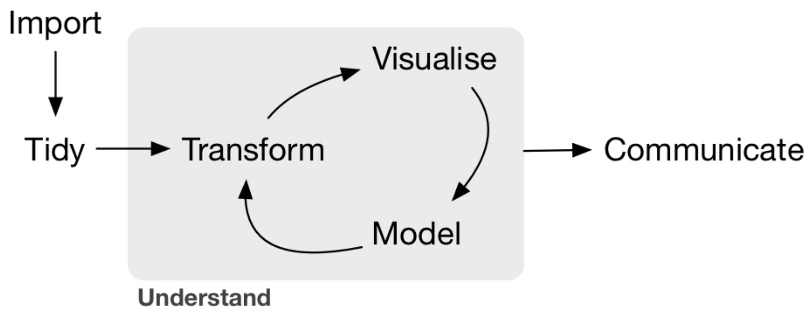
Data Wrangling with `tidyr`

Stephanie Hicks, Rafael Irizarry

The data analysis process can be thought about in four parts

1. Data cleaning
2. Data transformation
3. Data visualization
4. Modeling

where we each of these steps need their own tools and software to complete.



As we have seen in class, one of the most time-consuming aspects of the data analysis process is "data wrangling". This is also known as "data munging", which is a trendy term for *cleaning up a messy data set*. This refers to the first two steps in the data analysis process:

1. Data cleaning (or tidying data)
2. Data transformation

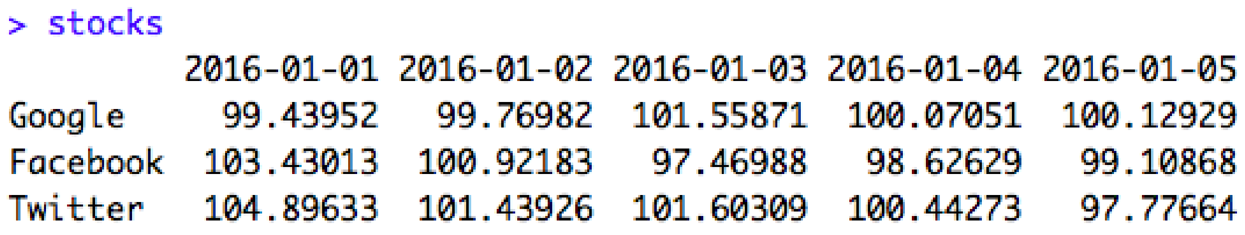
It can take a long time to clean and transform messy data into a format that is useful for data visualization and modeling, but there are tools that can help turn messy data into clean data.

Defining data structures

There are many ways to define the structure of a data set. Most data frames are made up of **rows** and **columns** where the columns are almost always labeled and the rows are *sometimes* labeled.

For example, a data set could be structured in the following way:

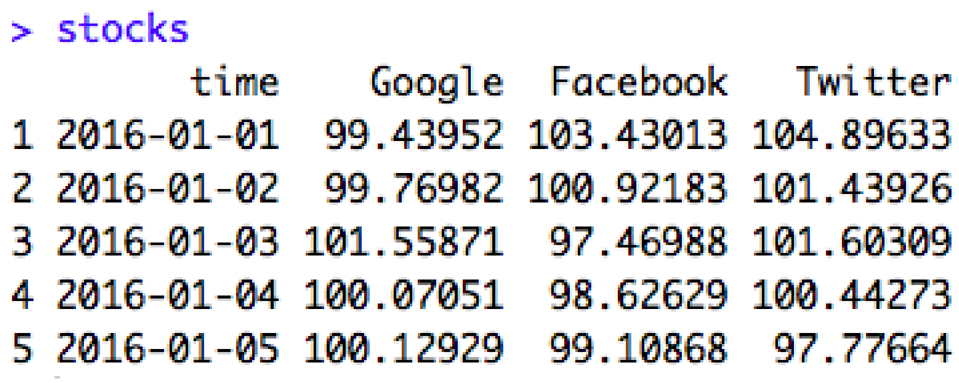
- each row represents one company (row names are companies)
- each column represent one time point
- the stock prices are defined for each row/column pair



```
> stocks
         2016-01-01 2016-01-02 2016-01-03 2016-01-04 2016-01-05
Google     99.43952   99.76982  101.55871  100.07051  100.12929
Facebook  103.43013  100.92183   97.46988   98.62629   99.10868
Twitter   104.89633  101.43926  101.60309  100.44273   97.77664
```

