More data manipulation with dplyr and tidy

Completed by Leon Woltermann

Contents

NB: The worksheet has beed developed and prepared by Lincoln Mullen. Source: Lincoln A. Mullen, *Computational Historical Thinking: With Applications in R (2018–)*: http://dh-r.lincolnmullen.com. Minor modifications added by Maxim Romanov (loading `methodists` dataset).

The best way to learn R or computational history is to practice. These worksheets contain a series of questions designed to teach you about R or different computational methods. The worksheets are R Markdown documents that include text and code together. The places where you are expected to answer questions are marked like this.

`(@) Can you make a plot from this dataset?`

Beneath each question is a space to either create a code block or write an answer.

Aims of this worksheet

In an earlier worksheet, you learned the basic data manipulation verbs from the dplyr package: `select()`, `filter()`, `mutate()`, `arrange()`, `group_by()`, and `summarize()`. In this worksheet you will learn additional data verbs from the dplyr and tidyr packages. These data verbs relate to window functions (`lead()` and `lag()`), data table joins (`left_join()` et al.), and data reshaping (`spread()` and `gather()`)

To begin, we will load the necessary packages, as well as the Methodist data.

```
library(tidyverse)
library(historydata)
#data("methodists")
#methodists
```

`methodists data (MGR)`

If `methodists` dataset does not load, we can try the following. First, restart `R` (in the menu: `Session` > `Restart R`), then run the following lines:

```
devtools::install_github("ropensci/historydata", force=TRUE)
```

```
## Downloading GitHub repo ropensci/historydata@HEAD
```

```
## * checking for file '/private/var/folders/6f/0x08zkks1754nb4kts9p0t240000gn/T/Rtmp0RxYtj/remotes2c8f
## * preparing 'historydata':
## * checking DESCRIPTION meta-information ... OK
## * checking for LF line-endings in source and make files and shell scripts
## * checking for empty or unneeded directories
## * building 'historydata_0.2.9001.tar.gz'
```

```r
library(historydata)
data(methodists)
#methodists
```

Alternatively, we can load the data differently. The package itself is available on gitHub (https://github.com/ropensci/historydata), so we can try a different way of getting the data that we need for the worksheet. Specifically, if we know the exact address of the data file (url), we can open it with the `read.csv` command, like shown below (you need to be connected to Internet, of course):

```r
methodists <- read.csv("https://raw.githubusercontent.com/ropensci/historydata/master/data-raw/methodis
#methodists
```

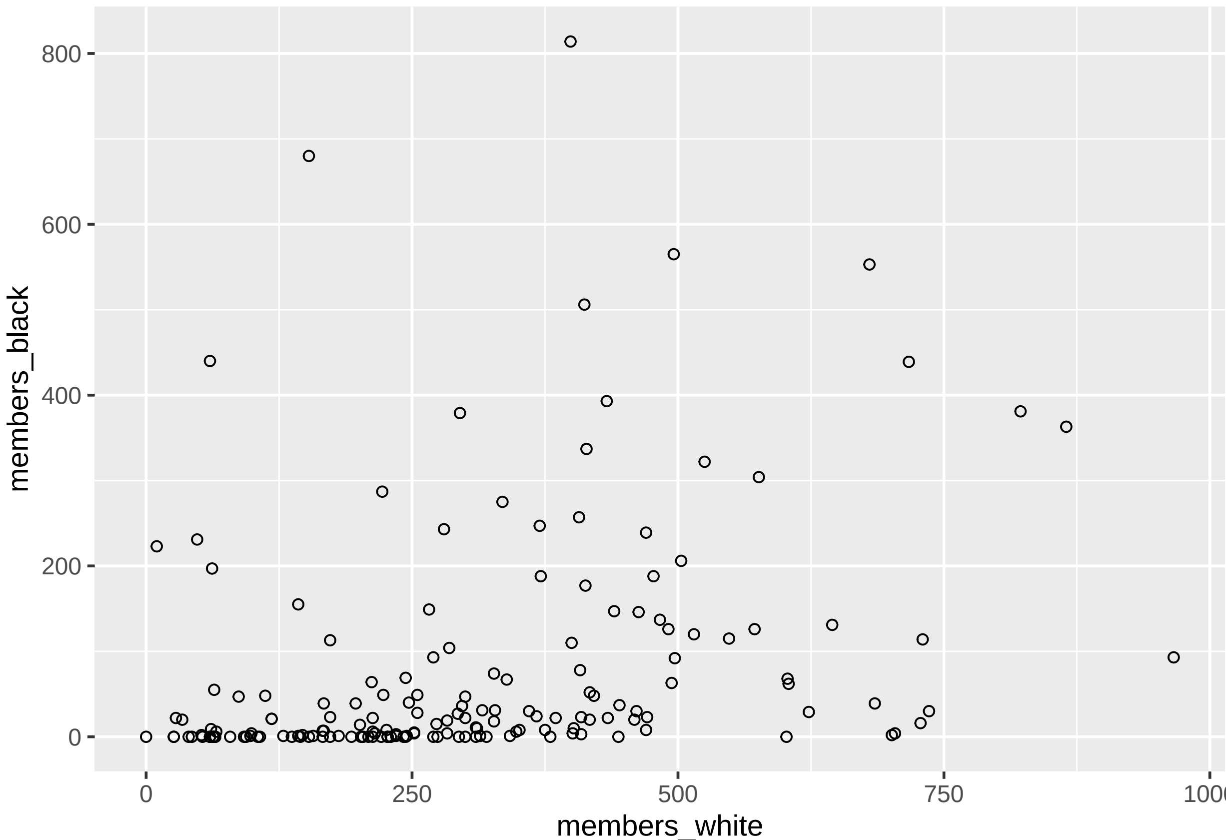
This data file, however, is slightly different from what we need, so some minor modifications will be necessary. You do not need to be concerned about the code in the next chunk, just run it.

```r
#library(dplyr)

replace_na <- function(x, val = 0L) {
  ifelse(is.na(x), val, x)
}

methodists <- methodists %>%
  as_tibble() %>%
  filter(minutes_year != 1778,
         minutes_year != 1779,
         minutes_year != 1785) %>%
  filter(minutes_year >= 1786, minutes_year <= 1834) %>%
  dplyr::rename(members_black = members_colored,
         year = minutes_year) %>%
  mutate(members_indian = as.integer(members_indian)) %>%
  mutate(members_white = replace_na(members_white),
         members_black = replace_na(members_black),
         members_indian = replace_na(members_indian)) %>%
  rowwise() %>%
  mutate(members_total = sum(members_general, members_white, members_black,
                             members_indian, na.rm = TRUE)) %>%
  ungroup() %>%
  select(year, conference, district, meeting, state, members_total,
         starts_with("members_"), url)
```

