Tidy Data and Tidying Data

David Gerard

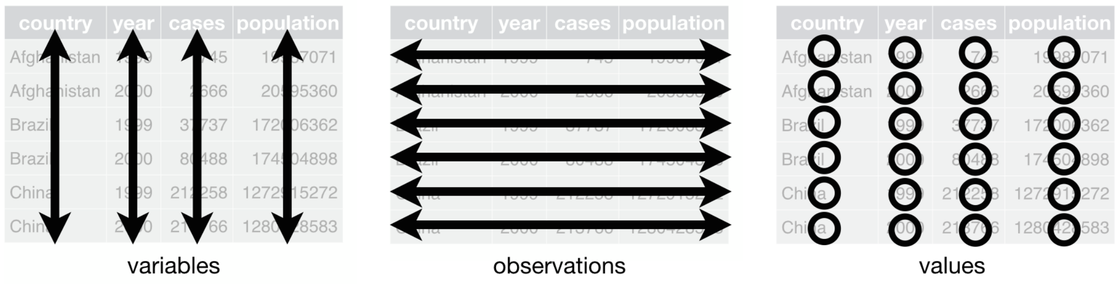
2019-02-13

Learning Objectives

- What is tidy data?
- Learn to make your data tidy with `gather()`, `spread()`, `separate()`, and `unite()`.
- Chapter 12 of RDS
- Data Import Cheat Sheet

Tidy Data

- Recall:
    - Observations/units/subjects/individuals/cases: objects described by a set of data (e.g. cars, people, countries).
    - Variable: describes some characteristic of the units (e.g. mpg, age, GDP).
    - Each unit has a single value of each variable (e.g. 20 mpg, 31 years old, $20,513,000US$million).
- Tidy Data:
    - One unit per row.
    - One variable per column.
    - One value per cell.
- Hadley's visualization:



- We will use the tidyr package (a member of the tidyverse) to make data tidy.

```r
library(tidyverse)
```

- Example of tidy data:

```r
tidyr::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
```

  - Variables: Country, Year, Cases, Population
  - Units: location×time

- Untidy data: Each unit is spread across multiple rows

```r
print(tidyr::table2, n = 12)
```

```
## # 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
```

- Untidy data: Two variables are in one column

```r
tidyr::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
```

- Untidy data: Data are spread across two data frames. Within each data frame, multiple units are in one row.

```r
tidyr::table4a
```

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

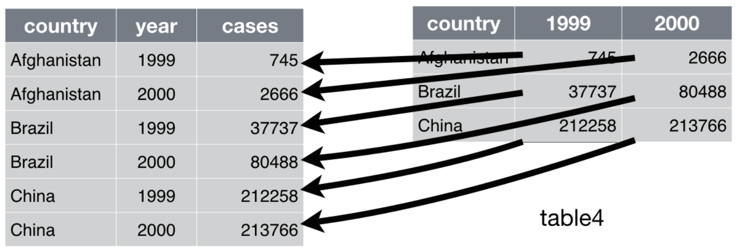
```
tidyr::table4b
```

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

- Sometimes it is easy to determine the units and the variables.
- Sometimes it is very hard and you need to talk to the data collectors to find out.
- We want tidy data because R easily manipulates vectors. So in the long run it will make your life easier to first make data tidy.

Gather

- Problem: One variable spread across multiple columns.
- Column names are actually *values* of a variable
- `table4a` and `table4b`
- Solution: `gather()`
- Hadley's visualization:



- Specify

  i. The columns that are values, not variables,
  ii. The name of the variable that will take the values of the column names (`key`), and
  iii. The name of the variable that will take the values spread in the cells (`value`).

```
tidyr::table4a %>%
  gather(`1999`, `2000`, key = "Year", value = "cases") ->
  tidy4a
tidy4a
```

```
## # 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
```

```
tidyr::table4b %>%
  gather(`1999`, `2000`, key = "Year", value = "population") ->
  tidy4b
tidy4b
```

```
## # A tibble: 6 x 3
##   country     Year  population
##   <chr>       <chr>      <int>
## 1 Afghanistan 1999    19987071
## 2 Brazil      1999   172006362
## 3 China       1999  1272915272
## 4 Afghanistan 2000    20595360
## 5 Brazil      2000   174504898
## 6 China       2000  1280428583
```

