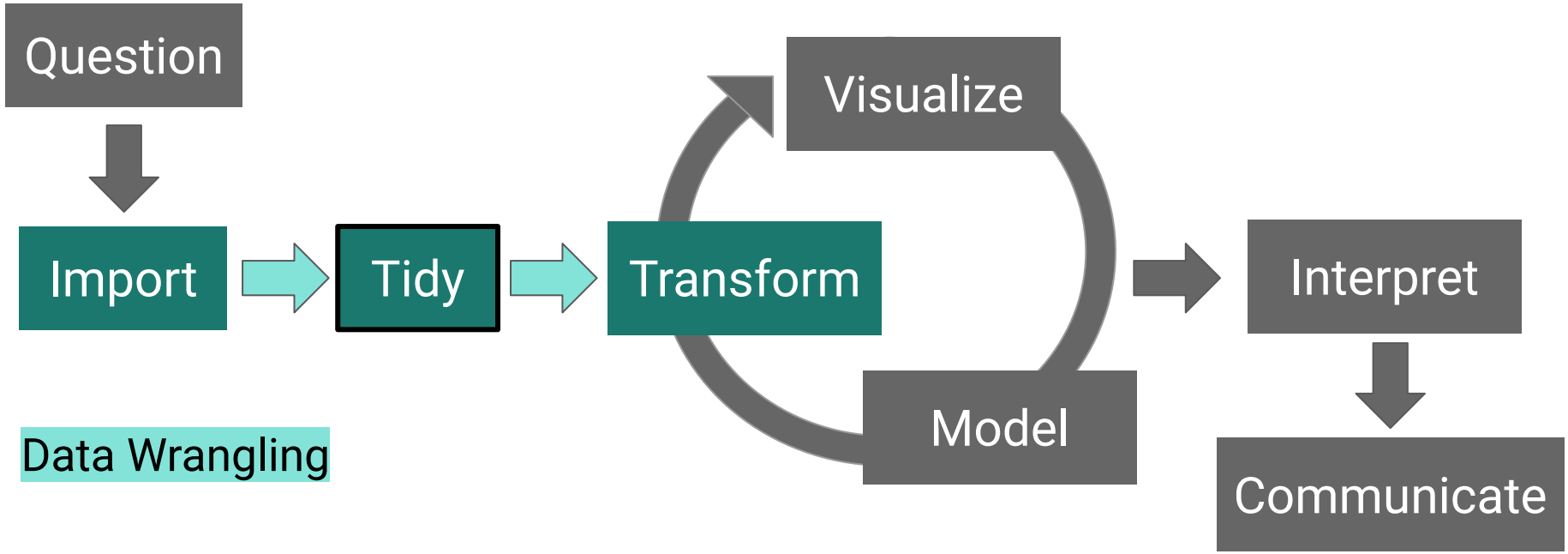
Tidying Data

Lecture 5

Objective

- To tidy data

Motivation



Modified from https://r4ds.had.co.nz/explore-intro.html#explore-intro

Definition of terms

- A **dataset** is a collection of **values**, either numbers (quantitative) or strings (qualitative)
- Values are organized in 2 ways: each value belongs to a **variable** and an **observation**
- A **variable** contains all values that measure the same attribute (e.g. age, height, temperature) across units
- An **observation** contains all values measured on the same unit (e.g. a person, a day, a population) across attributes

Tidy data

1. Each column is a variable
2. Each row is an observation
3. Each cell is a single value

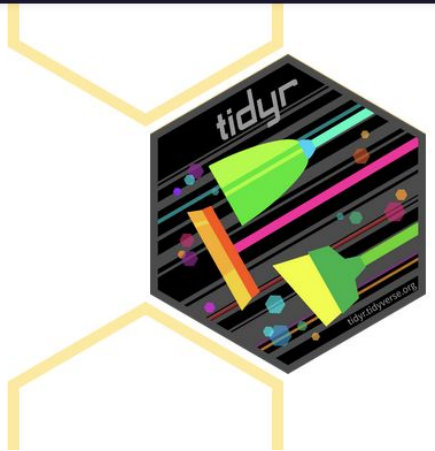
```
> patient
```

```
# A tibble: 5 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
4 P004       male          3      13.2        95
5 P005       male          4      15.9       104
```

tidyr package

Tidyverse

Packages



tidyr

tidyr provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable. Go to docs...