Data joining with two table verbs (`left_join()` et al.)

It is often the case that we want to use some variable in our data to create a new variable. Consider the Methodist data for the year 1800. Perhaps we are interested in the racial composition of the churches. Do they tend to be all white and all black, or do some churches have both white and black members in varying proportions? The simplest way to get a look at that question is to create a scatter plot of the figures for white and black membership.

```r
methodists_1800 <- methodists %>%
  filter(year == 1800) %>%
  select(meeting, state, members_white, members_black)

ggplot(methodists_1800, aes(x = members_white, y = members_black)) +
  geom_point(shape = 1)
```



That scatterplot is interesting as far as it goes, but we might reasonably suspect that the racial composition of methodist meetings varies by region. We could use the `state` variable to facet the plot by state. However, this has two problems. There are 20 states represented in that year. Our faceted plot would have 20 panels, which is too many. But more important, by looking at individual states we might be getting *too* fine grained a look at the data. We have good reason to think that it is regions that matter more than states.

It is easy enough to describe what we would do to translate states into a new column with regions. We would look at each state name and assign it to a region. Connecticut would be in the Northeast, New York would be in the Mid-Atlantic, and so on. We can think of this problem as looking up a value in one table (our Methodist data) in another table. That other table will have a row for each state, where each state name is associated with a region. (In many cases, though, it would make more sense to create a CSV file with the data and read it in as a data frame.)

```
regions <- tibble(
  state = c("Connecticut", "Delaware", "Georgia", "Kentucky", "Maine",
            "Maryland", "Massachusetts", "Mississippi", "New Hampshire",
            "New Jersey", "New York", "North Carolina",
            "Northwestern Territory", "Pennsylvania", "Rhode Island",
            "South Carolina", "Tennessee", "Upper Canada", "Vermont",
            "Virginia"),
  region = c("Northeast", "Atlantic South", "Atlantic South", "West",
             "Northeast", "Atlantic South", "Northeast", "Deep South",
             "Northeast", "Mid-Atlantic", "Mid-Atlantic", "Atlantic South",
             "West", "Mid-Atlantic", "Northeast", "Atlantic South", "West",
             "Canada", "Northeast", "Atlantic South")
)
```

And now we can inspect the table.

```r
regions
```

```
## # A tibble: 20 x 2
##    state                 region
##    <chr>                 <chr>
##  1 Connecticut           Northeast
##  2 Delaware              Atlantic South
##  3 Georgia               Atlantic South
##  4 Kentucky              West
##  5 Maine                 Northeast
##  6 Maryland              Atlantic South
##  7 Massachusetts         Northeast
##  8 Mississippi           Deep South
##  9 New Hampshire         Northeast
## 10 New Jersey            Mid-Atlantic
## 11 New York              Mid-Atlantic
## 12 North Carolina        Atlantic South
## 13 Northwestern Territory West
## 14 Pennsylvania          Mid-Atlantic
## 15 Rhode Island          Northeast
## 16 South Carolina        Atlantic South
## 17 Tennessee             West
## 18 Upper Canada          Canada
## 19 Vermont               Northeast
## 20 Virginia              Atlantic South
```