Alternatively, a data set can be structured in the following way:

- each row represents one time point (but no row names)
- the first column defines the time variable and the last three columns contain the stock prices for three companies



```
> stocks
        time    Google  Facebook   Twitter
1 2016-01-01  99.43952 103.43013 104.89633
2 2016-01-02  99.76982 100.92183 101.43926
3 2016-01-03 101.55871  97.46988 101.60309
4 2016-01-04 100.07051  98.62629 100.44273
5 2016-01-05 100.12929  99.10868  97.77664
```

In both cases, the data is the same, but the structure is different. This can be *frustrating* to deal with because the meaning of the values (rows and columns) in the two data sets are different. Providing a standardized way of organizing values within a data set would alleviate a major portion of this frustration.

Defining tidy data

Now, we will introduce the concept of **tidy** data. Tidy data is a standard way of mapping the meaning of a dataset to its structure. The properties of a tidy data set are based on:

- Each column is a variable
- Each rows is an observation

Working with tidy data is useful because it creates a structured way of organizing data values within a data set. This makes the data analysis process more efficient and simplifies the development of data analysis tools that work together. In this way, you can focus on the problem you are investigating, rather than the uninteresting logistics of data.

What is `tidyr`?

`tidyr` is an R package that transforms data sets to a tidy format.

There are two main functions in `tidyr`:

- `gather()` = takes multiple columns, and gathers them into key-value pairs
  (it makes "wide" data longer)
- `separate()` = turns a single character column into multiple columns (it makes "long" data wider)

We'll explore what it means to go between a "wide" and "long" data format using `gather()` and `separate()` next.

How do I get tidyr?

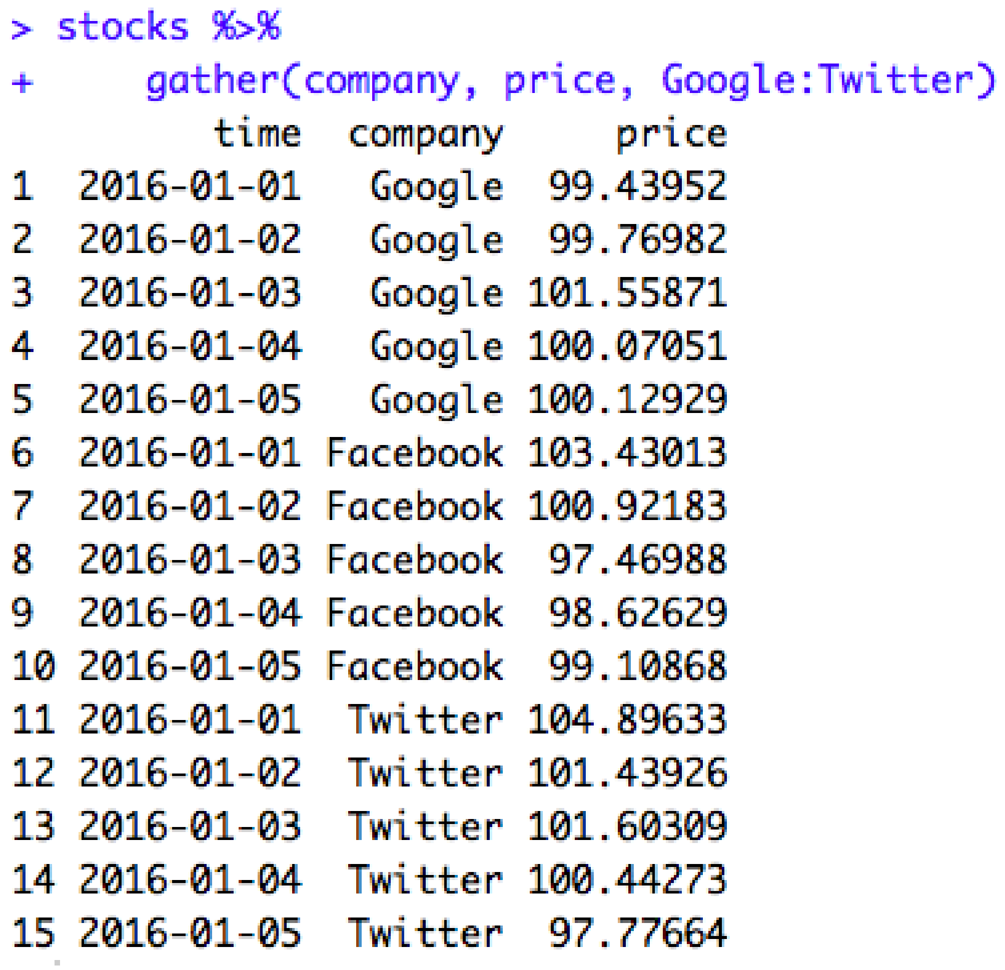
To install `tidyr`

```r
install.packages("tidyr")
```