https://www.tidyverse.org/packages/

Advantages with tidy data

- Easy to work with data in consistent structure
- All packages in tidyverse work with tidy data
- Variables as columns facilitate vectorization since most function in R work with vector of values

```
# with vectorization
> a <- c(1,2,3)
> b <- c(4,5,6)
> c <- a + b
> c
```

[1] 5 7 9

```
# without vectorization
> a <- c(1,2,3)
> b <- c(4,5,6)
> c <- numeric(length(c))
for(i in seq_along(c)) {
        c[i] <- a[i] + b[i]
}
c
```

[1] 5 7 9

dplyr package

Tidyverse

Packages

dplyr

dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges. Go to docs...

https://www.tidyverse.org/packages/

dplyr tools

- **filter( )** - choose observation by values
- **arrange( )** - reorder rows
- **select( )** - choose variables by names
- **mutate( )** - create new variables of existing variables
- **summarize( )** - collapse several values to a single number

filter( )

- subset **rows** based on their values

```
> patient
```

```
# A tibble: 5 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
4 P004       male          3      13.2        95
5 P005       male          4      15.9       104
```

```
> filter(patient, sex == "female")
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
```

%in% operator

- x %in% y
- select every row where x is one of the values in y

```
> filter(patient, age_year %in% c(1, 4))
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P005       male          4      15.9       104
```

arrange( )

- similar to filter() except that it changes the order of **rows**
- inputs are data frame and a set of column names to order by

```
> arrange(patient, height_cm)
```

```
# A tibble: 5 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P003       female        2      10.5        85
3 P004       male          3      13.2        95
4 P002       female        4      16.4        96
5 P005       male          4      15.9       104
```

arrange( )

- use **desc( )** to reorder in a decreasing manner

```
> arrange(patient, desc(height_cm))
```

patient_ID	sex	age_year	weight_kg	height_cm
<chr>	<chr>	<dbl>	<dbl>	<dbl>
1 P005	male	4	15.9	104
2 P002	female	4	16.4	96
3 P004	male	3	13.2	95
4 P003	female	2	10.5	85
5 P001	female	1	9.1	73

slice( )

- select **rows** based on the location in the data frame

```
> slice(patient, 1:3)
```

```
# A tibble: 3 x 5
  patient_ID sex      age_year  weight_kg  height_cm
  <chr>      <chr>       <dbl>      <dbl>      <dbl>
1 P001       female          1        9.1         73
2 P002       female          4       16.4         96
3 P003       female          2       10.5         85
```

select( )

- subset using operations based on **column** names

```
> select(patient, patient_ID, sex, age_year)
```

```
# A tibble: 5 x 3
  patient_ID sex    age_year
  <chr>      <chr>  <dbl>
1 P001       female 1
2 P002       female 4
3 P003       female 2
4 P004       male   3
5 P005       male   4
```

mutate( )

- to add new **columns** that are functions of existing columns

```
> mutate(patient, age_month = age_year * 12)
```

```
# A tibble: 5 x 6
  patient_ID sex   age_year weight_kg height_cm age_month
  <chr>      <chr>    <dbl>     <dbl>     <dbl>     <dbl>
1 P001       female       1       9.1        73        12
2 P002       female       4      16.4        96        48
3 P003       female       2      10.5        85        24
4 P004       male         3      13.2        95        36
5 P005       male         4      15.9       104        48
```

summarize( )

- collapses a data frame to a single row

helper tool

- **group_by( )** - changes the scope of a function from operating on the entire dataset to operating on a certain group

```
> group <−   group_by(patient, sex)
> summarize(group, mean(weight_kg))
```

```
# A tibble: 2 x 2
  sex  `mean(weight_kg)`
  <chr>              <dbl>
1 female             12
2 male               14.6
```