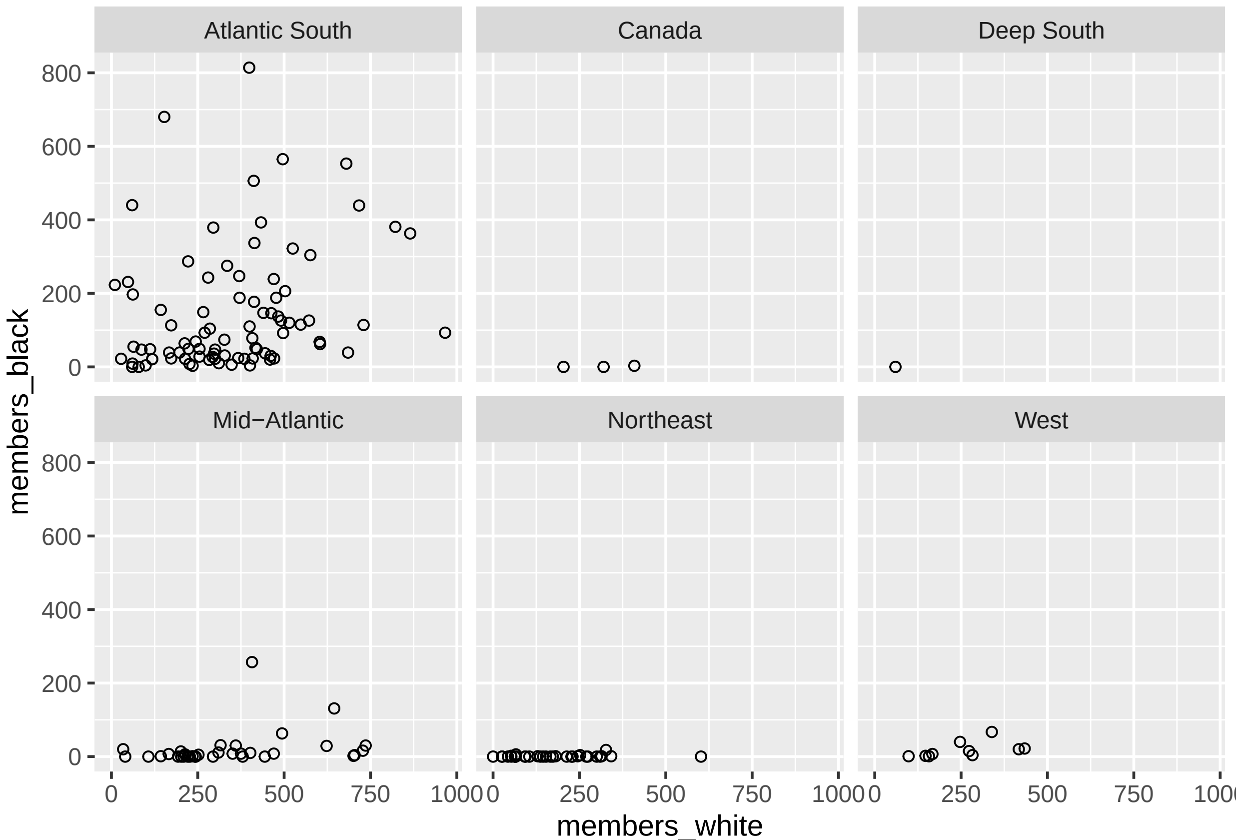
We can do a look up where we take the `state` column in the `methodists_1800` data frame and associate it with the `states` column in our `regions` data frame. The result will be a new column `region`. Notice how we use the `by =` argument to specify which column in the left hand table matches which column in the right hand table.

```r
methodists_region <- methodists_1800 %>%
  left_join(regions, by = "state")

methodists_region
```

```
## # A tibble: 169 x 5
##    meeting     state          members_white members_black region
##    <chr>       <chr>                  <int>         <int> <chr>
##  1 Augusta     Georgia                   61             9 Atlantic South
##  2 Burke       Georgia                  297            36 Atlantic South
##  3 Richmond    Georgia                  548           115 Atlantic South
##  4 Washington  Georgia                  497            92 Atlantic South
##  5 Broad River South Carolina           604            62 Atlantic South
##  6 Bush River  South Carolina           328            31 Atlantic South
##  7 Charleston  South Carolina            60           440 Atlantic South
##  8 Cherokee    South Carolina            79             0 Atlantic South
##  9 Edisto      South Carolina           572           126 Atlantic South
## 10 Georgetown  South Carolina            10           223 Atlantic South
## # ... with 159 more rows
```

