Tidy Data with tidyr

Data Analysis and Visualization (Fall 2019)

Instructor: Debopriya Ghosh

This exercise help to organize data in R. The tools provided in the package will help munging data from one representation to another.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

Reshaping Data

The underlying data can be represented in multiple ways. The three rules for making the data tidy are: (1) Each variable must have its own column (2) Each observation muct have its own row (3) Each value must have its own cell

```r
data("table1")
data("table2")
data("table3")
data("table4a")
data("table4b")
```

```r
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
```

```r
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
```

All these are representations of the same underlying data, but they are not equally easy to use.

Spreading and Gathering

Unfortunately, most data that you will encounter will be untidy. A lot of time is spent working on the data.

First, figure out what the variables and observations are. Second, resolve the common problems: (i) One variable might be spread across multiple columns (ii) One observation might be scattered across multiple rows

Gathering

A common problem is a dataset where some column names are not names of varibles, but values of a variable. Example, in Table 4a, the columns 1999 and 2000 represent the values of the year variable and each row represents two observations, not one. To tidy the dataset, we need to gather those columns into a new pair of variables.

```
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
```

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

```
## # 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
```

Combine table4a and table4b into a single tibble.

```
tidy4a = table4a%>%
  gather(`1999`,`2000`, key = "year",value = "cases")
tidy4b = table4b %>%
  gather(`1999`,`2000`,key = "year", value = "population")
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

Spreading is opposite to gathering. We use it when an observation is scattered across multiple rows. Example, in table2, an observation is a country in a year, but each observation is spread across two rows.

```
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
```

spread() and gather() are complements. gather() makes wide tables narrower and longer; spread() makes long table shorter and wider.

Separating and Pull

Seperate separate() pulls one column apart into multiple columns, by splitting wherever a separator character appears.

```
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
```

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

```
## # 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(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
```

By default separate() will split values wherever there is a alphanumeric character. But we can also specify a character to separate a column.

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

```
## # 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
```

Unite()

Unite is inverse of separate. It combines multiple columns into a single column.

```
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

Missing values can be encountered in one of two possible ways: * Explicitly: flagged with NA * Implicitly: Simply not present in the data

```
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)
)
```

There are two missing values in this dataset. * The return for the fourth quarter of 2015 is explicitly missing because the cell contains NA * The return for first quarter of 2016 is implicitly missing, because it simply doesnot appear in the dataset

We can make the implicit missing value explicit by putting years in columns

```
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
```

Because the explicit missing values may not be important in representations of the data, we can set na.rm = TRUE in gather()

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

```
## # 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
```

Another alternative is to use complete()

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

```
## # 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
```