Chapter9

Ayush Kumar Shah

9/23/2020

Tidying data

- tidyr package, member of tidyverse package.

Tidy data

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

untidy != messy data

Most data is untidy.

Two ways of becoming untidy:

- One variable might be spread across multiple columns. Solution - tidyr: gather()
- One observation might be scattered across multiple rows. Solution - tidyr: spread()

Gathering

```r
library(tidyverse)
```

```
## -- Attaching packages -------------------------------------------------------------------------------
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------ ti
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
table4a
```

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

Some of the column names are not names of variables, but values of a variable. The column names 1999 and 2000 represent values of the year variable, and each row represents two observations, not one.

```
tidy4a <- table4a %>%
gather(`1999`, `2000`, key = "year", value = "cases")
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
```

Same with table4b

```
table4b
```

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

Some of the column names are not names of variables, but values of a variable. The column names 1999 and 2000 represent values of the year variable, and each row represents two observations, not one.

Parameters:

- The set of columns that represent values, not variables. In this example, those are the columns 1999 and 2000.
- The name of the variable whose values form the column names. I call that the key, and here it is year.
- The name of the variable whose values are spread over the cells. I call that value, and here it's the number of cases.

```
tidy4b <- table4b %>%
gather(`1999`, `2000`, key = "year", value = "population")
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
```

Left join (by dplyr)

```
left_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
```

Spreading

```
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
```

When an observation is scattered across multiple rows. For example, take table2—an observation is a country in a year, but each observation is spread across two rows:

Parameters

- The column that contains variable names, the key column. Here, it's type.
- The column that contains values forms multiple variables, the value column. Here, it's count.

```
spread(table2, 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
```

Separating

```
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
```

one column (rate) that contains two variables (cases and population).

```
table3 %>%
separate(rate, into = c("cases", "population"), convert=TRUE)
```

```
## # 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
```

```
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
```

```
table3 %>%
separate(year, into = c("century", "year"), sep = 2, convert=TRUE)
```

```
## # A tibble: 6 x 4
##   country     century  year rate
##   <chr>         <int> <int> <chr>
## 1 Afghanistan      19    99 745/19987071
## 2 Afghanistan      20     0 2666/20595360
## 3 Brazil           19    99 37737/172006362
## 4 Brazil           20     0 80488/174504898
## 5 China            19    99 212258/1272915272
## 6 China            20     0 213766/1280428583
```

Unite

a single variable is spread across multiple columns.

```
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
```

```
table5 %>%
unite(new, century, year)
```

```
## # A tibble: 6 x 3
##   country     new   rate
##   <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
```

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

```
## # A tibble: 6 x 3
##   country     new   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
```

Missing values

- Explicitly (flagged with NA) - presence of an absence
- Implicitly (not present in the data) - absence of a presence

```
stocks <- tibble(
year = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
qtr = c( 1, 2, 3, 4, 2, 3, 4),
return = c(1.88, 0.59, 0.35, NA, 0.92, 0.17, 2.66)
)
stocks
```

```
## # A tibble: 7 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     2   0.92
## 6  2016     3   0.17
## 7  2016     4   2.66
```

There are two missing values in this dataset:

- The return for the fourth quarter of 2015 is explicitly missing, because the cell where its value should be instead contains NA.
- The return for the first quarter of 2016 is implicitly missing, because it simply does not appear in the dataset.

Making implicit missing values explicit

```
stocks %>%
spread(year, return)
```

```
## # A tibble: 4 x 3
##     qtr ‘2015‘ ‘2016‘
##   <dbl>  <dbl>  <dbl>
## 1     1   1.88  NA
## 2     2   0.59   0.92
## 3     3   0.35   0.17
## 4     4  NA      2.66
```

```
stocks %>%
complete(year, qtr)
```

Using complete

```
## # A tibble: 8 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     1  NA
## 6  2016     2   0.92
## 7  2016     3   0.17
## 8  2016     4   2.66
```

Making explicit missing values implicit

```
stocks %>%
spread(year, return) %>%
gather(year, return, `2015`:`2016`, na.rm = TRUE)
```

```
## # A tibble: 6 x 3
##     qtr year   return
##   <dbl> <chr>  <dbl>
## 1     1 2015     1.88
## 2     2 2015     0.59
## 3     3 2015     0.35
## 4     2 2016     0.92
## 5     3 2016     0.17
## 6     4 2016     2.66
```

```
stocks %>%
  filter(!is.na(return))
```

```
## # A tibble: 6 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2016     2   0.92
## 5  2016     3   0.17
## 6  2016     4   2.66
```

Fill

```
treatment <- tribble(
~ person, ~ treatment, ~response,
"Derrick Whitmore", 1, 7,
NA, 2, 10,
NA, 3, 9,
"Katherine Burke", 1, 4
)
treatment
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 <NA>                     2       10
## 3 <NA>                     3        9
## 4 Katherine Burke          1        4
```

Fill by most recent non missing value.

```
treatment %>%
fill(person)
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Derrick Whitmore         2       10
## 3 Derrick Whitmore         3        9
## 4 Katherine Burke          1        4
```

Always document how you made the tidy data from the untidy data.