Then we can plot the results. As we suspected, there is a huge regional variation.

```r
ggplot(methodists_region, aes(x = members_white, y = members_black)) +
  geom_point(shape = 1) +
  facet_wrap(~ region)
```



(1) Can you summarize the racial composition of the different regions by year (i.e., a region had a certain percentage white and black members for a given year) and create a plot of the changing racial composition in each region over time?

```
methodists_full <- methodists %>%
  select(year, state, members_total, members_white, members_black)

methodists_full_region <- methodists_full %>%
  left_join(regions, by="state") %>%
  select(-state)

methodists_full_region
```

```
## # A tibble: 20,241 x 5
##     year members_total members_white members_black region
##    <int>         <int>         <int>         <int> <chr>
##  1  1786           356           330            26 <NA>
##  2  1786           488           416            72 <NA>
##  3  1786           364           305            59 <NA>
##  4  1786           412           382            30 <NA>
##  5  1786           429           392            37 <NA>
##  6  1786           540           524            16 <NA>
##  7  1786           449           374            75 <NA>
##  8  1786           178           167            11 <NA>
##  9  1786           368           350            18 <NA>
## 10  1786           166           140            26 <NA>
## # ... with 20,231 more rows
```

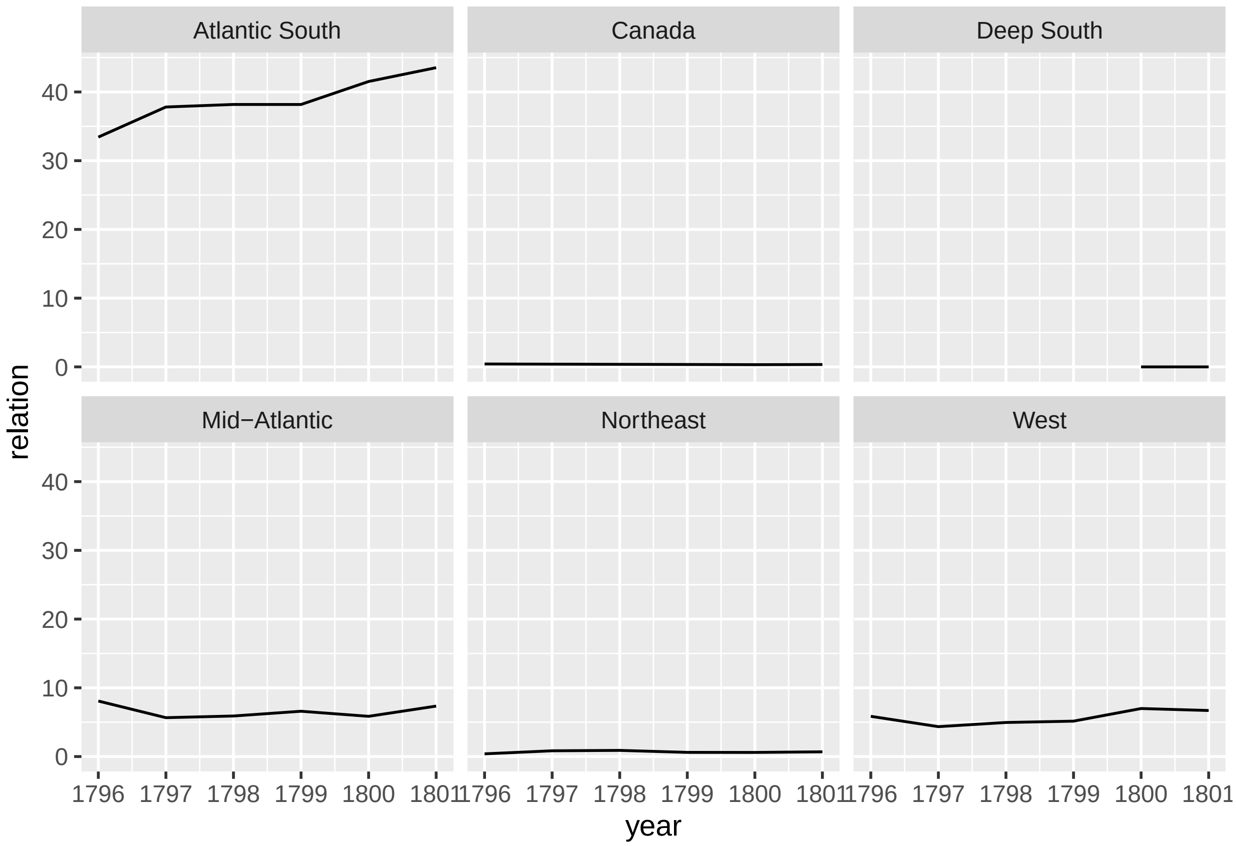
```r
racial_composition <- methodists_full_region %>%
  filter(!is.na(region)) %>%
