Lesson 1: Tidy Data Manipulation I

David Robinson

January 26, 2015

Setup and Installation

First you need to install the dplyr package:

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

Then load it:

```r
library(dplyr)
```

You can see some great, detailed introductions and tutorials in the *vignettes*:

```r
browseVignettes("dplyr")
```

Cleaning Your Data

United Nations voting data

The UN voting data comes from this page- it is otherwise entirely unchanged. You can download it as:

```r
load(url("http://varianceexplained.org/courses/WS1015/files/undata-213.RData"))
```

This loads the variables from the RData file into your workspace. What variable is it? You can use `ls()` to find what variables you currently have loaded:

```r
ls()
```

```
## [1] "x"
```

Whatever you do, do not try printing x! It is too big for R to print, and this will crash it! You can, however, display some basic summaries of it:

```r
dim(x)
```

```
## [1] 1024539      20
```

You can even view it like a spreadsheet (it won't show the whole thing):

```r
View(x)
```

The dplyr package provides a way to change x's behavior so that if we accidentally print it, it won't ruin our day. This is the `tbl_df` class:

```r
x <- tbl_df(x)
class(x)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
x
```

```
## Source: local data frame [1,024,539 x 20]
##
##    rcid session       date    unres vote ccode                 uniquename
## 1     3       1 1946-01-01 "R/1/66"    1     2 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3    20                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    9    31                  "Bahamas"
## 4     3       1 1946-01-01 "R/1/66"    1    40                     "Cuba"
## 5     3       1 1946-01-01 "R/1/66"    1    41                    "Haiti"
## 6     3       1 1946-01-01 "R/1/66"    1    42       "Dominican Republic"
## 7     3       1 1946-01-01 "R/1/66"    9    51                  "Jamaica"
## 8     3       1 1946-01-01 "R/1/66"    9    52      "Trinidad and Tobago"
## 9     3       1 1946-01-01 "R/1/66"    9    53                 "Barbados"
## 10    3       1 1946-01-01 "R/1/66"    9    54                 "Dominica"
## ..  ...     ...        ...      ...  ...   ...                        ...
## Variables not shown: voetenoldcode (int), voetenname (AsIs),
##   voetenshortcode (AsIs), cowshortcode (AsIs), cowcode (int), cowlongname
##   (AsIs), aclpcode (int), wdicode (AsIs), imfcode (int), politycode (int),
##   bankscode (int), dpicode (AsIs), uncode (int)
```

Notice that it cuts off after a certain number of columns, and also a certain number of rows. However, it otherwise works just like a data.frame:

```r
head(x$rcid)
```

```
## [1] 3 3 3 3 3 3
```

```r
head(x$session)
```

```
## [1] 1 1 1 1 1 1
```

Looking at the data's structure

Now, let's look at the code book, which describes each of these columns. It can be downloaded from this page. Some of the things it shows are:

- rcid: Roll call vote ID: each of these identifies one vote
- session: One United Nations session: a year
- unres: a UN resolution (there might be multiple votes per resolution)
- vote: Coded vote:
    - 1 = Yes
    - 2 = Abstain
    - 3 = No

    - 8 = Absent
    - 9 = Not a member
- uniqueName: the name of the country

Everything else is lots and lots of other ways of describing countries. For us, these are not important.

We can find out a bit more about the columns using summary:

```r
summary(x)
```

```
##       rcid         session           date               unres
##  Min.   :   3   Min.   : 1.00   Length:1024539     Length:1024539
##  1st Qu.:1303   1st Qu.:26.00   Class :AsIs        Class :AsIs
##  Median :2603   Median :38.00   Mode  :character   Mode  :character
##  Mean   :2653   Mean   :36.74
##  3rd Qu.:3910   3rd Qu.:49.00
##  Max.   :9056   Max.   :67.00
##
##       vote           ccode        uniquename        voetenoldcode
##  Min.   :1.000   Min.   :  2.0   Length:1024539     Min.   :  2.0
##  1st Qu.:1.000   1st Qu.:290.0   Class :AsIs        1st Qu.:290.0
##  Median :1.000   Median :452.0   Mode  :character   Median :452.0
##  Mean   :3.658   Mean   :469.8                      Mean   :470.1
##  3rd Qu.:8.000   3rd Qu.:680.0                      3rd Qu.:680.0
##  Max.   :9.000   Max.   :990.0                      Max.   :990.0
##
##   voetenname        voetenshortcode    cowshortcode          cowcode
##  Length:1024539     Length:1024539     Length:1024539     Min.   :  2.0
##  Class :AsIs        Class :AsIs        Class :AsIs        1st Qu.:290.0
##  Mode  :character   Mode  :character   Mode  :character   Median :452.0
##                                                           Mean   :469.8
##                                                           3rd Qu.:680.0
##                                                           Max.   :990.0
##
##  cowlongname           aclpcode       wdicode            imfcode
##  Length:1024539     Min.   :  1.0   Length:1024539     Min.   :111.0
##  Class :AsIs        1st Qu.: 49.0   Class :AsIs        1st Qu.:299.0
##  Mode  :character   Median : 97.0   Mode  :character   Median :576.0
##                     Mean   : 97.5                      Mean   :550.3
##                     3rd Qu.:146.0                      3rd Qu.:734.0
##                     Max.   :199.0                      Max.   :968.0
##                     NA's   :26055                      NA's   :51096
##    politycode      bankscode         dpicode              uncode
##  Min.   :  2.0   Min.   :  10.0   Length:1024539     Min.   :  4.0
##  1st Qu.:290.0   1st Qu.: 302.0   Class :AsIs        1st Qu.:208.0
##  Median :452.0   Median : 660.0   Mode  :character   Median :428.0
##  Mean   :469.8   Mean   : 647.9                      Mean   :430.2
##  3rd Qu.:680.0   3rd Qu.: 986.2                      3rd Qu.:646.0
##  Max.   :990.0   Max.   :1300.0                      Max.   :894.0
##                  NA's   :46899                       NA's   :33435
```

