing values in the current representation, so we'll use values_drop_na

Case Study – data dictionary

- The first three letters of each column denote whether the column contains new or old cases of TB. In this dataset, each column contains new cases.
- The next two letters describe the type of TB:
  - rel stands for cases of relapse
  - ep stands for cases of extrapulmonary TB
  - sn stands for cases of pulmonary TB that could not be diagnosed by a pulmonary smear (smear negative)
  - sp stands for cases of pulmonary TB that could be diagnosed by a pulmonary smear (smear positive)
- The sixth letter gives the sex of TB patients. The dataset groups cases by males (m) and females (f).
- The remaining numbers gives the age group. The dataset groups cases into seven age groups:
  - 014 = 0 – 14 years old
  - 1524 = 15 – 24 years old
  - 2534 = 25 – 34 years old
  - 3544 = 35 – 44 years old
  - 4554 = 45 – 54 years old
  - 5564 = 55 – 64 years old
  - 65 = 65 or older

- What do you think should be done to tidy this data?