Tidy data

```
> population_data
```

```
# A tibble: 19,747 x 3
   `Country Name`  year population
   <chr>          <dbl>      <dbl>
 1 Afghanistan     1960    8996973
 2 Afghanistan     1961    9169410
 3 Afghanistan     1962    9351441
 4 Afghanistan     1963    9543205
 5 Afghanistan     1964    9744781
 6 Afghanistan     1965    9956320
 7 Afghanistan     1966   10174836
 8 Afghanistan     1967   10399926
 9 Afghanistan     1968   10637063
10 Afghanistan     1969   10893776
# … with 19,737 more rows
```

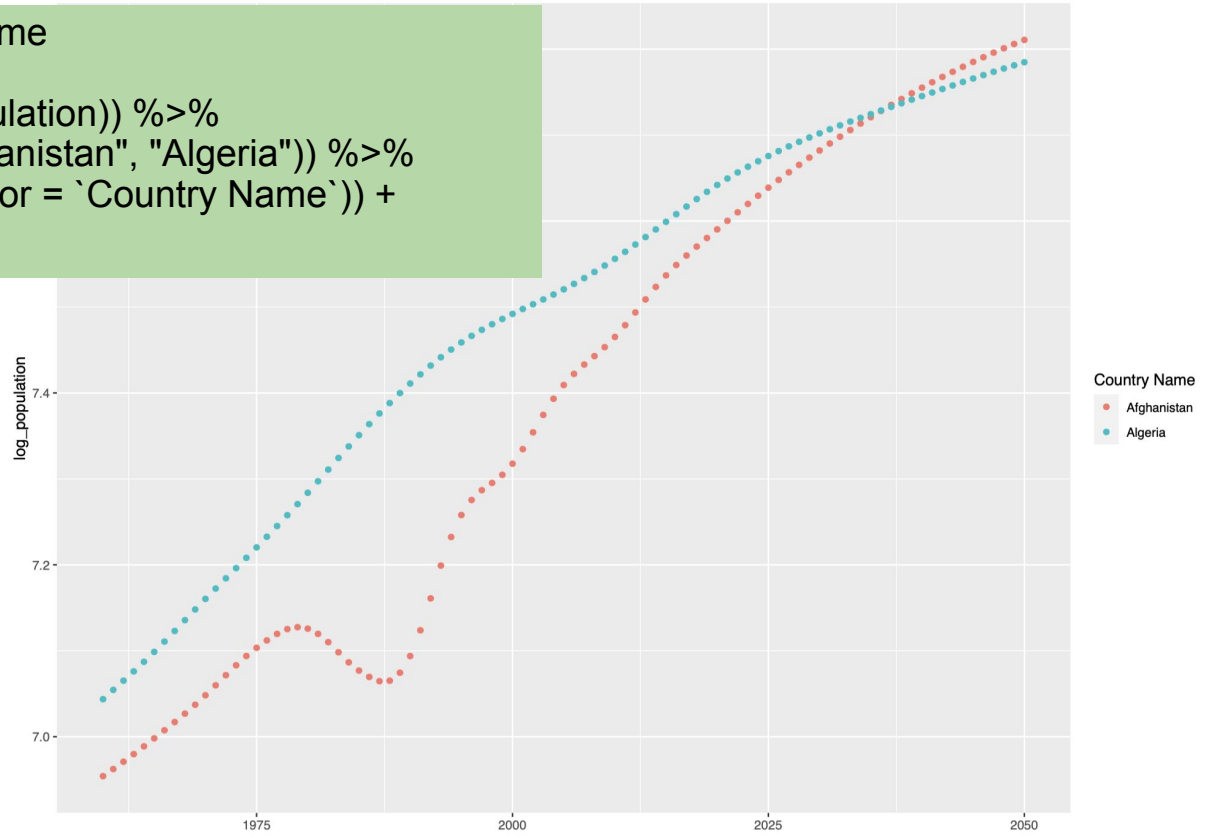
Examples of tidy data functionality

```
# Compute log population
> population_data %>%
      mutate(log_population = log10(population))
```

```
# A tibble: 19,747 x 4
   `Country Name`  year population log_population
   <chr>           <dbl>      <dbl>          <dbl>
 1 Afghanistan     1960    8996973           6.95
 2 Afghanistan     1961    9169410           6.96
 3 Afghanistan     1962    9351441           6.97
 4 Afghanistan     1963    9543205           6.98
 5 Afghanistan     1964    9744781           6.99
 6 Afghanistan     1965    9956320           7.00
 7 Afghanistan     1966   10174836           7.01
 8 Afghanistan     1967   10399926           7.02
 9 Afghanistan     1968   10637063           7.03
10 Afghanistan     1969   10893776           7.04
# … with 19,737 more rows
```