We can see how the roll call ID differs from the UN resolutions (turns out there can be more than one vote per resolution) by using `length` and `unique`:

```r
length(unique(x$rcid))
```

```
## [1] 5211
```

```r
length(unique(x$unres))
```

```
## [1] 5043
```

dplyr: selecting columns

dplyr provides functions for manipulating our data. We mentioned before that there are some columns we dont care about. dplyr provides the `select` function to extract the columns we want:

```r
select(x, rcid, session, date)
```

```
## Source: local data frame [1,024,539 x 3]
##
##    rcid session       date
## 1     3       1 1946-01-01
## 2     3       1 1946-01-01
## 3     3       1 1946-01-01
## 4     3       1 1946-01-01
## 5     3       1 1946-01-01
## 6     3       1 1946-01-01
## 7     3       1 1946-01-01
## 8     3       1 1946-01-01
## 9     3       1 1946-01-01
## 10    3       1 1946-01-01
## ..  ...     ...        ...
```

```r
select(x, rcid, session, date, unres, vote, uniquename)
```

```
## Source: local data frame [1,024,539 x 6]
##
##    rcid session       date    unres vote                 uniquename
## 1     3       1 1946-01-01 "R/1/66"    1 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    9                  "Bahamas"
## 4     3       1 1946-01-01 "R/1/66"    1                     "Cuba"
## 5     3       1 1946-01-01 "R/1/66"    1                    "Haiti"
## 6     3       1 1946-01-01 "R/1/66"    1       "Dominican Republic"
## 7     3       1 1946-01-01 "R/1/66"    9                  "Jamaica"
## 8     3       1 1946-01-01 "R/1/66"    9      "Trinidad and Tobago"
## 9     3       1 1946-01-01 "R/1/66"    9                 "Barbados"
## 10    3       1 1946-01-01 "R/1/66"    9                 "Dominica"
## ..  ...     ...        ...      ...  ...                        ...
```

You can also select multiple consecutive columns using ::

```
select(x, rcid:uniquename)
```

```
## Source: local data frame [1,024,539 x 7]
##
##    rcid session       date    unres vote ccode                 uniquename
## 1     3       1 1946-01-01 "R/1/66"    1     2 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3    20                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    9    31                  "Bahamas"
## 4     3       1 1946-01-01 "R/1/66"    1    40                     "Cuba"
## 5     3       1 1946-01-01 "R/1/66"    1    41                    "Haiti"
## 6     3       1 1946-01-01 "R/1/66"    1    42       "Dominican Republic"
## 7     3       1 1946-01-01 "R/1/66"    9    51                  "Jamaica"
## 8     3       1 1946-01-01 "R/1/66"    9    52      "Trinidad and Tobago"
## 9     3       1 1946-01-01 "R/1/66"    9    53                 "Barbados"
## 10    3       1 1946-01-01 "R/1/66"    9    54                 "Dominica"
## ..  ...     ...        ...      ...  ...   ...                        ...
```

```
select(x, rcid:vote, uniquename)
```

```
## Source: local data frame [1,024,539 x 6]
##
##    rcid session       date    unres vote                 uniquename
## 1     3       1 1946-01-01 "R/1/66"    1 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    9                  "Bahamas"
## 4     3       1 1946-01-01 "R/1/66"    1                     "Cuba"
## 5     3       1 1946-01-01 "R/1/66"    1                    "Haiti"
## 6     3       1 1946-01-01 "R/1/66"    1       "Dominican Republic"
## 7     3       1 1946-01-01 "R/1/66"    9                  "Jamaica"
## 8     3       1 1946-01-01 "R/1/66"    9      "Trinidad and Tobago"
## 9     3       1 1946-01-01 "R/1/66"    9                 "Barbados"
## 10    3       1 1946-01-01 "R/1/66"    9                 "Dominica"
## ..  ...     ...        ...      ...  ...                        ...
```

```
select(x, rcid:vote, country = uniquename)
```

```
## Source: local data frame [1,024,539 x 6]
##
##    rcid session       date    unres vote                    country
## 1     3       1 1946-01-01 "R/1/66"    1 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    9                  "Bahamas"
## 4     3       1 1946-01-01 "R/1/66"    1                     "Cuba"
## 5     3       1 1946-01-01 "R/1/66"    1                    "Haiti"
## 6     3       1 1946-01-01 "R/1/66"    1       "Dominican Republic"
## 7     3       1 1946-01-01 "R/1/66"    9                  "Jamaica"
## 8     3       1 1946-01-01 "R/1/66"    9      "Trinidad and Tobago"
## 9     3       1 1946-01-01 "R/1/66"    9                 "Barbados"
## 10    3       1 1946-01-01 "R/1/66"    9                 "Dominica"
## ..  ...     ...        ...      ...  ...                        ...
```

or can remove specific columns with `-`:

```r
select(x, -rcid, -date, -session, -ccode)
```

```
## Source: local data frame [1,024,539 x 16]
##
##       unres vote                 uniquename voetenoldcode
## 1  "R/1/66"    1 "United States of America"             2
## 2  "R/1/66"    3                   "Canada"            20
## 3  "R/1/66"    9                  "Bahamas"            31
## 4  "R/1/66"    1                     "Cuba"            40
## 5  "R/1/66"    1                    "Haiti"            41
## 6  "R/1/66"    1       "Dominican Republic"            42
## 7  "R/1/66"    9                  "Jamaica"            51
## 8  "R/1/66"    9      "Trinidad and Tobago"            52
## 9  "R/1/66"    9                 "Barbados"            53
## 10 "R/1/66"    9                 "Dominica"            54
## ..      ...  ...                        ...           ...
## Variables not shown: voetenname (AsIs), voetenshortcode (AsIs),
##   cowshortcode (AsIs), cowcode (int), cowlongname (AsIs), aclpcode (int),
##   wdicode (AsIs), imfcode (int), politycode (int), bankscode (int),
##   dpicode (AsIs), uncode (int)
```