To load `tidyr` and we'll need `dplyr`

```r
library(tidyr)
library(dplyr)
```

For motivation, a tidy version of the stock data we looked at above looks like this: (we'll learn how the functions work in just a moment)



```
> stocks %>%
+     gather(company, price, Google:Twitter)
         time  company     price
1  2016-01-01   Google  99.43952
2  2016-01-02   Google  99.76982
3  2016-01-03   Google 101.55871
4  2016-01-04   Google 100.07051
5  2016-01-05   Google 100.12929
6  2016-01-01 Facebook 103.43013
7  2016-01-02 Facebook 100.92183
8  2016-01-03 Facebook  97.46988
9  2016-01-04 Facebook  98.62629
10 2016-01-05 Facebook  99.10868
11 2016-01-01  Twitter 104.89633
12 2016-01-02  Twitter 101.43926
13 2016-01-03  Twitter 101.60309
14 2016-01-04  Twitter 100.44273
15 2016-01-05  Twitter  97.77664
```

In this "tidy" data set, we have three columns representing three variables (time, company name and stock price). Every row represents contains one stock price from a particular time and for a specific company.

Pipe operator: %>%

We have introduced the operator: `%>%`. dplyr imports this operator from another package (`magrittr` see help file here). This operator allows you to pipe the output from one function to the input of another function. Instead of nesting functions (reading from the inside to the outside), the idea of of piping is to read the functions from left to right.

Now in this case, we pipe the `stocks` data frame to the function that will gather multiple columns into key-value pairs.

Data

2016 Iowa Presidential Caucus

We will explore public poll data from HuffPost Pollster from the 2016 Iowa Democratic and Republican Presidential Caucus.

First we will read in the data:

```
library(readr)
dem_polls = read_csv("http://elections.huffingtonpost.com/pollster/2016-iowa-presidential-democratic-cau
rep_polls = read_csv("http://elections.huffingtonpost.com/pollster/2016-iowa-presidential-republican-cau
```

Let's take a look at data

```
View(dem_polls)
View(rep_polls)

glimpse(dem_polls)
glimpse(rep_polls)
```

We see there is a lot of information in each data frame. First let's use `dplyr` to select a subset of the columns.

```
dem_polls <- dem_polls %>%
                select(Pollster, `End Date`, Clinton:Undecided)

rep_polls <- rep_polls %>%
                select(Pollster, `End Date`, Trump:Walker)
```

In the democratic and republican polling data sets, there is one column representing the polling percentages for each candidate, similar to the stock price data set with multiple columns representing different companies. To **tidy** it, we need to *gather* these columns into a two-column *key-value* pair. This is often described as transforming a *wide* data set into a *long* data set.

gather()

This function gathers multiple columns and collapses them into new *key-value* pairs. This transform data from *wide* format into a *long* format.

- The `key` is the name of the *new* column that you are creating which contains the values of the column headings that you are gathering
- The `value` is the name of the *new* column that will contain the values themselves
- The third argument defines the columns to gather

```
dem_polls %>%
    gather(key = candidate, value = percentage, Clinton:Undecided)
```

```
## Source: local data frame [648 x 4]
##
##                                   Pollster   End Date candidate percentage
##                                      (chr)     (date)     (chr)      (int)
## 1       Emerson College Polling Society 2016-01-31   Clinton         51
## 2                            Quinnipiac 2016-01-31   Clinton         46
## 3  Des Moines Register/Bloomberg/Selzer 2016-01-29   Clinton         45
## 4     Gravis Marketing/One America News 2016-01-27   Clinton         53
## 5                  PPP (D-Progress Iowa) 2016-01-27   Clinton         48
## 6                         NBC/WSJ/Marist 2016-01-26   Clinton         48
## 7                    Monmouth University 2016-01-26   Clinton         47
## 8                                    ARG 2016-01-24   Clinton         45
## 9                             Quinnipiac 2016-01-24   Clinton         45
## 10                  Iowa State/WHO-HD 2016-01-22   Clinton         47
## ..                                   ...        ...       ...        ...
```

To select a range of columns by name, use the ":" (colon) operator

Assessment Using the democratic poll data, apply the `gather()` function to tidy the poll data by *excluding* the Pollster and End Date columns, rather than directly providing the column names to gather.

Hint: Look at the `gather()` help file on how to exclude column names.

```
## Provide your code here

dem_polls %>%
    gather(key = candidate, value = percentage, -c(Pollster, `End Date`))
```

```
## Source: local data frame [648 x 4]
##
##                                 Pollster   End Date candidate percentage
##                                      (chr)     (date)     (chr)      (int)
## 1       Emerson College Polling Society 2016-01-31   Clinton         51
## 2                            Quinnipiac 2016-01-31   Clinton         46
## 3  Des Moines Register/Bloomberg/Selzer 2016-01-29   Clinton         45
## 4     Gravis Marketing/One America News 2016-01-27   Clinton         53
## 5                  PPP (D-Progress Iowa) 2016-01-27   Clinton         48
## 6                         NBC/WSJ/Marist 2016-01-26   Clinton         48
## 7                    Monmouth University 2016-01-26   Clinton         47
## 8                                    ARG 2016-01-24   Clinton         45
## 9                             Quinnipiac 2016-01-24   Clinton         45
## 10                  Iowa State/WHO-HD 2016-01-22   Clinton         47
## ..                                   ...        ...       ...        ...
```

```
## To select all the columns *except* a specific column,
## use the "-" (subtraction) operator (also known as negative indexing)
```

Assessment Using the "tidy" democratic poll data, use dplyr to filter for only the following candidates (Clinton, Sanders, O'Malley) and for polls only ending after May 1, 2015.

```
## Provide your code here

dem_polls %>%
    gather(key = candidate, value = percentage, Clinton:Undecided) %>%
    filter(candidate %in% c("Clinton", "Sanders", "O'Malley") &
           `End Date` >= "2015-05-01")
```

```
## Source: local data frame [195 x 4]
##
##                                Pollster   End Date candidate percentage
##                                   (chr)     (date)     (chr)      (int)
## 1       Emerson College Polling Society 2016-01-31   Clinton         51
## 2                            Quinnipiac 2016-01-31   Clinton         46
## 3  Des Moines Register/Bloomberg/Selzer 2016-01-29   Clinton         45
## 4     Gravis Marketing/One America News 2016-01-27   Clinton         53
## 5                 PPP (D-Progress Iowa) 2016-01-27   Clinton         48
## 6                       NBC/WSJ/Marist 2016-01-26   Clinton         48
## 7                   Monmouth University 2016-01-26   Clinton         47
## 8                                   ARG 2016-01-24   Clinton         45
## 9                            Quinnipiac 2016-01-24   Clinton         45
## 10                   Iowa State/WHO-HD 2016-01-22   Clinton         47
## ..                                  ...        ...       ...        ...
```