Examples of tidy data functionality

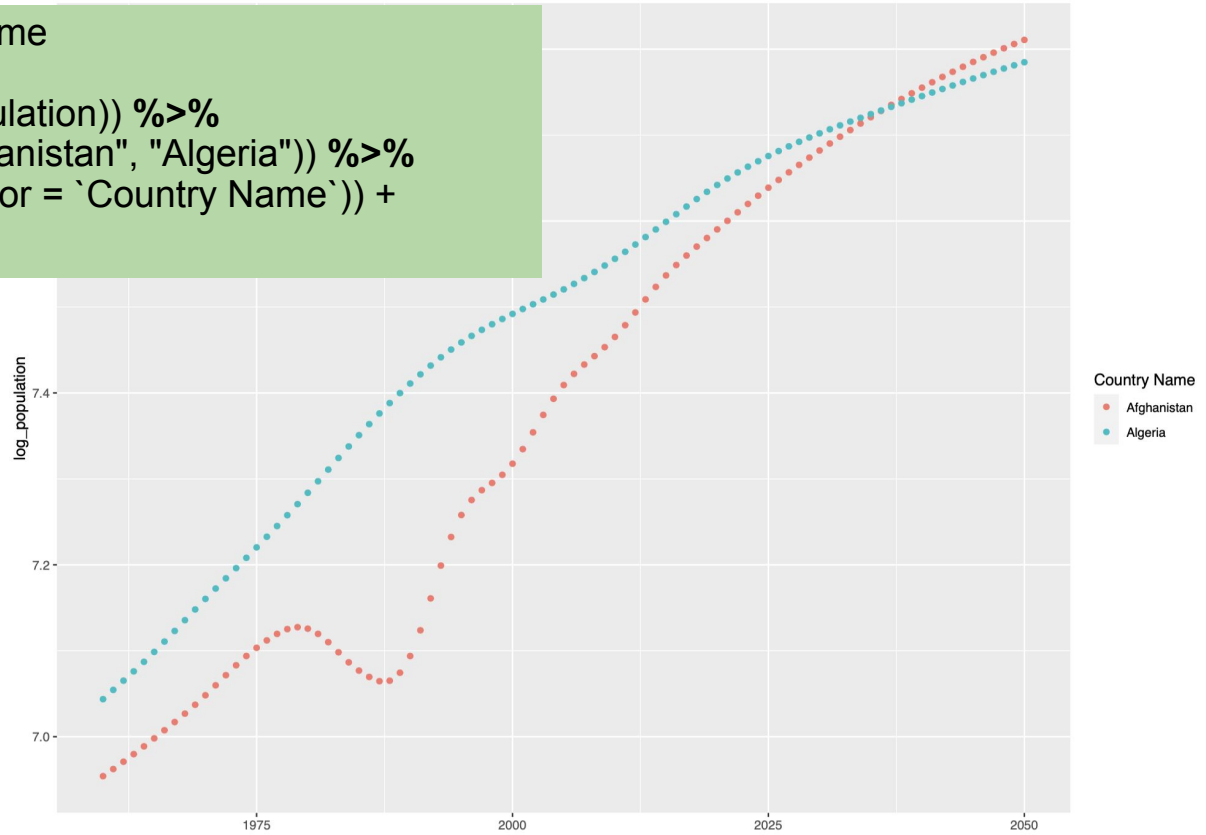
```
# Visualize population change over time
> population_data %>%
  mutate(log_population = log10(population)) %>%
  filter(`Country Name` %in% c("Afghanistan", "Algeria")) %>%
  ggplot(aes(year, log_population, color = `Country Name`)) +
  geom_point()
```