The %>% operator:

Notice that the first argument to `select` is our data. That is true of all dplyr's functions. If we want to perform multiple operations, this becomes a hassle, because we're nesting function calls within function calls. But dplyr provides another way to write it:

```r
x %>% select(rcid:vote, country = uniquename)
```

```
## Source: local data frame [1,024,539 x 6]
##
##    rcid session       date    unres vote                    country
## 1     3       1 1946-01-01 "R/1/66"    1 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    9                  "Bahamas"
## 4     3       1 1946-01-01 "R/1/66"    1                     "Cuba"
## 5     3       1 1946-01-01 "R/1/66"    1                    "Haiti"
## 6     3       1 1946-01-01 "R/1/66"    1       "Dominican Republic"
## 7     3       1 1946-01-01 "R/1/66"    9                  "Jamaica"
## 8     3       1 1946-01-01 "R/1/66"    9      "Trinidad and Tobago"
## 9     3       1 1946-01-01 "R/1/66"    9                 "Barbados"
## 10    3       1 1946-01-01 "R/1/66"    9                 "Dominica"
## ..  ...     ...        ...      ...  ...                        ...
```

The `%>%` operator, which is typically pronounced "then", lets us pipe together multiple steps of an analysis. But it's nothing more than a simple conversion:

```r
a %>% f(b, c)
# becomes
f(a, b, c)
```

```r
a %>% f(b) %>% g(c, d, e)
# becomes
g(f(a, b), c, d, e)
```

Many data analyses consist of these consecutive operations. This makes the use of `%>%` very natural. So from now on we'll write them like:

```r
x %>% select(rcid:vote, country = uniquename)
```

Filter: removing rows based on a condition

Let's say we don't care about the Abstain or Absent votes. We can filter them out using another dplyr function, `filter`:

```r
x %>% select(rcid:vote, country = uniquename) %>%
    filter(vote < 8)
```

```
## Source: local data frame [699,744 x 6]
##
##    rcid session       date    unres vote                      country
## 1     3       1 1946-01-01 "R/1/66"    1 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    1                     "Cuba"
## 4     3       1 1946-01-01 "R/1/66"    1                    "Haiti"
## 5     3       1 1946-01-01 "R/1/66"    1       "Dominican Republic"
## 6     3       1 1946-01-01 "R/1/66"    1                   "Mexico"
## 7     3       1 1946-01-01 "R/1/66"    1                "Guatemala"
## 8     3       1 1946-01-01 "R/1/66"    1                 "Honduras"
## 9     3       1 1946-01-01 "R/1/66"    1              "El Salvador"
## 10    3       1 1946-01-01 "R/1/66"    1                "Nicaragua"
## ..  ...     ...        ...      ...  ...                          ...
```

Recall that this is just the same as:

```r
filter(select(x, rcid:vote, country = uniquename), vote < 8)
```

```
## Source: local data frame [699,744 x 6]
##
##    rcid session       date    unres vote                      country
## 1     3       1 1946-01-01 "R/1/66"    1 "United States of America"
## 2     3       1 1946-01-01 "R/1/66"    3                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"    1                     "Cuba"
## 4     3       1 1946-01-01 "R/1/66"    1                    "Haiti"
## 5     3       1 1946-01-01 "R/1/66"    1       "Dominican Republic"
## 6     3       1 1946-01-01 "R/1/66"    1                   "Mexico"
## 7     3       1 1946-01-01 "R/1/66"    1                "Guatemala"
## 8     3       1 1946-01-01 "R/1/66"    1                 "Honduras"
## 9     3       1 1946-01-01 "R/1/66"    1              "El Salvador"
## 10    3       1 1946-01-01 "R/1/66"    1                "Nicaragua"
## ..  ...     ...        ...      ...  ...                          ...
```

But it is already more readable.

Mutate: changing columns or adding new ones

Right now, votes are represented as 1 (Yes), 2 (Abstain), 3 (No). Let's turn them into a factor.

```
votes <- c("Yes", "Abstain", "No")
x %>% select(rcid:vote, country = uniquename) %>%
    filter(vote < 8) %>%
    mutate(vote = factor(votes[vote]))
```

```
## Source: local data frame [699,744 x 6]
##
##    rcid session       date    unres vote                    country
## 1     3       1 1946-01-01 "R/1/66"  Yes "United States of America"
## 2     3       1 1946-01-01 "R/1/66"   No                   "Canada"
## 3     3       1 1946-01-01 "R/1/66"  Yes                     "Cuba"
## 4     3       1 1946-01-01 "R/1/66"  Yes                    "Haiti"
## 5     3       1 1946-01-01 "R/1/66"  Yes       "Dominican Republic"
## 6     3       1 1946-01-01 "R/1/66"  Yes                   "Mexico"
## 7     3       1 1946-01-01 "R/1/66"  Yes                "Guatemala"
## 8     3       1 1946-01-01 "R/1/66"  Yes                 "Honduras"
## 9     3       1 1946-01-01 "R/1/66"  Yes              "El Salvador"
## 10    3       1 1946-01-01 "R/1/66"  Yes                "Nicaragua"
## ..  ...     ...        ...      ...  ...                        ...
```