  filter(members_total!=0) %>%
  group_by(year, region) %>%
  summarize(members_total = sum(members_total), members_white = sum(members_white), members_black = sum
  mutate(percentage_white = members_white / members_total * 100, percentage_black = members_black / meml
  mutate(relation = percentage_black / percentage_white * 100)
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

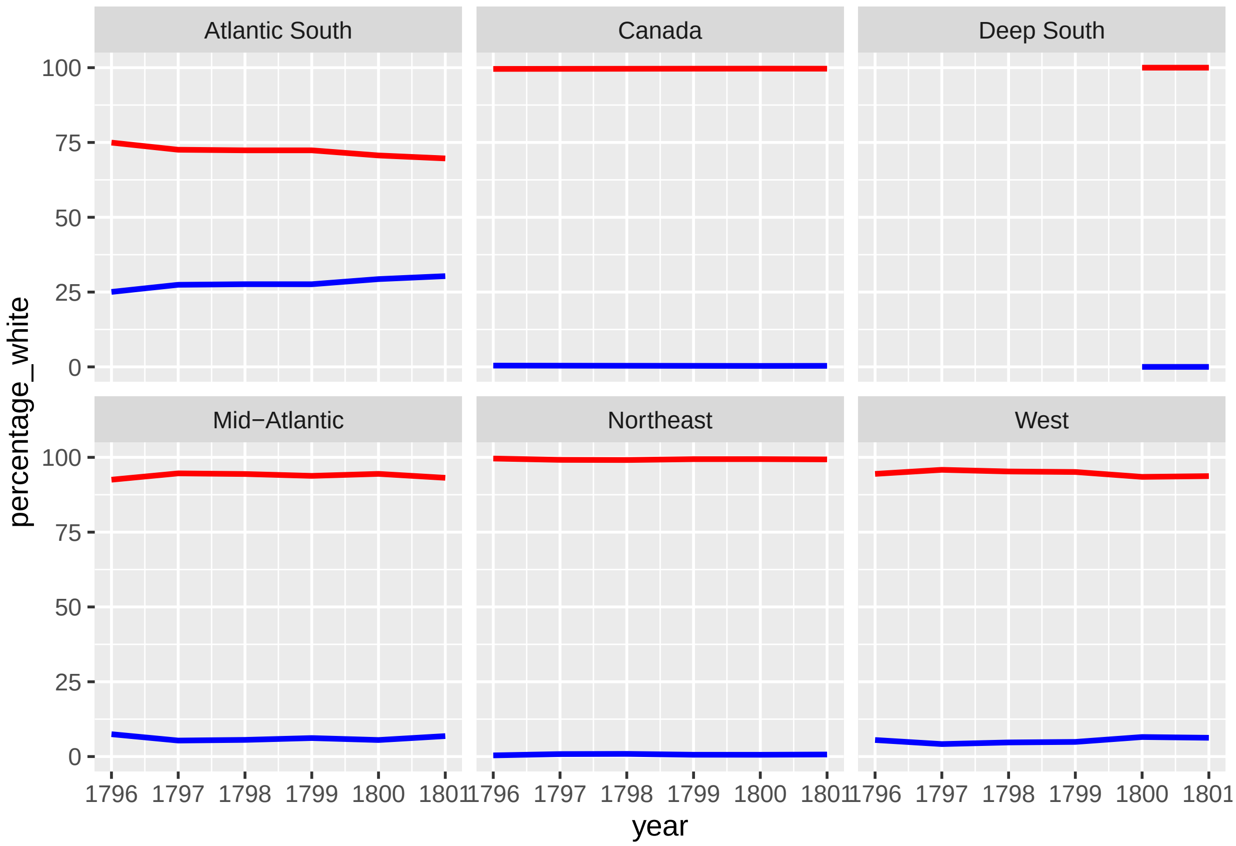
```r
racial_composition
```

```
## # A tibble: 31 x 8
## # Groups:   year [6]
##     year region       members_total members_white members_black percentage_white
##    <int> <chr>                <int>         <int>         <int>            <dbl>
##  1  1796 Atlantic So~         42059         31521         10538             74.9
##  2  1796 Canada                 474           472             2             99.6
##  3  1796 Mid-Atlantic          9406          8703           703             92.5
##  4  1796 Northeast             2519          2509            10             99.6
##  5  1796 West                  2296          2169           127             94.5
##  6  1797 Atlantic So~         41976         30459         11517             72.6
##  7  1797 Mid-Atlantic         10513          9950           563             94.6
##  8  1797 Northeast             2999          2974            25             99.2
##  9  1797 West                  2373          2274            99             95.8
## 10  1798 Atlantic So~         41822         30267         11555             72.4
## # ... with 21 more rows, and 2 more variables: percentage_black <dbl>,
## #   relation <dbl>
```