Pipe operator %>%

```
# Visualize population change over time
> population_data %>%
  mutate(log_population = log10(population)) %>%
  filter(`Country Name` %in% c("Afghanistan", "Algeria")) %>%
  ggplot(aes(year, log_population, color = `Country Name`)) +
  geom_point()
```

- useful when coding multiple operations in a succession



Pivoting

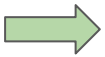
- most data will be untidy
- most data is organized to facilitate data entry without consideration for downstream analysis
- need to figure out which are variable and observations
- need to reshape data to be tidy

pivot_longer( )

- a common problem is a dataset where some of the column names are not names of variables, but values of a variable
- makes data frames narrower and longer

```
> data_1
```

```
# A tibble: 3 x 3
  `Country/Region` `2020-01-22` `2021-01-22`
  <chr>                   <dbl>         <dbl>
1 China                     548         98886
2 Japan                       2        357174
3 Philippines                 0        505939
```

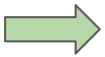


```
# A tibble: 6 x 3
  `Country/Region` date        cases
  <chr>            <date>      <dbl>
1 China            2020-01-22    548
2 China            2021-01-22  98886
3 Japan            2020-01-22      2
4 Japan            2021-01-22 357174
5 Philippines      2020-01-22      0
6 Philippines      2021-01-22 505939
```

pivot_longer( )

```
> data_1 %>%
      pivot_longer(cols = c(`2020-01-22`, `2021-01-22`),
            names_to = "date",
            values_to = "cases")
```

```
# A tibble: 3 x 3
  `Country/Region` `2020-01-22` `2021-01-22`
  <chr>                   <dbl>         <dbl>
1 China                     548         98886
2 Japan                       2        357174
3 Philippines                 0        505939
```



```
# A tibble: 6 x 3
  `Country/Region` date       cases
  <chr>            <date>     <dbl>
1 China            2020-01-22   548
2 China            2021-01-22 98886
3 Japan            2020-01-22     2
4 Japan            2021-01-22 357174
5 Philippines      2020-01-22     0
6 Philippines      2021-01-22 505939
```

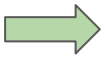
pivot_wider( )

- use it when an observation is spread across multiple rows
- makes data frames shorter and wider

```
> data_2
```

A tibble: 12 x 4

	`Country/Region`	date	type	count
	<chr>	<date>	<chr>	<dbl>
1	China	2020-01-22	cases	548
2	China	2020-01-22	population	1433783692
3	China	2021-01-22	cases	98886
4	China	2021-01-22	population	1439323774
5	Japan	2020-01-22	cases	2
6	Japan	2020-01-22	population	126860299
7	Japan	2021-01-22	cases	357174
8	Japan	2021-01-22	population	126476458
9	Philippines	2020-01-22	cases	0
10	Philippines	2020-01-22	population	108116622
11	Philippines	2021-01-22	cases	505939
12	Philippines	2021-01-22	population	109581085



A tibble: 6 x 4

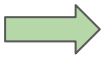
	`Country/Region`	date	cases	population
	<chr>	<date>	<dbl>	<dbl>
1	China	2020-01-22	548	1433783692
2	China	2021-01-22	98886	1439323774
3	Japan	2020-01-22	2	126860299
4	Japan	2021-01-22	357174	126476458
5	Philippines	2020-01-22	0	108116622
6	Philippines	2021-01-22	505939	109581085

pivot_wider( )

```
> data_2 %>%
      pivot_wider(names_from = type,
            values_from = count)
```

A tibble: 12 x 4

`Country/Region`	date	type	count
<chr>	<date>	<chr>	<dbl>
1 China	2020-01-22	cases	548
2 China	2020-01-22	population	1433783692
3 China	2021-01-22	cases	98886
4 China	2021-01-22	population	1439323774
5 Japan	2020-01-22	cases	2
6 Japan	2020-01-22	population	126860299
7 Japan	2021-01-22	cases	357174
8 Japan	2021-01-22	population	126476458
9 Philippines	2020-01-22	cases	0
10 Philippines	2020-01-22	population	108116622
11 Philippines	2021-01-22	cases	505939
12 Philippines	2021-01-22	population	109581085