Secondly, you might be bothered that the country name, and the UN resolution, have quotes around them. The `stringr` package provides a function, `str_replace`, to replace letters in a string with another letter.

```
library(stringr)

x %>% select(rcid:vote, country = uniquename) %>%
    filter(vote < 8) %>%
    mutate(vote = factor(votes[vote]),
           country = str_replace(country, '"', ''))
```

```
## Source: local data frame [699,744 x 6]
##
##    rcid session       date    unres vote                   country
## 1     3       1 1946-01-01 "R/1/66"  Yes United States of America"
## 2     3       1 1946-01-01 "R/1/66"   No                   Canada"
## 3     3       1 1946-01-01 "R/1/66"  Yes                     Cuba"
## 4     3       1 1946-01-01 "R/1/66"  Yes                    Haiti"
## 5     3       1 1946-01-01 "R/1/66"  Yes       Dominican Republic"
## 6     3       1 1946-01-01 "R/1/66"  Yes                   Mexico"
## 7     3       1 1946-01-01 "R/1/66"  Yes                Guatemala"
## 8     3       1 1946-01-01 "R/1/66"  Yes                 Honduras"
## 9     3       1 1946-01-01 "R/1/66"  Yes              El Salvador"
## 10    3       1 1946-01-01 "R/1/66"  Yes                Nicaragua"
## ..  ...     ...        ...      ...  ...                       ...
```

Dividing date into year/month/day with tidyr's separate

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

Right now, year, month and day are combined in the same variable, which limits the operations we can perform with them. Let's try separating them, using dplyr's `separate` operation.

```r
library(tidyr)

x %>% select(rcid:vote, country = uniquename) %>% filter(vote < 8) %>%
    mutate(vote = factor(votes[vote]), country = gsub('"', '', country)) %>%
    separate(date, c("year", "month", "day"))
```

```
## Source: local data frame [699,744 x 8]
##
##    rcid session year month day    unres vote                  country
## 1     3       1 1946    01  01 "R/1/66"  Yes United States of America
## 2     3       1 1946    01  01 "R/1/66"   No                   Canada
## 3     3       1 1946    01  01 "R/1/66"  Yes                     Cuba
## 4     3       1 1946    01  01 "R/1/66"  Yes                    Haiti
## 5     3       1 1946    01  01 "R/1/66"  Yes       Dominican Republic
## 6     3       1 1946    01  01 "R/1/66"  Yes                   Mexico
## 7     3       1 1946    01  01 "R/1/66"  Yes                Guatemala
## 8     3       1 1946    01  01 "R/1/66"  Yes                 Honduras
## 9     3       1 1946    01  01 "R/1/66"  Yes              El Salvador
## 10    3       1 1946    01  01 "R/1/66"  Yes                Nicaragua
## ..  ...     ...  ...   ... ...      ...  ...                      ...
```

Right now, `year`, `month` and `day` are all character vectors. We want them to be numbers. That's handled by the `convert` argument of `separate`. This time, let's save it into a data frame called `votes`:

```r
votes <- x %>% select(rcid:vote, country = uniquename) %>%
    filter(vote < 8) %>%
    mutate(vote = factor(votes[vote]), country = gsub('"', '', country)) %>%
    separate(date, c("year", "month", "day"), convert = TRUE)
```

This will be the final version of our `votes` data- we've processed the columns and given them reasonable names. Now we can get to the actually interesting operations.

Exploratory Data Analysis

Now that we have the data in the format we want, we can start actually exploring it to answer questions.

Grouping and Summarizing

An essential operation in data science is the "split-apply-combine" pattern (described here). This breaks up your data into smaller subgroups, performs some analysis on them, and then recombines the results.

This operation by itself doesn't do anything except record, inside the `votes` table, that we're grouping by that variable:

```r
votes %>% group_by(year)
```

```
## Source: local data frame [699,744 x 8]
## Groups: year
##
##    rcid session year month day    unres vote                  country
## 1     3       1 1946     1   1 "R/1/66"  Yes United States of America
## 2     3       1 1946     1   1 "R/1/66"   No                   Canada
## 3     3       1 1946     1   1 "R/1/66"  Yes                     Cuba
## 4     3       1 1946     1   1 "R/1/66"  Yes                    Haiti
## 5     3       1 1946     1   1 "R/1/66"  Yes       Dominican Republic
## 6     3       1 1946     1   1 "R/1/66"  Yes                   Mexico
## 7     3       1 1946     1   1 "R/1/66"  Yes                Guatemala
## 8     3       1 1946     1   1 "R/1/66"  Yes                 Honduras
## 9     3       1 1946     1   1 "R/1/66"  Yes              El Salvador
## 10    3       1 1946     1   1 "R/1/66"  Yes                Nicaragua
## ..  ...     ...  ...   ... ...      ...  ...                      ...
```

But when we apply the `summarize` operation later, that operation takes that grouping variable into account, and performs summaries within each `year`:

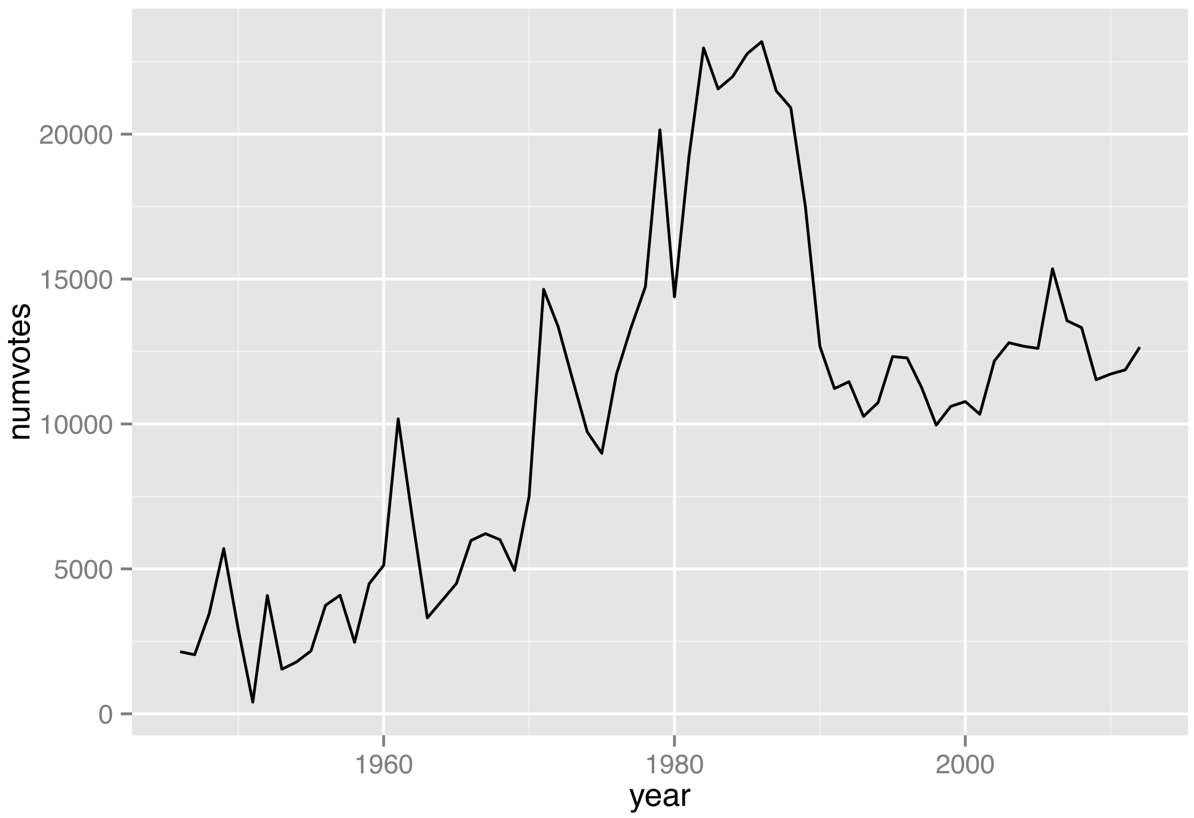
```r
votesumm <- votes %>% group_by(year) %>%
    summarize(numvotes = n())
votesumm
```

```
## Source: local data frame [66 x 2]
##
##    year numvotes
## 1  1946     2143
## 2  1947     2039
## 3  1948     3454
## 4  1949     5700
## 5  1950     2911
## 6  1951      402
## 7  1952     4082
## 8  1953     1537
## 9  1954     1788
## 10 1955     2169
## ..  ...      ...
```

Notice there is now one line per year (the original group), containing a new variable, `numvotes`, with the number of votes in that year.

To see why this might be useful, try using `ggplot2` to make a graph of votes per year:

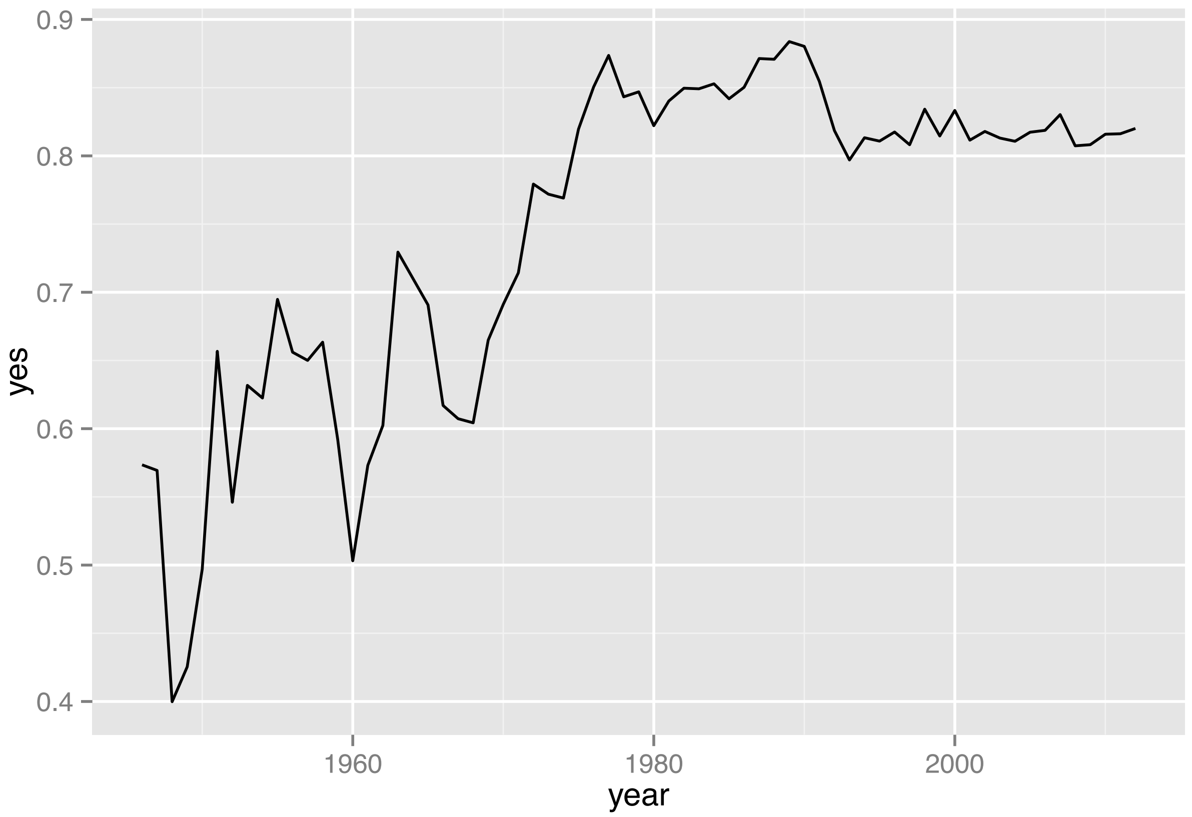
```r
library(ggplot2)
ggplot(votesumm, aes(year, numvotes)) + geom_line()
```



You can see that summarizing within each group makes it easy to produce graphs like these. But that's a pretty simple plot. Let's collect more information per year, and plot that. For starters, we could see how the proportion of countries that vote "Yes" on a resolution (a sort of measure of "general agreement") changes from year to year.