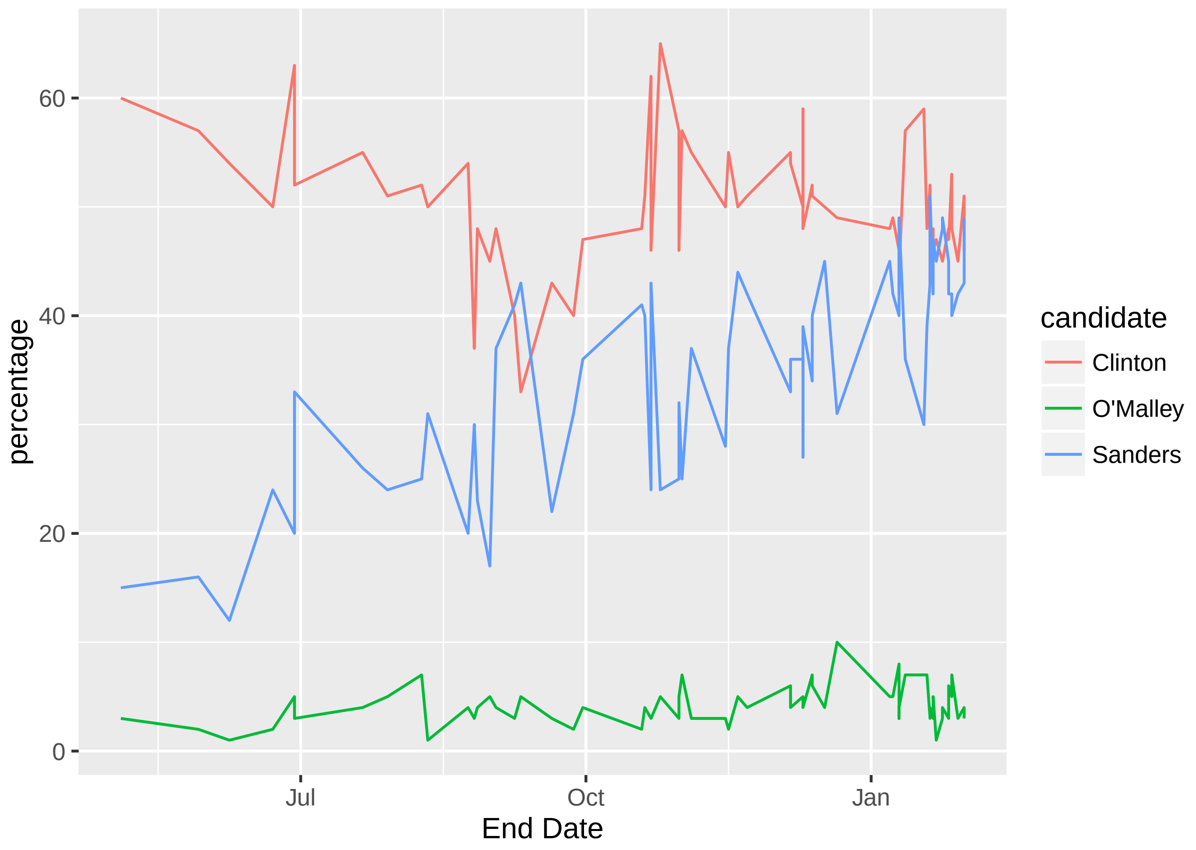
Assessment (optional) Using the tidy and filtered democratic poll data set, use `ggplot2` to plot the results from each poll (percentage) for each of the candiates. Color the lines by the candidate.

```
## Provide your code here

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
dem_polls %>%
    gather(key = candidate, value = percentage, Clinton:Undecided) %>%
    filter(candidate %in% c("Clinton", "Sanders", "O'Malley") &
           `End Date` >= "2015-05-01") %>%
    ggplot(aes(x=`End Date`, y = percentage, color = candidate)) +
    geom_line()
```

