London Business School

Session 4:
Tidying data
Confidence Intervals

Kostis Christodoulou
London Business School

London Business School

Tidy data (1/3)

The tables below show the same values of four variables **country, year, cases, population,** but each dataset organises the values in a different way.

```
> table1
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

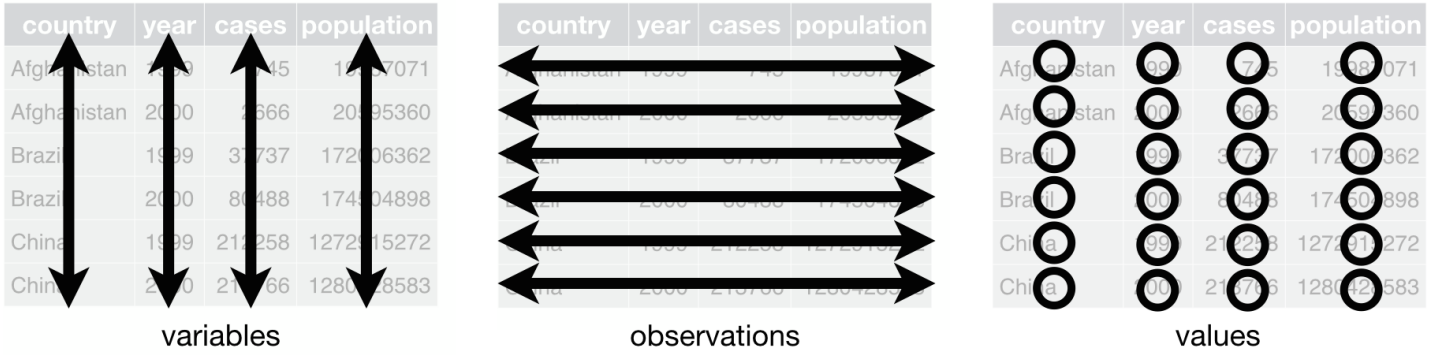
```
> table3
# A tibble: 6 x 3
  country      year rate
* <chr>       <int> <chr>
1 Afghanistan  1999 745/19987071
2 Afghanistan  2000 2666/20595360
3 Brazil       1999 37737/172006362
4 Brazil       2000 80488/174504898
5 China        1999 212258/1272915272
6 China        2000 213766/1280428583
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
* <chr>        <int>  <int>
1 Afghanistan    745   2666
2 Brazil       37737  80488
3 China       212258 213766
> table4b
# A tibble: 3 x 3
  country         `1999`     `2000`
* <chr>            <int>      <int>
1 Afghanistan   19987071   20595360
2 Brazil       172006362  174504898
3 China       1272915272 1280428583
```

Tidy data 2/3

Tidy data is a specific way of organising data to facilitate analysis with the tidyverse. The tidy data standard has been designed to facilitate exploratory data analysis; tidy datasets and tidy tools help make data analysis easier, allowing you to focus on the interesting domain problem, not on the logistics of cleaning data.

There are three rules which make a dataset tidy:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.



We often need to reshape our datasets and should have a way to go:

- from wide format to long (tidy) format using *gather()* or *pivot_longer()*
- from long (tidy) to wide format using *spread()* or *pivot_wider()*

London Business School

Tidy data 3/3

Five main ways data tend not to be tidy:

1. *Column headers are values, not variable names*.
2. *Multiple variables are stored in one column*.
3. *Variables are stored in both rows and columns*.
4. *Multiple types of observational units are stored in the same table*.
5. *A single observational unit is stored in multiple tables*.

```
> table1
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

```
> table3
# A tibble: 6 x 3
  country      year rate
* <chr>       <int> <chr>
1 Afghanistan  1999 745/19987071
2 Afghanistan  2000 2666/20595360
3 Brazil       1999 37737/172006362
4 Brazil       2000 80488/174504898
5 China        1999 212258/1272915272
6 China        2000 213766/1280428583
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
* <chr>        <int>  <int>
1 Afghanistan    745   2666
2 Brazil       37737  80488
3 China       212258 213766
> table4b
# A tibble: 3 x 3
  country         `1999`     `2000`
* <chr>            <int>      <int>
1 Afghanistan   19987071   20595360
2 Brazil       172006362  174504898
3 China       1272915272 1280428583
```

London Business School

Why tidy? (and not dirty/untidy/messy)

1. There's a general advantage to picking one consistent way of storing data.

2. Having variables in columns allows to exploit R's vectorised nature, as most built-in R functions work with vectors of values. That makes transforming tidy data feel particularly natural.

```
# The easiest way to get tidyr is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just tidyr:
install.packages("tidyr")

# Or the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/tidyr")
```

Use the development version of ***tidyr*** to use ***pivot_longer()*** and ***pivot_wider()***

London Business School

pivot_longer() or *gather()*

- A common problem is when column names are not **names** of variables, but **values** of a variable.
- Column names **1999** and **2000** represent values of the year variable, and each row represents two observations, not one.

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

We need to gather those columns into a new pair of variables. We need to map the names and the values.