A tibble: 6 x 4

`Country/Region`	date	cases	population
<chr>	<date>	<dbl>	<dbl>
1 China	2020-01-22	548	1433783692
2 China	2021-01-22	98886	1439323774
3 Japan	2020-01-22	2	126860299
4 Japan	2021-01-22	357174	126476458
5 Philippines	2020-01-22	0	108116622
6 Philippines	2021-01-22	505939	109581085

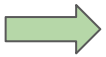
separate( )

- use when a column contains multiple variables
- pulls apart one column into multiple columns, by splitting wherever a separator character appears

```
> data_3
```

A tibble: 6 x 3

	`Country/Region`	date	rate
	<chr>	<date>	<chr>
1	China	2020-01-22	548/1433783692
2	China	2021-01-22	98886/1439323774
3	Japan	2020-01-22	2/126860299
4	Japan	2021-01-22	357174/126476458
5	Philippines	2020-01-22	0/108116622
6	Philippines	2021-01-22	505939/109581085



A tibble: 6 x 4

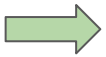
	`Country/Region`	date	cases	population
	<chr>	<date>	<dbl>	<dbl>
1	China	2020-01-22	548	1433783692
2	China	2021-01-22	98886	1439323774
3	Japan	2020-01-22	2	126860299
4	Japan	2021-01-22	357174	126476458
5	Philippines	2020-01-22	0	108116622
6	Philippines	2021-01-22	505939	109581085

separate( )

```
> data_3 %>%
      separate(col = rate,
                  into = c("cases", "population"))
```

A tibble: 6 x 3

`Country/Region`	date	rate
<chr>	<date>	<chr>
1 China	2020-01-22	548/1433783692
2 China	2021-01-22	98886/1439323774
3 Japan	2020-01-22	2/126860299
4 Japan	2021-01-22	357174/126476458
5 Philippines	2020-01-22	0/108116622
6 Philippines	2021-01-22	505939/109581085



A tibble: 6 x 4

`Country/Region`	date	cases	population
<chr>	<date>	<dbl>	<dbl>
1 China	2020-01-22	548	1433783692
2 China	2021-01-22	98886	1439323774
3 Japan	2020-01-22	2	126860299
4 Japan	2021-01-22	357174	126476458
5 Philippines	2020-01-22	0	108116622
6 Philippines	2021-01-22	505939	109581085

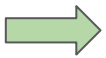
unite( )

- opposite of separate( )
- combines several columns into one column

```
> data_4
```

A tibble: 6 x 5

`Country/Region`	year	month	day	cases
<chr>	<chr>	<chr>	<chr>	<dbl>
1 China	2020	01	22	548
2 China	2021	01	22	98886
3 Japan	2020	01	22	2
4 Japan	2021	01	22	357174
5 Philippines	2020	01	22	0
6 Philippines	2021	01	22	505939



A tibble: 6 x 3

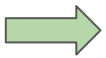
`Country/Region`	date	cases
<chr>	<chr>	<dbl>
1 China	2020-01-22	548
2 China	2021-01-22	98886
3 Japan	2020-01-22	2
4 Japan	2021-01-22	357174
5 Philippines	2020-01-22	0
6 Philippines	2021-01-22	505939

unite( )

```
> data_4 %>%
      unite(col = date,
            year, month, day,
            sep = "-")
```

A tibble: 6 x 5

`Country/Region`	year	month	day	cases
<chr>	<chr>	<chr>	<chr>	<dbl>
1 China	2020	01	22	548
2 China	2021	01	22	98886
3 Japan	2020	01	22	2
4 Japan	2021	01	22	357174
5 Philippines	2020	01	22	0
6 Philippines	2021	01	22	505939



A tibble: 6 x 3

`Country/Region`	date	cases
<chr>	<chr>	<dbl>
1 China	2020-01-22	548
2 China	2021-01-22	98886
3 Japan	2020-01-22	2
4 Japan	2021-01-22	357174
5 Philippines	2020-01-22	0
6 Philippines	2021-01-22	505939

Take-away message

- Tidying data is necessary before doing any analyses and visualizations.