```r
ggplot(racial_composition) +
geom_line(aes(x = year, y = relation)) + facet_wrap(~ region)
```




```r
ggplot(racial_composition) +
  geom_line(aes(x=year,y=percentage_white), size=1, col="red") +
  geom_line(aes(x=year,y=percentage_black), size=1, col="blue") +
  facet_wrap(~ region)
```



(2) In the europop package there are two data frames, `europop` with the historical populations of European cities, and `city_coords` which has the latitudes and longitudes of those cities. Load that package and join the two tables together. Can you get the populations of cities north of 48° of latitude?

```
#devtools::install_github("mdlincoln/europop", force=TRUE)
library(europop)
data("europop")

merged_cities <- europop %>%
  left_join(city_coords, by="city") %>%
  filter(lat>=42)


merged_cities[merged_cities$lat > 42, ]
```

```
## # A tibble: 2,128 x 6
##    city       region            year population   lon   lat
##    <chr>      <chr>            <int>      <int> <dbl> <dbl>
##  1 BERGEN     Scandinavia       1500          0  5.33  60.4
##  2 COPENHAGEN Scandinavia       1500         NA 12.6   55.7
##  3 GOTEBORG   Scandinavia       1500          0 12.0   57.7
##  4 KARLSKRONA Scandinavia       1500          0 15.6   56.2
##  5 OSLO       Scandinavia       1500          0 10.7   59.9
##  6 STOCKHOLM  Scandinavia       1500          0 18.1   59.3
##  7 BATH       England and Wales  1500          0 -2.36  51.4
##  8 BIRMINGHAM England and Wales  1500          0 -1.90  52.5
##  9 BLACKBURN  England and Wales  1500          0 -2.48  53.8
## 10 BOLTON     England and Wales  1500          0 -2.43  53.6
```

```
## # ... with 2,118 more rows
```

(3) In the historydata package there are two tables, `judges_people` and `judges_appointments`. Join them together. What are the names of black judges who were appointed to the Supreme Court?

```
judge_merge <- judges_people %>%
  left_join(judges_appointments, by="judge_id")

judge_merge[judge_merge$court_name == "Supreme Court of the United States" & judge_merge$race == "Africa
```

```
## # A tibble: 2 x 27
##   judge_id name_first name_middle name_last name_suffix birth_date
##      <int> <chr>      <chr>       <chr>     <chr>            <int>
## 1     1489 Thurgood   <NA>        Marshall  <NA>              1908
## 2     2362 Clarence   <NA>        Thomas    <NA>              1948
## # ... with 21 more variables: birthplace_city <chr>, birthplace_state <chr>,
## #   death_date <int>, death_city <chr>, death_state <chr>, gender <chr>,
## #   race <chr>, court_name <chr>, court_type <chr>, president_name <chr>,
## #   president_party <chr>, nomination_date <chr>, predecessor_last_name <chr>,
## #   predecessor_first_name <chr>, senate_confirmation_date <chr>,
## #   commission_date <chr>, chief_judge_begin <int>, chief_judge_end <int>,
## #   retirement_from_active_service <chr>, termination_date <chr>, ...
```

(4) What courts did those justices serve on before the Supreme Court?

```
filter(judge_merge, judge_id == 1489 | judge_id == 2362, court_name != "Supreme Court of the United Sta
```

```
## # A tibble: 2 x 27
##   judge_id name_first name_middle name_last name_suffix birth_date
##      <int> <chr>      <chr>       <chr>     <chr>            <int>
## 1     1489 Thurgood   <NA>        Marshall  <NA>              1908
## 2     2362 Clarence   <NA>        Thomas    <NA>              1948
## # ... with 21 more variables: birthplace_city <chr>, birthplace_state <chr>,
## #   death_date <int>, death_city <chr>, death_state <chr>, gender <chr>,
## #   race <chr>, court_name <chr>, court_type <chr>, president_name <chr>,
## #   president_party <chr>, nomination_date <chr>, predecessor_last_name <chr>,
## #   predecessor_first_name <chr>, senate_confirmation_date <chr>,
## #   commission_date <chr>, chief_judge_begin <int>, chief_judge_end <int>,
## #   retirement_from_active_service <chr>, termination_date <chr>, ...
```

Data reshaping (`spread()` and `gather()`)

It can be helpful to think of tabular data as coming in two forms: wide data, and long data. Let's load in a table of data. This data contains total membership figures for the Virginia conference of the Methodist Episcopal Church for the years 1812 to 1830.