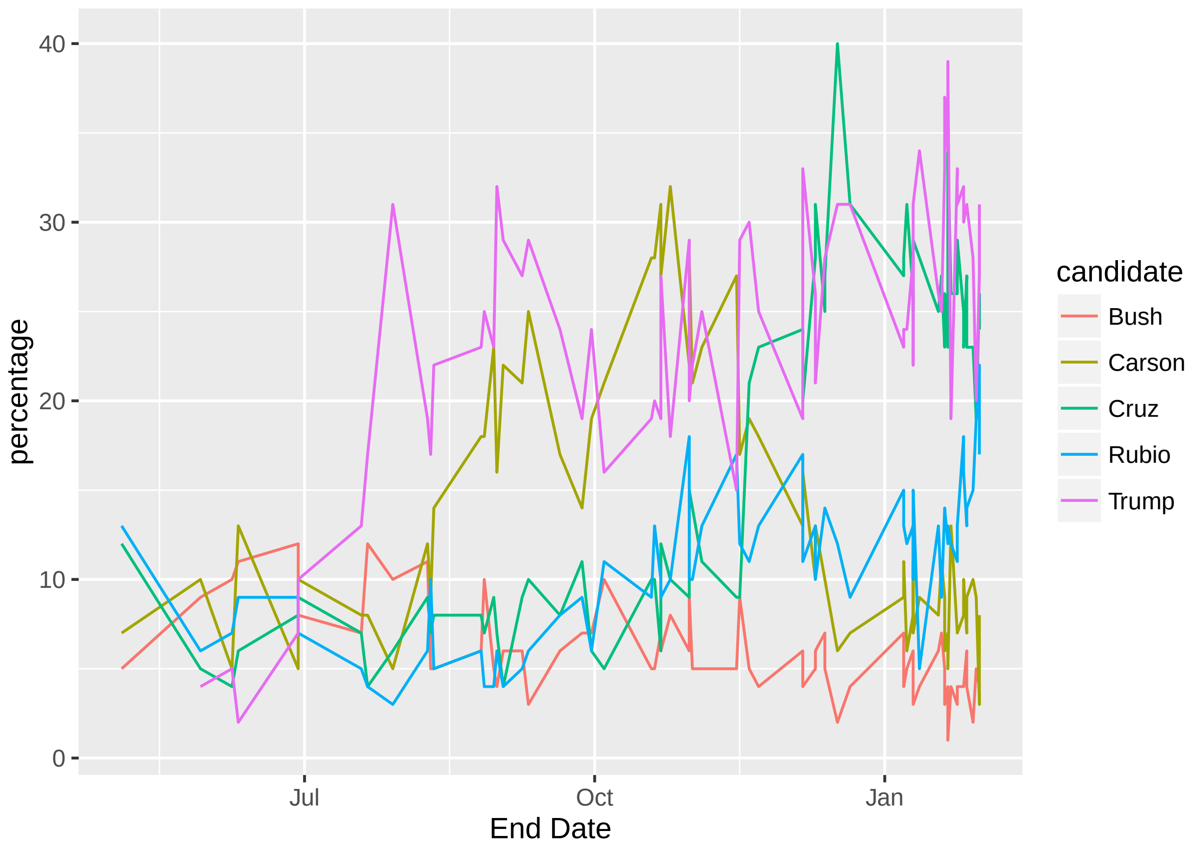


Assessment (optional) Repeat this analysis using the republican poll data. Filter for candidates (Trump, Cruz, Rubio, Carson, Bush) and for polls only after May 1, 2015. Color the lines by candidates.

```
## Provide your code here

rep_polls %>%
    gather(key = candidate, value = percentage, Trump:Walker) %>%
    filter(candidate %in% c("Trump", "Cruz", "Rubio", "Carson", "Bush") &
           `End Date` >= "2015-05-01") %>%
    ggplot(aes(x=`End Date`, y = percentage, color = candidate)) +
    geom_line()
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



spread()

In contrast to *gathering* multiple columns into key-value pairs, we can *spread* a key-value pair across multiple columns.

The function `spread()` does just that. It transforms data from a *long* format into a *wide* format.

- The `key` is the name of the column in your data set that contains the values of the column headings that you are spreading across multiple columns
- The `value` is the name of the column that contains the values for the multiple columns

```
dem_polls_gathered <- dem_polls %>%
                         gather(key = candidate, value = percentage,
                                Clinton:Undecided)
dem_polls_gathered
```

```
## Source: local data frame [648 x 4]
##
##                                Pollster   End Date candidate percentage
##                                   (chr)     (date)     (chr)      (int)
## 1       Emerson College Polling Society 2016-01-31   Clinton         51
## 2                            Quinnipiac 2016-01-31   Clinton         46
## 3  Des Moines Register/Bloomberg/Selzer 2016-01-29   Clinton         45
## 4     Gravis Marketing/One America News 2016-01-27   Clinton         53
## 5                 PPP (D-Progress Iowa) 2016-01-27   Clinton         48
## 6                        NBC/WSJ/Marist 2016-01-26   Clinton         48
## 7                   Monmouth University 2016-01-26   Clinton         47
## 8                                   ARG 2016-01-24   Clinton         45
## 9                            Quinnipiac 2016-01-24   Clinton         45
## 10                   Iowa State/WHO-HD 2016-01-22   Clinton         47
## ..                                  ...        ...       ...        ...
```