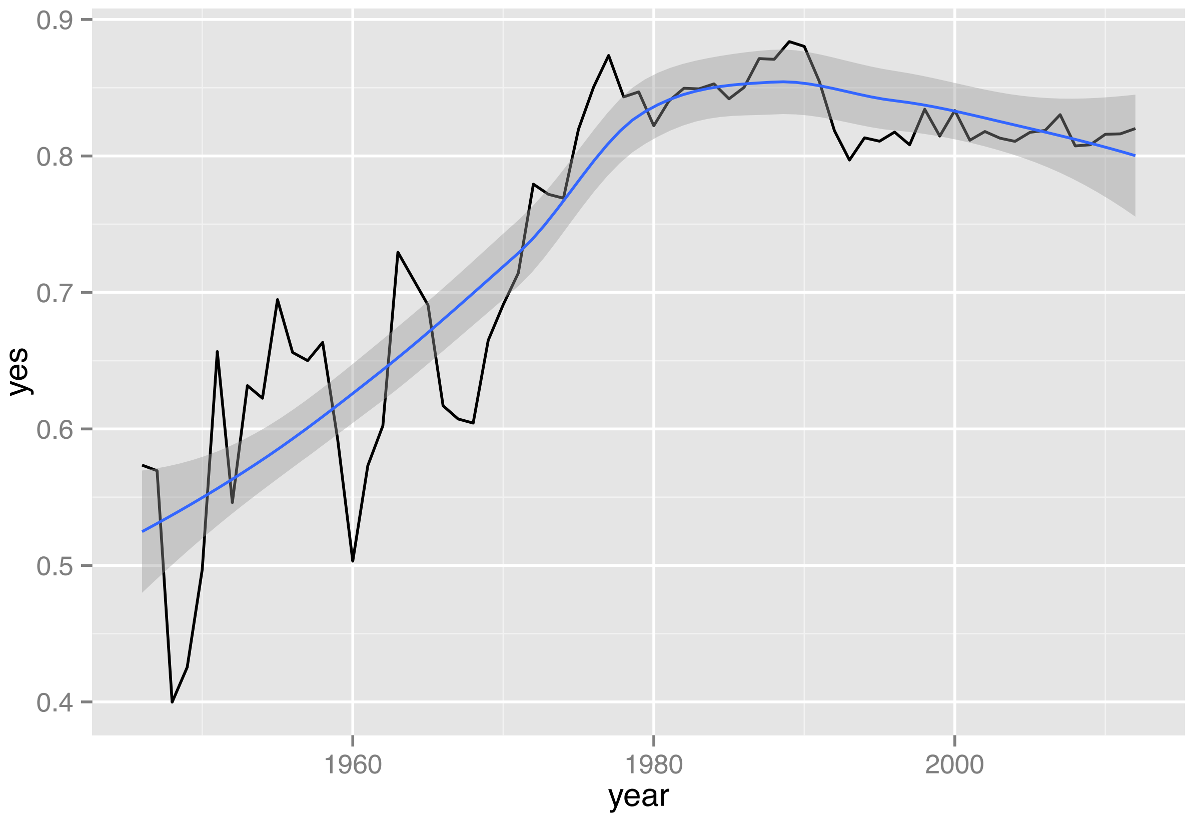
```
votesumm <- votes %>% group_by(year) %>%
    summarize(numvotes = n(), yes = mean(vote == "Yes"))

ggplot(votesumm, aes(year, yes)) + geom_line()
```



Throw in a trend line with `geom_smooth`:

```
ggplot(votesumm, aes(year, yes)) + geom_line() + geom_smooth()
```



Now, that's about as interesting as we can get while grouping by year. Let's get more interesting and group within *years and countries.* Do this by adding a second variable to the `group_by`

```
votesumm <- votes %>% group_by(year, country) %>%
    summarize(numvotes = n(), yes = mean(vote == "Yes"))
```

Notice that there is now one row for each *year-country pair.*

What countries are there to work with? Some of them may have names we're not used to. Here's a trick to visualize them:

```
sort(unique(votesumm$country))
```

