Tidy data: : **CHEAT SHEET**

Overview

The **Definition** of tidy data given by Hadley Wickham:

- Each **variable** forms a **column**.
- Each **observation** forms a **row**.
- Each **observational unit** forms a **value in the table**.

Package

- library(dplyr)
- library(tidyr)

Both two packages are a part of **tidyverse**, a collection of helpful packages in R.

dplyr

rename(data_frame, new_name = old_name)

Change names of the column from 'old_name' to 'new_name'

select(data_frame, column1, column2)
select(data_frame, -column1, -column2)

Return a subset of the data. use a minus sign **'-'** to drop columns.

mutate(data_frame, new_colunm = function)

Compute new variables from existing variables and add them to the table.

e.g. starwars %>%
select(name, mass) %>%
mutate(mass2 = mass * 2)

filter(data_frame, filter_function)

Return a modified copy of certain rows based on the filter_function.

e.g. filter(iris, Petal.Width > 0.3)

arrange(data_frame, column1, desc(column2))

Reorder the rows based on their value in the ascending order by default. Use **desc()** to reorder in descending order.

group_by(data_frame, column1, column2)
ungroup(grouped_data_frame)
summarize(grouped_data_frame,
max(column1), min(column2))

The **group_by** function groups the data by the values of the variables. Then we can use **summarize** function to create a new data frame with summary statistics such as minimum, maximum, average.

Use **ungroup** to remove grouping.

tidyr

pivot_longer(data_frame, columns,
names_to, values_to)

A common problem is a dataset where some of the column names are **not names** of variables, but **values** of a variable.

Pivot the offending columns into a new pair of variables.

pivot_wider(data_frame, names_from,
values_from)

The opposite of **pivot_longer()**.

separate(data_frame, column,
into = c(col1, col2), sep = '/')

separate() pulls apart one column into multiple columns, by splitting wherever a separator character appears.

Default separator is **forward slash.** You can use a specific character or pass a vector of integers to specify the position to split at.

unite(data_frame, new_column, col1, col2,
sep = '')

The inverse of **separate()**. We can use a specific character to concatenate columns to be united.

Example

Pivoting

```
table4a
#> # A tibble: 3 x 3
#>   country     `1999` `2000`
#> * <chr>        <int>  <int>
#> 1 Afghanistan    745   2666
#> 2 Brazil       37737  80488
#> 3 China       212258 213766
```

Obviously, here the column '1999' and '2000' are **values** of year. So we use **pivot_longer()** to add columns 'year' and 'cases' based on the names of columns and values of them.

```
table4a %>%
pivot_longer(c(`1999`, `2000`),
        names_to = "year", values_to = "cases")
```

```
#> # A tibble: 6 x 3
#>   country     year   cases
#>   <chr>       <chr>  <int>
#> 1 Afghanistan 1999     745
#> 2 Afghanistan 2000    2666
#> 3 Brazil      1999   37737
#> 4 Brazil      2000   80488
#> 5 China       1999  212258
#> 6 China       2000  213766
```

See more detailed examples about function in **dplyr** and **tidyr** on following websites:
dplyr: https://edav.info/tidy.html
tidyr: https://r4ds.had.co.nz/tidy-data.html

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK