Programming for Data Science

Tidy Data in R

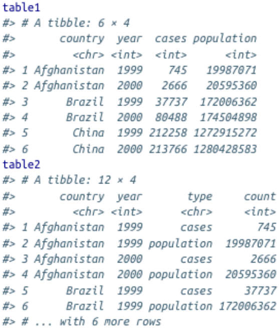
Marco Beccuti

Università degli Studi di Torino

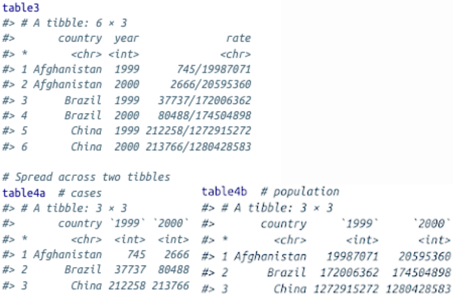
Dipartimento di Informatica

Tidy data

- You can represent the same underlying data in multiple ways;
- The following example shows the same data organized in four different ways;
- Each dataset shows the same values of four variables, country, year, cases, and population, but each dataset organizes the values in a different way:



```
table1
#> # A tibble: 6 × 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 × 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
#> # ... with 6 more rows
```

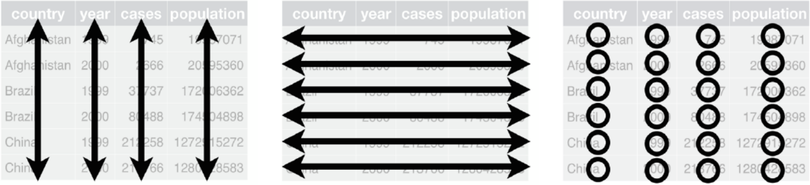


```
table3
#> # A tibble: 6 × 3
#>       country  year              rate
#> *       <chr> <int>             <chr>
#> 1 Afghanistan  1999      745/19987071
#> 2 Afghanistan  2000     2666/20595360
#> 3      Brazil  1999   37737/172006362
#> 4      Brazil  2000   80488/174504898
#> 5       China  1999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

# Spread across two tibbles
table4a  # cases                      table4b  # population
#> # A tibble: 3 × 3                 #> # A tibble: 3 × 3
#>       country `1999` `2000`       #>       country     `1999`     `2000`
#> *       <chr>  <int>  <int>       #> *       <chr>      <int>      <int>
#> 1 Afghanistan    745   2666       #> 1 Afghanistan   19987071   20595360
#> 2      Brazil  37737  80488       #> 2      Brazil  172006362  174504898
#> 3       China 212258 213766       #> 3       China 1272915272 1280428583
```

Tidy data

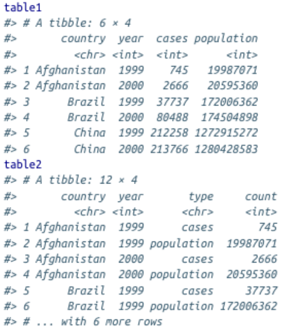
- There are three interrelated rules which make a dataset tidy:
  1. Each variable must have its own column;
  2. Each observation must have its own row;
  3. Each value must have its own cell.



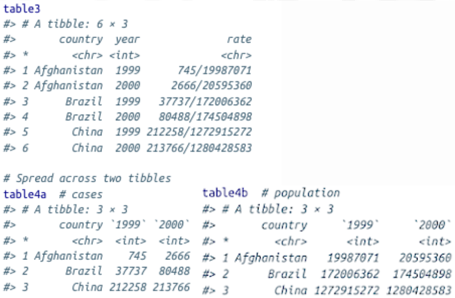
variables observations values

Tidy data

- In this example, only *table1* is tidy;
- It is the only representation where each column is a variable.



```
table1
#> # A tibble: 6 × 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 × 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
#> # ... with 6 more rows
```



```
table3
#> # A tibble: 6 × 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
```

```
# Spread across two tibbles
table4a  # cases
#> # A tibble: 3 × 3
#>       country `1999` `2000`
#> *       <chr>  <int>  <int>
#> 1 Afghanistan    745   2666
#> 2      Brazil  37737  80488
#> 3       China 212258 213766
```

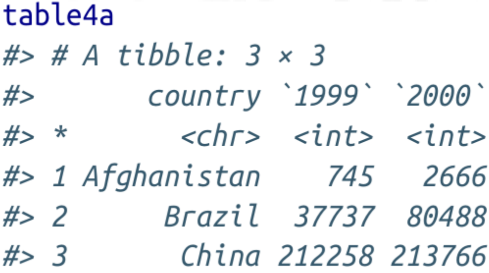
```
table4b  # population
#> # A tibble: 3 × 3
#>       country     `1999`     `2000`
#> *       <chr>      <int>      <int>
#> 1 Afghanistan   19987071   20595360
#> 2      Brazil  172006362  174504898
#> 3       China 1272915272 1280428583
```

Spreading and Gathering

- For most real analyses, you will need to do some tidying;
- The first step is always to figure out what the variables and observations are;
- The second step is to resolve one of two common problems:
  - One variable might be spread across multiple columns;
  - One observation might be scattered across multiple rows.
  - Multiple values might be stored in a same cells.
- in *tidyr* package the functions *gather()*, *spread()* and *separete()* can be exploited to fix these problems.

Gathering

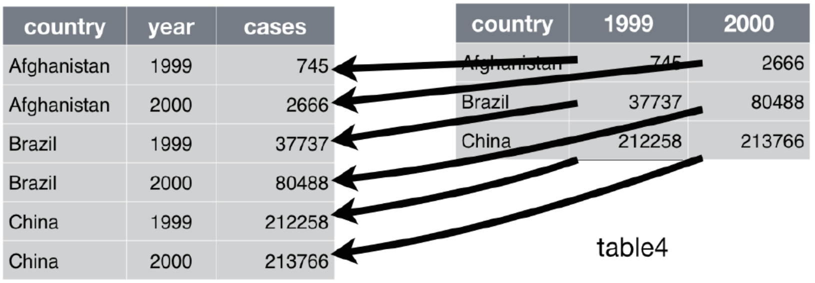
- A common problem is a dataset where some of the column names are not names of variables;
- Take *table4a* the column names *1999* and *2000* represent values of the year variable, and each row represents two observations;



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

- we need to collect these columns into a new pair of variables.

Gathering



country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

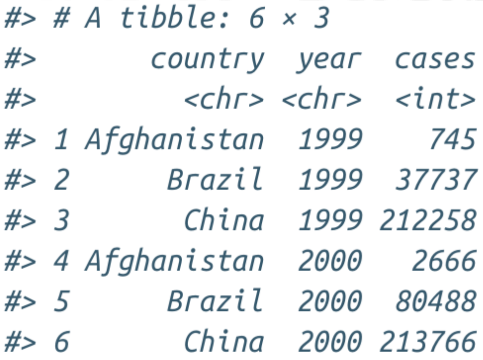
country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Gathering

- To achieve this task we have to specify three parameters:
  - The set of columns that represent values, not variables (i.e. *1999* and *2000*).
  - The name of the variable whose values form the column names (e.g. *year*).
  - The name of the variable whose values are spread over the cells (e.g. *cases*).

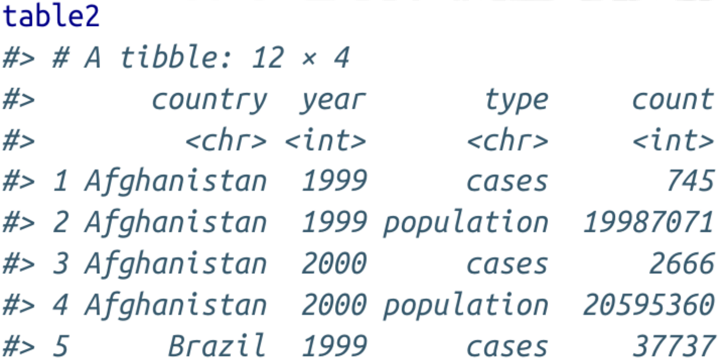
```
> gather(table4a, "1999", "2000", key = "year", value = "cases")
```



```
#> # A tibble: 6 × 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
```

Spreading

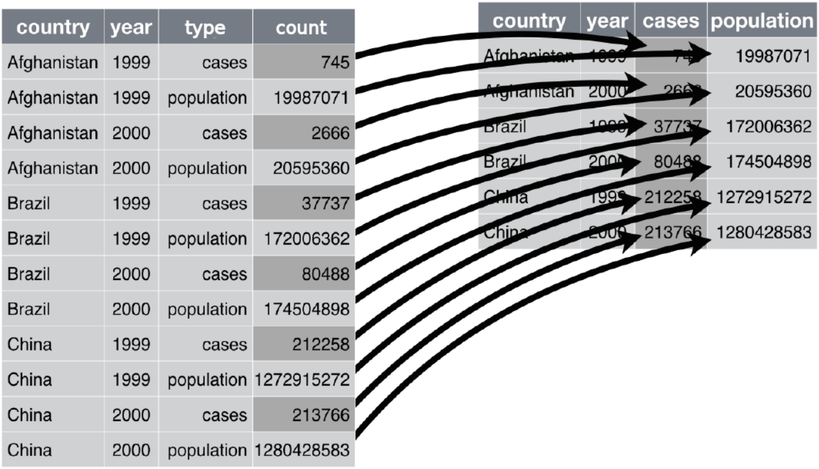
- Spreading is the opposite of gathering;
- It is used when an observation is scattered across multiple rows;
- For example, each observation is spread across two rows:



```
table2
#> # A tibble: 12 × 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
```

Spreading

- Spreading is the opposite of gathering;



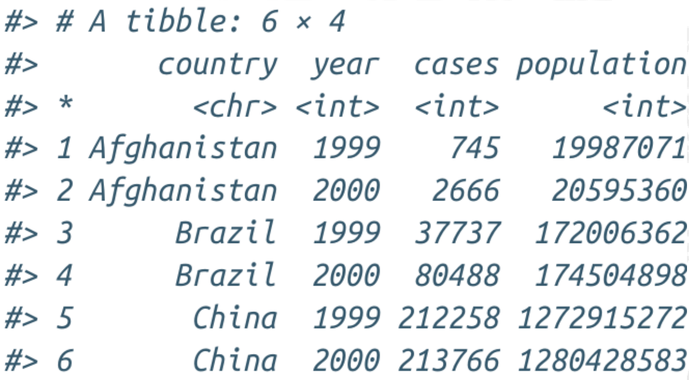
country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	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

Spreading

- Spreading is the opposite of gathering;

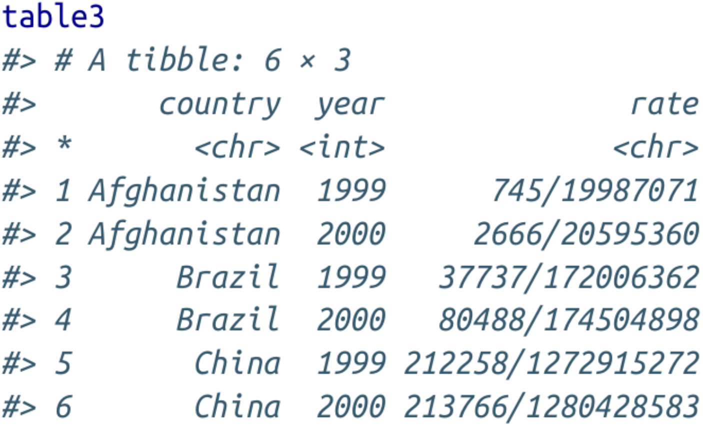
```
> spread(table2, key = type, value = count)
```



```
#> # A tibble: 6 × 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
```

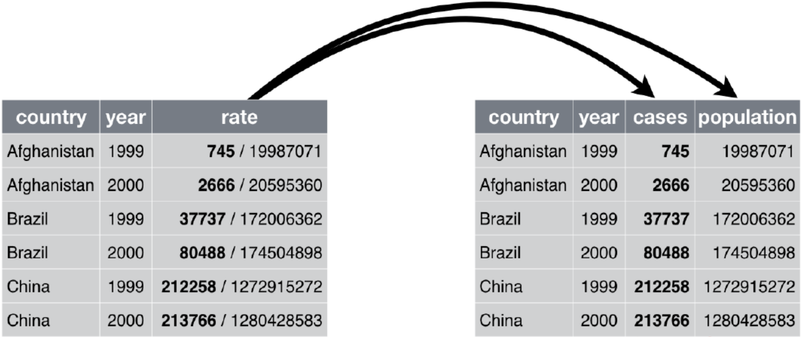
Separating

- it splits one column into multiple columns by separating wherever a separator character appears;
- For instance rate columns contains both cases then it must be split it into two variables:



```
table3
#> # A tibble: 6 × 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
```

Separating



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

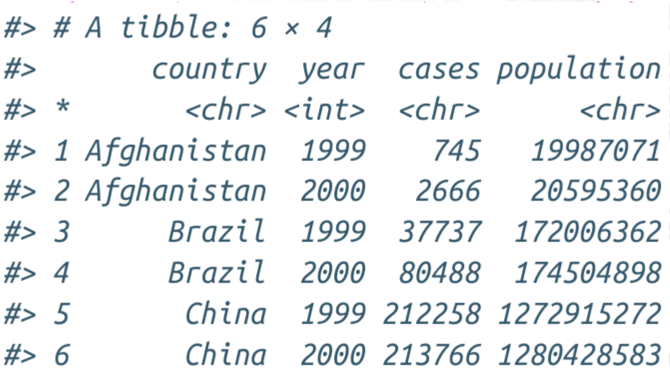
table3

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

Separating

- It splits one column into multiple columns by separating wherever a separator character appears;

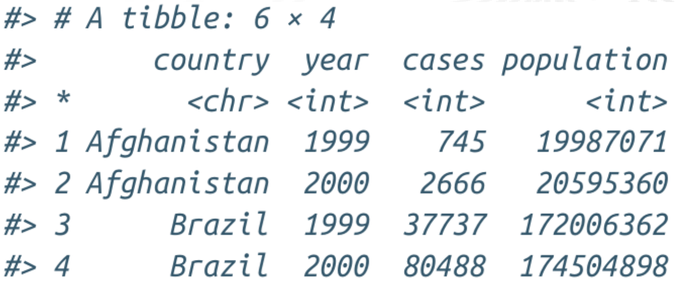
  > *separate*(*table*3, *rate*, *into* = *c*("*cases*", "*population*"), *sep* = "/")



```
#> # A tibble: 6 × 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
```

Separating

- It splits one column into multiple columns by separating wherever a separator character appears;
- We can ask **separate()** to try and convert to better types using convert = TRUE:
  > *separate(table3, rate, into = c(“cases”, “population”), sep = “/”, convert = TRUE)*



```
#> # A tibble: 6 × 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
```