- We will learn next class how to join these two data frames next week. But the code is

```
full_join(tidy4a, tidy4b)
```

```
## Joining, by = c("country", "Year")
```

```
## # A tibble: 6 x 4
##   country     Year   cases population
##   <chr>       <chr>  <int>      <int>
## 1 Afghanistan 1999     745   19987071
## 2 Brazil      1999   37737  172006362
## 3 China       1999  212258 1272915272
## 4 Afghanistan 2000    2666   20595360
## 5 Brazil      2000   80488  174504898
## 6 China       2000  213766 1280428583
```

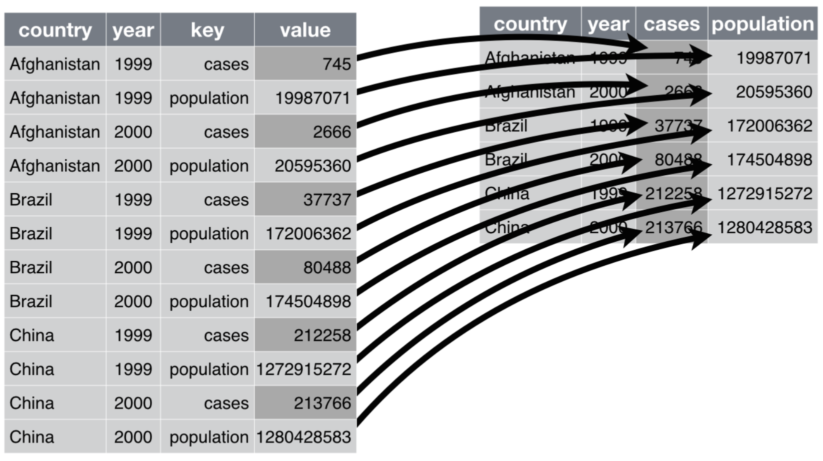
- **Exercise**: gather the `monkeymem` data frame (available at https://dcgerard.github.io/stat_412_612/data/monkeymem.csv). The cell values represent identification accuracy of some objects (in percent of 20 trials).
- **Exercise** (RDS 12.3.3.1): Why does this code fail?

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

```
## Error in inds_combine(.vars, ind_list): Position must be between 0 and n
```

Spread

- Problem: One observation is spread across multiple rows.
- One column contains variable names. One column contains values for the different variables.
- `table2`
- Solution: `spread()`
- Hadley's visualization:



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

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

table2

- Specify:
  i. The column that contains the column names (`key`), and
  ii. The column that contains the values (`value`).

```r
table2 %>%
  spread(key = type, value = count)
```

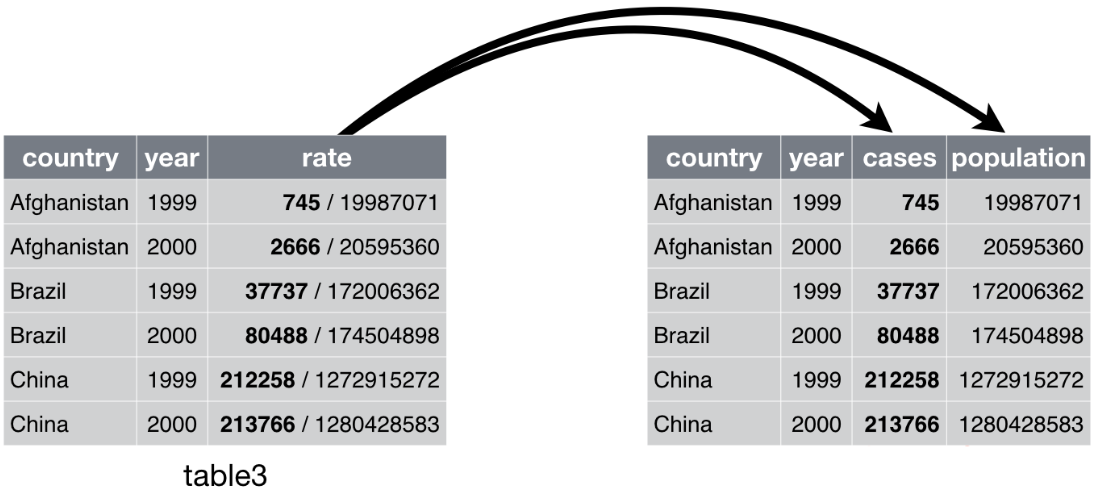
```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258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

- **Exercise**: Spread the `flowers1` data frame (available at https://dcgerard.github.io/stat_412_612/data/flowers1.csv).
- **Exercise** (RDS 13.3.3.3): Why does spreading this data frame fail?