```
va_wide <- read_csv("http://dh-r.lincolnmullen.com/data/va-methodists-wide.csv")
va_wide
```

```
## # A tibble: 10 x 21
##    conference district   `1812` `1813` `1814` `1815` `1816` `1817` `1818` `1819`
##    <chr>      <chr>       <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 Virginia   James Riv~   5348   4691   4520   4209   4118   3888   3713   3580
##  2 Virginia   Meherren     4882   4486   4771   4687   4702     NA     NA     NA
##  3 Virginia   Meherrin       NA     NA     NA     NA     NA   4435   3964   3860
##  4 Virginia   Neuse          NA     NA   3474   3475   3448   2702   3340   4667
##  5 Virginia   Newbern      3511   3558     NA     NA     NA     NA     NA     NA
```

```
## 6 Virginia   Norfolk      4686   6196   6127   6001   5661   6495   6471     NA
## 7 Virginia   Raleigh      3822   4018     NA     NA     NA     NA     NA     NA
## 8 Virginia   Roanoke       NA     NA     NA     NA   3049     NA   1507     NA
## 9 Virginia   Tar River     NA     NA   3834   3466     NA     NA     NA     NA
## 10 Virginia  Yadkin       3174   3216   3528   3323   3374   3323   4689   4547
## # ... with 11 more variables: `1820` <dbl>, `1821` <dbl>, `1822` <dbl>,
## #   `1823` <dbl>, `1824` <dbl>, `1825` <dbl>, `1826` <dbl>, `1827` <dbl>,
## #   `1828` <dbl>, `1829` <dbl>, `1830` <dbl>
```

The first thing we can notice about this data frame is that it is very wide because it has a column for each of the years. The data is also suitable for reading because it like a table in a publication. We can read from left to right and see when certain districts begin and end and get the values for each year. The difficulties of computing on or plotting the data will also become quickly apparent. How would you make a plot of the change over time in the number of members in each district? Or how would you filter by year, or summarize by year? For that matter, what do the numbers in the table represent, since they are not given an explicit variable name?

The problem with the table is that it is not *tidy data*, because the variables are not in columns and observations in rows. One of the variables is the year, but its values are in the column headers. And another of the variables is total membership, but its values are spread across rows and columns and it is not explicitly named.

The `gather()` function from the tidyr package lets us turn wide data into long data. We need to tell the function two kinds of information. First we need to tell it the name of the column to create from the column headers and the name of the implicit variable in the rows. In the example below, we create to new columns `minutes_year` and `total_membership`. Then we also have to tell the function if there are any columns which should remain unchanged. In this case, the `conference` and `district` variables should remain the same, so we remove them from the gathering using the same syntax as the `select()` function.

