Lecture 15: Tidy Data

STAT 450, Fall 2021

Reading: Sections 12.1–12.3 from *R for Data Science*

Reference: https://tidyr.tidyverse.org/

Today we discuss a consistent way to organize your data in R called *tidy data*.

A tidy data set has the following properties:

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

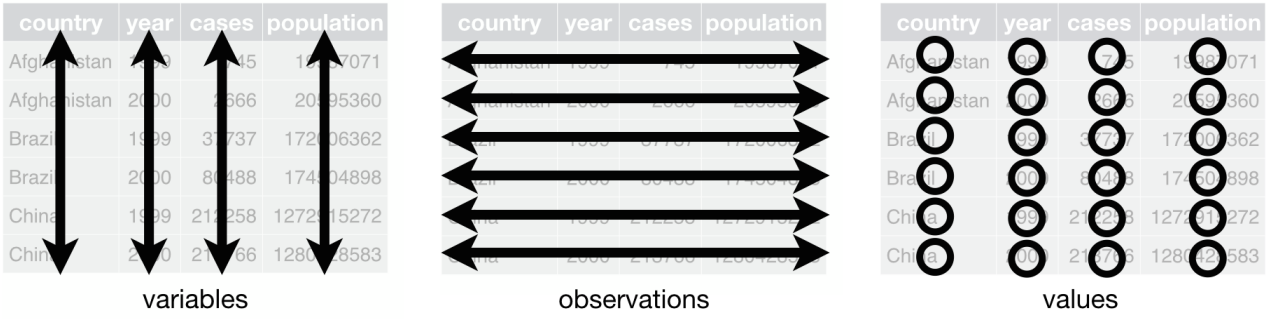
It can take some work to clean your data and make it tidy. But the pay off for this work is that you will be able to most effectively use all the tools provided in the tidyverse for data wrangling and visualization.

The following is an example of a tidy data set:

```
library(tidyverse)
```

```
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
```



Many of the data sets that you encounter will probably not be tidy. There are two main reasons:

1. Many people are not aware of best practices for organizing data frames.
2. Data may be organized for some use other than analysis.

For this lecture, we will go over some examples of data sets that are not tidy, and demonstrate how to make them tidy.

Example: `pivot_wider()`

Consider the following data set, which is not tidy since each observation is spread across two rows.

```
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
```

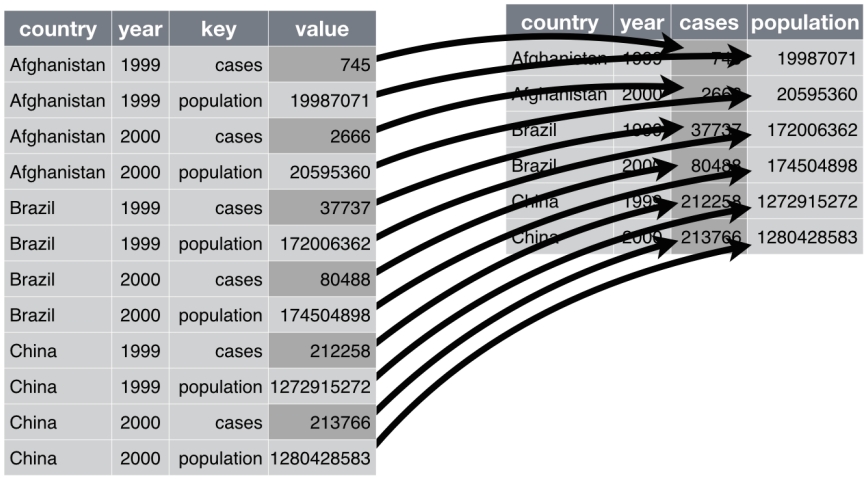
To make this a tidy data set, we need to put `cases` and `population` into two separate columns. To do this we can use the function `pivot_wider()`:

```
table2 %>%
  pivot_wider(names_from = type, values_from = count)
```

```
## # 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
```

Here we specify two arguments for `pivot_wider()`:

- `names_from`: the column to take variable names from
- `values_from`: the column to take values from



country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Exercise: Use `pivot_wider()` to make the following data set into a tidy data set:

```
people <- tibble(
  name = c("John", "John", "Mary", "Mary", "Robert", "Robert"),
  X1 = c("gender", "age", "gender", "age", "gender", "age"),
  X2 = c("m", "29", "f", "36", "m", "40")
)
people
```

```
## # A tibble: 6 x 3
##   name   X1     X2
##   <chr>  <chr>  <chr>
## 1 John   gender m
## 2 John   age    29
## 3 Mary   gender f
## 4 Mary   age    36
## 5 Robert gender m
## 6 Robert age    40
```

Example: `pivot_longer()`

A common problem is a data set where some of the column names are not names of variables, but values of a variable. For example:

```
table4a
```

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

Here the column names `1999` and `2000` represent values of the `year` variable, and the entries in those columns represent values of the `cases` variable.

To make this data set tidy we can use the function `pivot_longer()`:

```
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
```

Here we specify three arguments for `pivot_longer()`:

- The set of columns whose names are values, not variables
- `names_to`: the name of the variable to move the column names to
- `values_to`: the name of the variable to move the column values to

Note that `1999` and `2000` are non-syntactic names (because they don't start with a letter) so we have to surround them in backticks.

Exercise: Use `pivot_longer()` to make the following data set into a tidy data set:

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
## * <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```