```
people <- tribble(
  ~name,             ~key,    ~value,
  #-----------------|--------|------
  "Phillip Woods",   "age",       45,
  "Phillip Woods",   "height",   186,
  "Phillip Woods",   "age",       50,
  "Jessica Cordero", "age",       37,
  "Jessica Cordero", "height",   156
)
```

Separate

- Problem: One column contains two (or more) variables.
- `table3`
- Solution: `separate()`
- Hadley's visualization:



country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

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

- Specify:

   i. The column that contains two (or more) variables,
   ii. A character vector of the new names of the variables, and
   iii. The character that separates variables (or the position that separates variables).

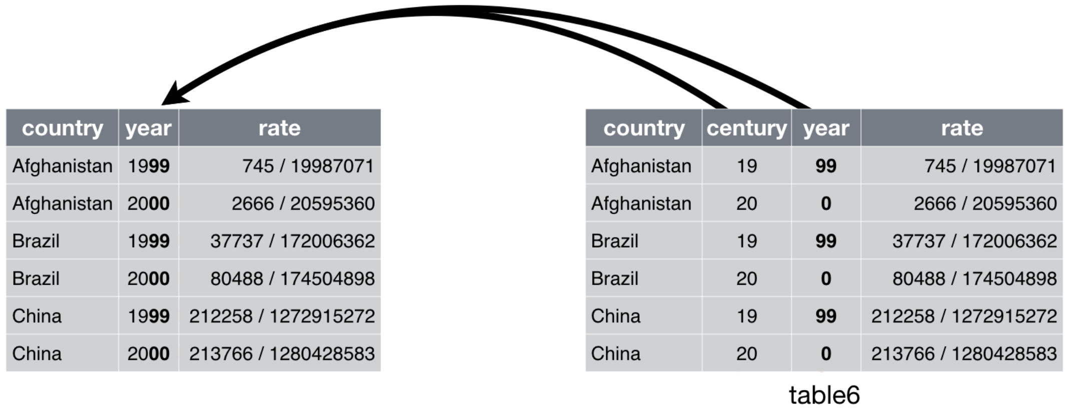
```
table3 %>%
  separate(rate, into = c("cases", "population"), sep = "/")
```

```
## # A tibble: 6 x 4
##   country      year cases  population
##   <chr>       <int> <chr>  <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
```

- **Exercise**: Separate the `flowers2` data frame (available at https://dcgerard.github.io/stat_412_612/data/flowers2.csv).

Unite

- Problem: One variable spread across multiple columns.
- Solution: `unite()`
- Hadley's visualization:



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

country	century	year	rate
Afghanistan	19	99	745 / 19987071
Afghanistan	20	0	2666 / 20595360
Brazil	19	99	37737 / 172006362
Brazil	20	0	80488 / 174504898
China	19	99	212258 / 1272915272
China	20	0	213766 / 1280428583

- Much less common problem.

```
table5
```

```
## # A tibble: 6 x 4
##   country     century year  rate
## * <chr>       <chr>   <chr> <chr>
## 1 Afghanistan 19      99    745/19987071
## 2 Afghanistan 20      00    2666/20595360
## 3 Brazil      19      99    37737/172006362
## 4 Brazil      20      00    80488/174504898
## 5 China       19      99    212258/1272915272
## 6 China       20      00    213766/1280428583
```

- Specify:

i. The name of the new column (`col`),
ii. The columns to unite, and
iii. The separator of the variables in the new column (`sep`).

```
table5 %>%
  unite(century, year, col = "Year", sep = "")
```

```
## # A tibble: 6 x 3
##   country     Year  rate
##   <chr>       <chr> <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
```

- **Exercise**: Re-unite the data frame you separated from the `flowers2` exercise. Use a comma for the separator.