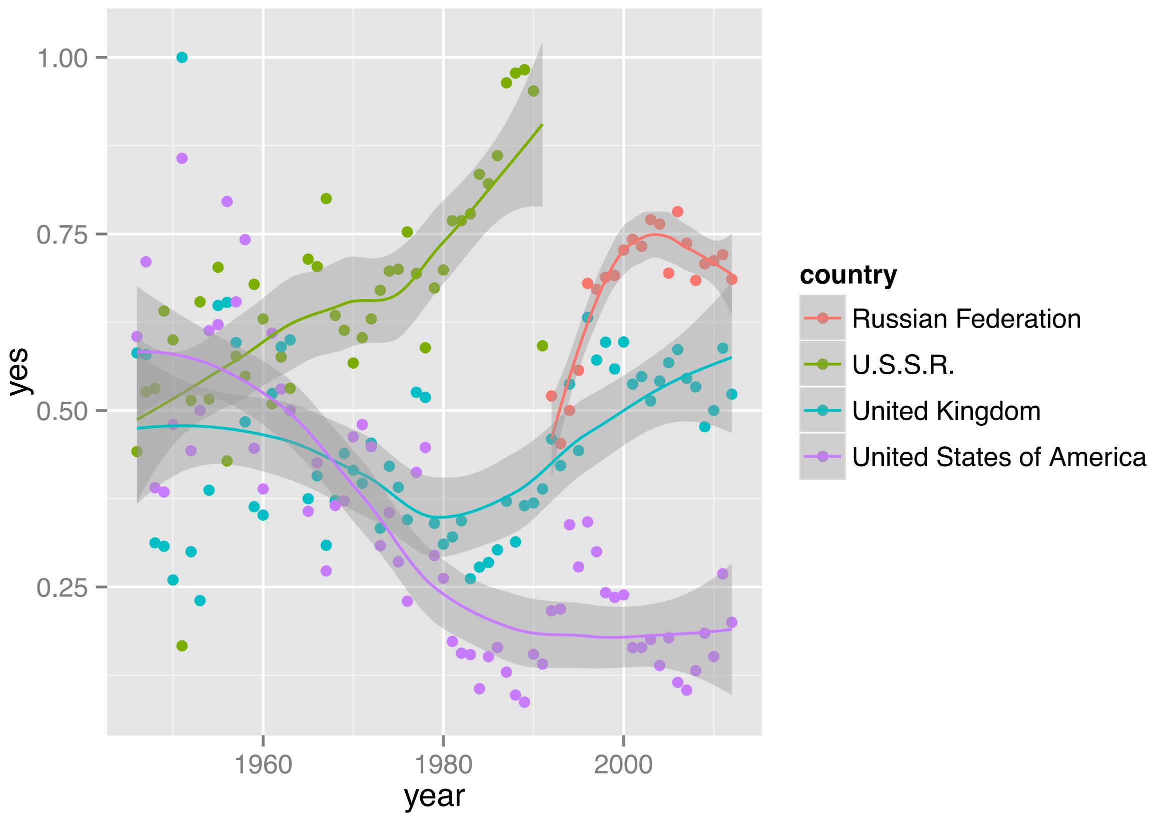
```
##   [1] "Afghanistan"                           "Albania"
##   [3] "Algeria"                               "Andorra"
##   [5] "Angola"                                "Antigua & Barbuda"
##   [7] "Argentina"                             "Armenia"
##   [9] "Australia"                             "Austria"
##  [11] "Azerbaijan"                            "Bahamas"
##  [13] "Bahrain"                               "Bangladesh"
##  [15] "Barbados"                              "Belarus"
##  [17] "Belgium"                               "Belize"
##  [19] "Benin"                                 "Bhutan"
##  [21] "Bolivia"                               "Bosnia and Herzegovina"
##  [23] "Botswana"                              "Brazil"
##  [25] "Brunei Darussalam"                     "Bulgaria"
##  [27] "Burkina Faso"                          "Burundi"
##  [29] "Cambodia"                              "Cameroon"
##  [31] "Canada"                           "Cape Verde"
##  [33] "Central African Republic"         "Chad"
##  [35] "Chile"                            "China"
##  [37] "Colombia"                         "Comoros"
##  [39] "Congo"                            "Costa Rica"
##  [41] "Cote d'Ivoire"                    "Croatia"
##  [43] "Cuba"                             "Cyprus"
##  [45] "Cyprus (old)"                     "Czech Republic"
##  [47] "Czechoslovakia"                   "Democratic Republic of the Congo"
##  [49] "Denmark"                          "Djibouti"
##  [51] "Dominica"                         "Dominican Republic"
##  [53] "East Timor"                       "Ecuador"
##  [55] "Egypt"                            "El Salvador"
##  [57] "Equatorial Guinea"                "Eritrea"
##  [59] "Estonia"                          "Ethiopia"
##  [61] "Ethiopia (new)"                   "Fiji"
##  [63] "Finland"                          "France"
##  [65] "Gabon"                            "Gambia"
##  [67] "Georgia"                          "Germany"
##  [69] "Germany, East"                    "Germany, West"
##  [71] "Ghana"                            "Greece"
##  [73] "Grenada"                          "Guatemala"
##  [75] "Guinea"                           "Guinea-Bissau"
##  [77] "Guyana"                           "Haiti"
##  [79] "Honduras"                         "Hungary"
##  [81] "Iceland"                          "India"
##  [83] "Indonesia"                        "Iran"
##  [85] "Iraq"                             "Ireland"
##  [87] "Israel"                           "Italy"
##  [89] "Jamaica"                          "Japan"
##  [91] "Jordan"                           "Kazakhstan"
##  [93] "Kenya"                            "Kiribati"
##  [95] "Kuwait"                           "Kyrgyzstan"
##  [97] "Laos"                             "Latvia"
##  [99] "Lebanon"                          "Lesotho"
## [101] "Liberia"                          "Libya"
## [103] "Liechtenstein"                    "Lithuania"
## [105] "Luxembourg"                       "Macedonia"
## [107] "Madagascar"                       "Malawi"
## [109] "Malaysia"                         "Maldives"
## [111] "Mali"                             "Malta"
## [113] "Marshall Islands"                 "Mauritania"
## [115] "Mauritius"                        "Mexico"
## [117] "Micronesia, Federated States of"  "Moldova"
## [119] "Monaco"                           "Mongolia"
## [121] "Montenegro"                       "Morocco"
## [123] "Mozambique"                       "Myanmar"
## [125] "Namibia"                          "Nauru"
## [127] "Nepal"                            "Netherlands"
## [129] "New Zealand"                      "Nicaragua"
## [131] "Niger"                            "Nigeria"
## [133] "North Korea"                      "Norway"
## [135] "Oman"                             "Pakistan"
## [137] "Pakistan (old)"                   "Palau"
## [139] "Panama"                          "Papua New Guinea"
## [141] "Paraguay"                        "Peru"
## [143] "Philippines"                     "Poland"
## [145] "Portugal"                        "Qatar"
## [147] "Romania"                         "Russian Federation"
## [149] "Rwanda"                          "Samoa"
## [151] "San Marino"                      "Sao Tome and Principe"
## [153] "Saudi Arabia"                    "Senegal"
## [155] "Serbia"                          "Serbia and Montenegro"
## [157] "Seychelles"                      "Sierra Leone"
## [159] "Singapore"                       "Slovakia"
## [161] "Slovenia"                        "Solomon Islands"
## [163] "Somalia"                         "South Africa"
## [165] "South Korea"                     "South Sudan"
## [167] "Spain"                           "Sri Lanka"
## [169] "St. Kitts and Nevis"             "St. Lucia"
## [171] "St. Vincent and the Grenadines"  "Sudan"
## [173] "Suriname"                        "Swaziland"
## [175] "Sweden"                          "Switzerland"
## [177] "Syria"                           "Taiwan"
## [179] "Tajikistan"                      "Tanzania"
## [181] "Thailand"                        "Togo"
## [183] "Tonga"                           "Trinidad and Tobago"
## [185] "Tunisia"                         "Turkey"
## [187] "Turkmenistan"                    "Tuvalu"
## [189] "U.S.S.R."                        "Uganda"
## [191] "Ukraine"                         "United Arab Emirates"
## [193] "United Kingdom"                  "United States of America"
## [195] "Uruguay"                         "Uzbekistan"
## [197] "Vanuatu"                         "Venezuela"
## [199] "Viet Nam"                        "Yemen"
## [201] "Yemen Arab Republic"             "Yemen PDR (South)"
## [203] "Yugoslavia"                      "Zambia"
## [205] "Zanzibar"                        "Zimbabwe"
```