1. The set of columns that represent values, not variables. In this example, those are the columns ***1999*** and ***2000***.
2. The name, or key, of the variable whose values will form the column names: **year.**
3. The name of the variable whose values are spread over the cells: ***cases***

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

table4a %>%
  gather(`1999`, `2000`, key = "year", value = "cases")
```

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

London Business School

Wide and long data (1/3)

There are three rules which make a dataframe tidy:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

Is this data in tidy format? What are the variables and observations?

student_id	final_exam	midterm	group_project
2457625	79	68	71
1758293	92	73	67
1622247	71	87	74

Work with your neighbour and sketch out on a piece of paper what this data would look like in tidy format. Some hints:

1. What are the variables?
2. What variable pairings are of interest?
3. Is each observation in its own row ?

Wide and long data (3/3)

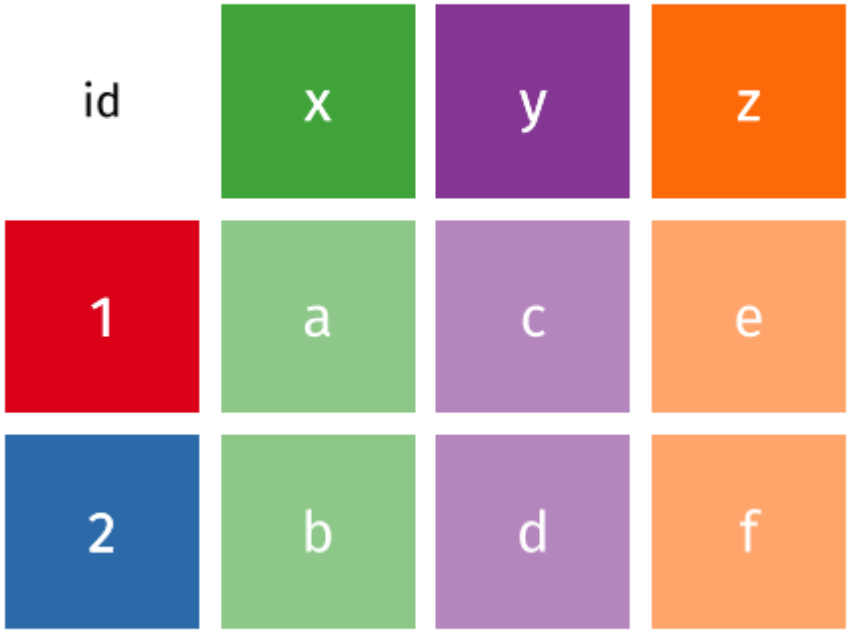
We often need to reshape our datasets and we have a way to go:

- from wide format to long (tidy) format using ***pivot_longer()*** or ***gather()***
- from long (tidy) to wide format using ***pivot_wider()*** or ***spread()***

```
pivot_longer(cols=c('final_exam', 'midterm', 'group_project'), names_to = "assignment", values_to = "score")
```

```
gather('final_exam', 'midterm', 'group_project', key = assignment, value = score)
```

wide

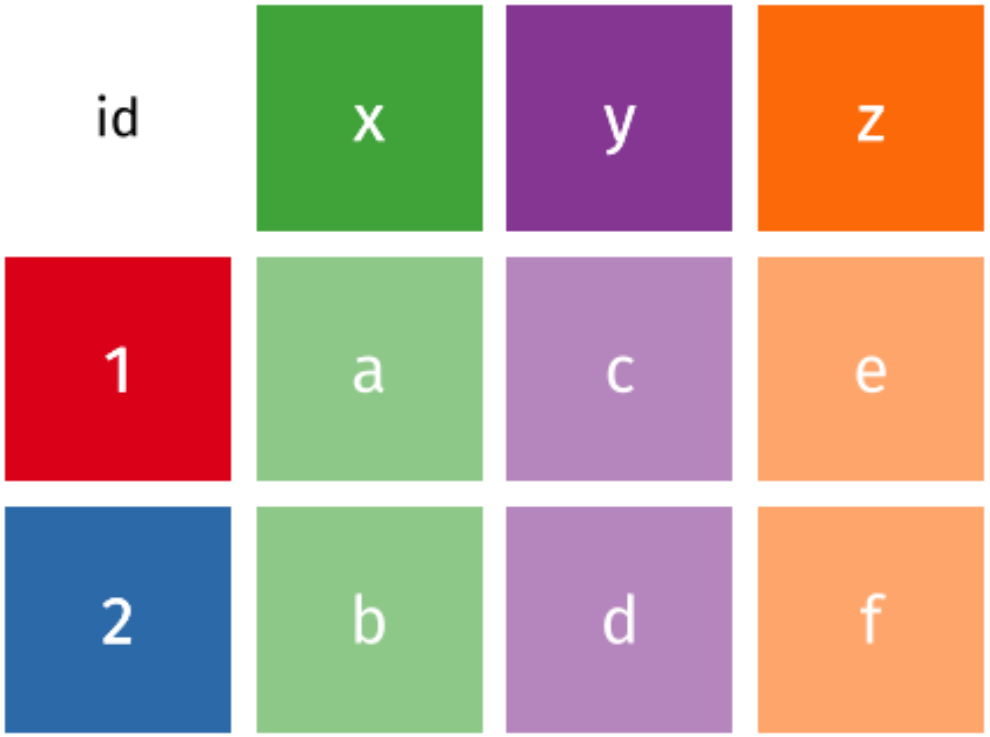


https://github.com/gadenbuie/tidyexplain#tidy-d

London Business School

wide

Tidying data




```
pivot_longer(cols=c('final_exam', 'midterm', 'group_project'), names_to = "assignment", values_to = "score")
gather('final_exam', 'midterm', 'group_project', key = assignment, value = score)
pivot_wider(names_from = assignment, values_from = score)
spread(key = assignment, value = score)
```


https://github.com/gadenbuie/tidyexplain#tidy-data

London Business School

Worked Examples: Live Coding