```
dem_polls_gathered %>%
    spread(key = candidate, value = percentage)
```

```
## Source: local data frame [81 x 10]
##
##      Pollster   End Date Biden Chafee Clinton Lessig O'Malley Sanders
##         (chr)     (date) (int)  (int)   (int)  (int)    (int)   (int)
## 1         ARG 2016-01-10    NA     NA      44     NA        3      47
## 2         ARG 2016-01-24    NA     NA      45     NA        3      48
## 3  CBS/YouGov 2015-09-10    10      1      33     NA        5      43
## 4  CBS/YouGov 2015-10-22    NA      1      46      0        3      43
## 5  CBS/YouGov 2015-11-19    NA     NA      50     NA        5      44
## 6  CBS/YouGov 2015-12-17    NA     NA      50     NA        4      45
## 7  CBS/YouGov 2016-01-21    NA     NA      46     NA        5      47
## 8         CNN 2014-09-10    15     NA      53     NA        2       5
## 9         CNN 2015-08-11    12      0      50     NA        1      31
## 10        CNN 2015-11-04    NA     NA      55     NA        3      37
## ..        ...        ...   ...    ...     ...    ...      ...     ...
## Variables not shown: Undecided (int), Webb (int)
```

Other supporting functions in tidyr

- `separate()` = separate one column into multiple columns
- `unite()` = unite multiple columns into one

```
dem_polls_separate <- dem_polls %>%
                        separate(col = `End Date`, into = c("y", "m", "d"))
dem_polls_separate
```

```
## Source: local data frame [81 x 12]
##
##                              Pollster     y     m     d Clinton Sanders
##                                 (chr) (chr) (chr) (chr)   (int)   (int)
## 1       Emerson College Polling Society  2016    01    31      51      43
## 2                            Quinnipiac  2016    01    31      46      49
## 3  Des Moines Register/Bloomberg/Selzer  2016    01    29      45      42
## 4      Gravis Marketing/One America News  2016    01    27      53      42
## 5                 PPP (D-Progress Iowa)  2016    01    27      48      40
## 6                        NBC/WSJ/Marist  2016    01    26      48      45
## 7                   Monmouth University  2016    01    26      47      42
## 8                                   ARG  2016    01    24      45      48
## 9                            Quinnipiac  2016    01    24      45      49
## 10                   Iowa State/WHO-HD  2016    01    22      47      45
## ..                                  ...   ...   ...   ...     ...     ...
## Variables not shown: O'Malley (int), Biden (int), Chafee (int), Lessig
##   (int), Webb (int), Undecided (int)
```

Assessment Use the `unite()` function to create a new column titled "end_date" that combines the columns `y`, `m` and `d` together into a single column separated by the "/" character.

```
## Provide your code here

dem_polls_separate %>%
    unite(col = end_date, y, m, d, sep = "/")
```

```
## Source: local data frame [81 x 10]
##
##                               Pollster   end_date Clinton Sanders
##                                  (chr)      (chr)   (int)   (int)
## 1       Emerson College Polling Society 2016/01/31      51      43
## 2                            Quinnipiac 2016/01/31      46      49
## 3  Des Moines Register/Bloomberg/Selzer 2016/01/29      45      42
## 4     Gravis Marketing/One America News 2016/01/27      53      42
## 5                  PPP (D-Progress Iowa) 2016/01/27      48      40
## 6                        NBC/WSJ/Marist 2016/01/26      48      45
## 7                    Monmouth University 2016/01/26      47      42
## 8                                   ARG 2016/01/24      45      48
## 9                            Quinnipiac 2016/01/24      45      49
## 10                     Iowa State/WHO-HD 2016/01/22      47      45
## ..                                   ...        ...     ...     ...
## Variables not shown: O'Malley (int), Biden (int), Chafee (int), Lessig
##   (int), Webb (int), Undecided (int)
```

Cheatsheets

- Data Wrangling with dplyr and tidyr from RStudio