Let's grab out a few that might interest us. (Note that the "U.S.S.R." turned into the "Russian Federation" starting in 1992).

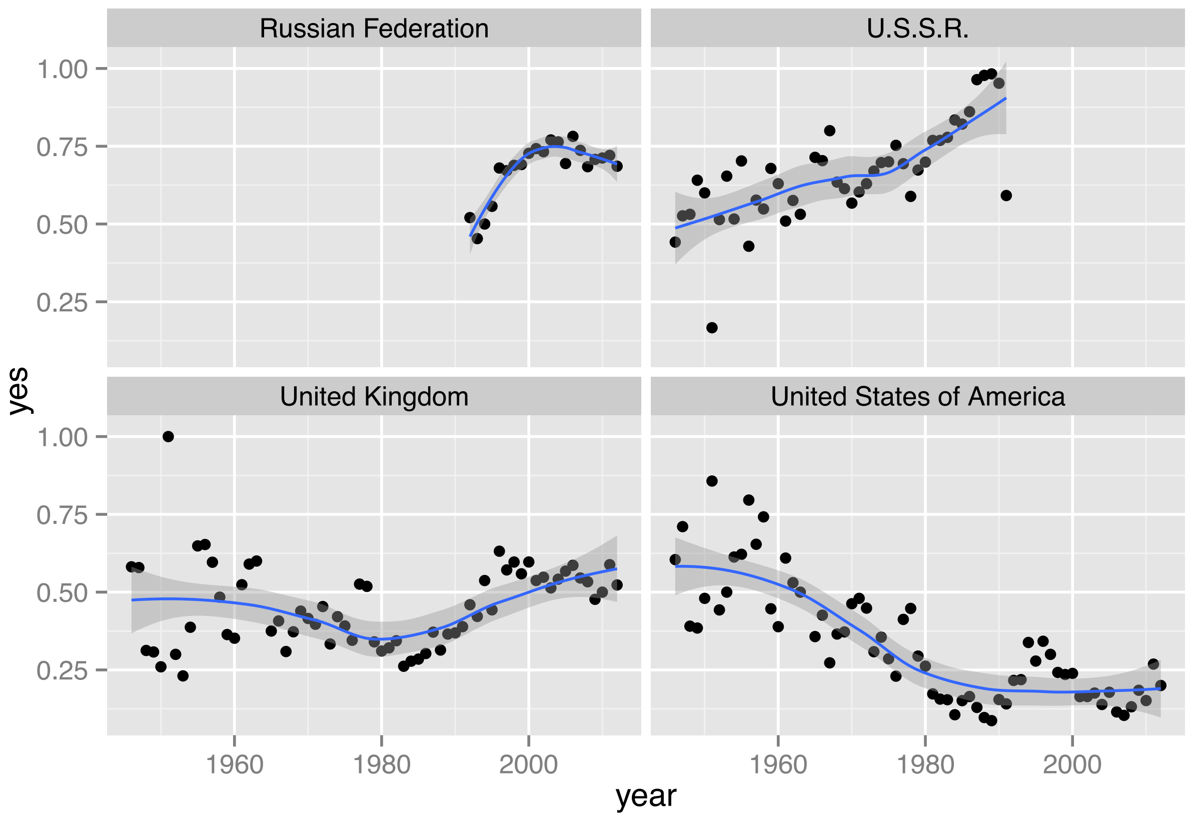
```
interesting_countries <- c("United States of America", "U.S.S.R.", "United Kingdom", "Russian Federatio
interesting <- votesumm %>% filter(country %in% interesting_countries)
```

Now that we've filtered for these countries, we can plot their "% Yes" metric separately over time. Here's two ways (of many) you can do this- separating countries by color and by facets (sub-plots):

```
ggplot(interesting, aes(year, yes, color = country)) + geom_point() +
    geom_smooth()
```



```r
ggplot(interesting, aes(year, yes)) + geom_point() +
    geom_smooth() +
    facet_wrap(~ country)
```



We can already pick out and start interpreting trends based on these four plots, of how each countries level of agreement with the UN's resolutions changed over time.

Tomorrow we'll continue diving into this data as an example of a tidy data analysis. We'll learn:

- How to merge this data with a different dataset that describes the topic and importance of each resolution, and create graphs based on these topics
- How to turn un-tidy data into tidy data using tidyr
- How to perform an analysis, such as a regression or spline, within each country using the broom package
- How to cluster countries by similarity in voting patterns, and construct heatmaps and trees