```
va_wide %>%
  gather(year, members_total, -conference, -district)
```

```
## # A tibble: 190 x 4
##    conference district    year  members_total
##    <chr>      <chr>       <chr>         <dbl>
##  1 Virginia   James River 1812           5348
##  2 Virginia   Meherren    1812           4882
##  3 Virginia   Meherrin    1812             NA
##  4 Virginia   Neuse       1812             NA
##  5 Virginia   Newbern     1812           3511
##  6 Virginia   Norfolk     1812           4686
##  7 Virginia   Raleigh     1812           3822
##  8 Virginia   Roanoke     1812             NA
##  9 Virginia   Tar River   1812             NA
## 10 Virginia   Yadkin      1812           3174
## # ... with 180 more rows
```

We can see the results above. There are two ways that this result is not quite what we want. Because the years were column headers they are treated as character vectors rather than integers. We can manually convert them in a later step, but we can also let `gather()` do the right thing with the `convert =` argument. Then we have a lot of `NA` values which were explicit in the wide table but which can be removed from the long table with `na.rm =`.

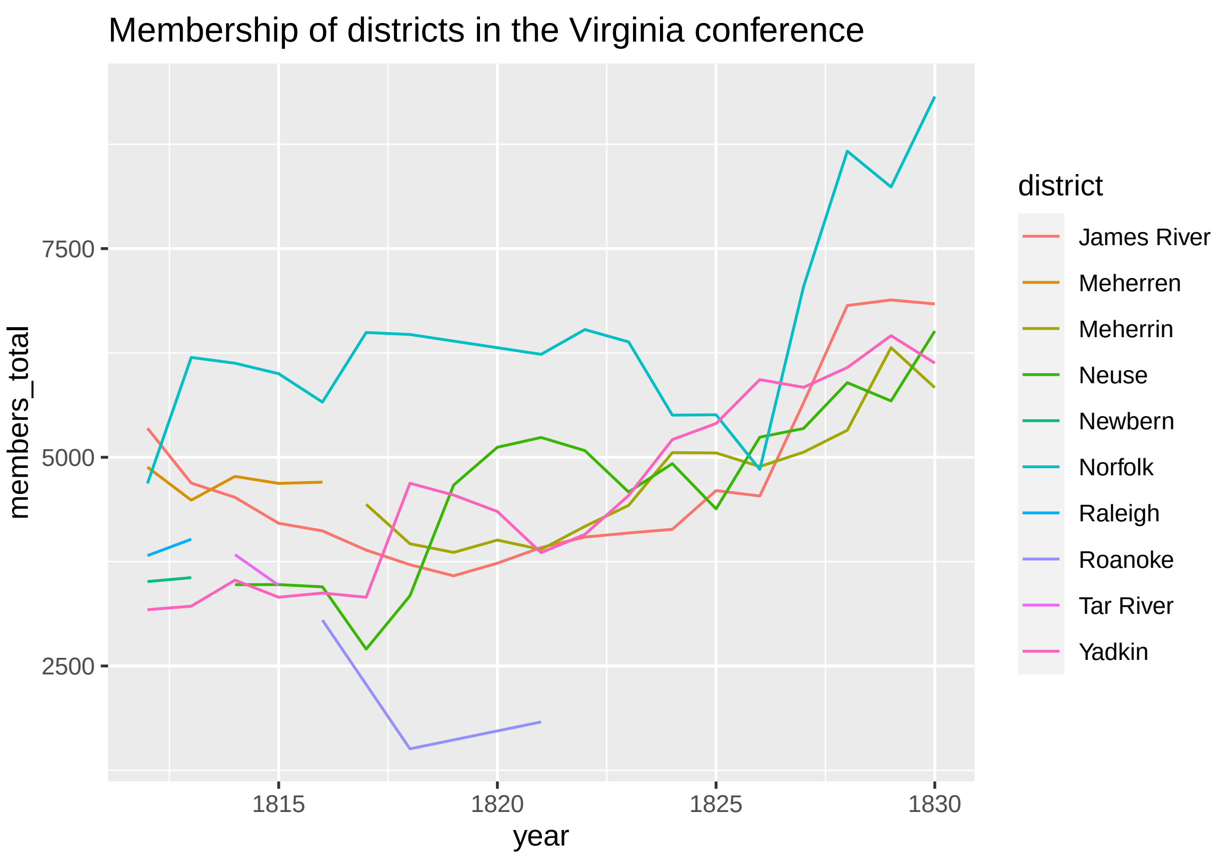
```
va_long <- va_wide %>%
  gather(year, members_total, -conference, -district,
         convert = TRUE, na.rm = TRUE)
```

```r
va_long
```

```
## # A tibble: 100 x 4
##    conference district     year members_total
##    <chr>      <chr>       <int>         <dbl>
##  1 Virginia   James River  1812          5348
##  2 Virginia   Meherren     1812          4882
##  3 Virginia   Newbern      1812          3511
##  4 Virginia   Norfolk      1812          4686
##  5 Virginia   Raleigh      1812          3822
##  6 Virginia   Yadkin       1812          3174
##  7 Virginia   James River  1813          4691
##  8 Virginia   Meherren     1813          4486
##  9 Virginia   Newbern      1813          3558
## 10 Virginia   Norfolk      1813          6196
## # ... with 90 more rows
```

Notice that now we can use the data in ggplot2 without any problem.

```r
ggplot(va_long,
       aes(x = year, y = members_total, color = district)) +
  geom_line() +
  ggtitle("Membership of districts in the Virginia conference")
```



The inverse operation of `gather()` is `spread()`. With `spread()` we specify the name of the column which should become the new column headers (in this case `minutes_year`), and then the name of the column to fill in underneath those new column headers (in this case, `total_membership`). We can see the results below.

```
va_wide2 <- va_long %>%
  spread(year, members_total)

va_wide2
```

```
## # A tibble: 10 x 21
##    conference district   `1812` `1813` `1814` `1815` `1816` `1817` `1818` `1819`
##    <chr>      <chr>       <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 Virginia   James Riv~   5348   4691   4520   4209   4118   3888   3713   3580
##  2 Virginia   Meherren     4882   4486   4771   4687   4702     NA     NA     NA
##  3 Virginia   Meherrin       NA     NA     NA     NA     NA   4435   3964   3860
##  4 Virginia   Neuse          NA     NA   3474   3475   3448   2702   3340   4667
##  5 Virginia   Newbern      3511   3558     NA     NA     NA     NA     NA     NA
##  6 Virginia   Norfolk      4686   6196   6127   6001   5661   6495   6471     NA
##  7 Virginia   Raleigh      3822   4018     NA     NA     NA     NA     NA     NA
##  8 Virginia   Roanoke        NA     NA     NA     NA   3049     NA   1507     NA
##  9 Virginia   Tar River      NA     NA   3834   3466     NA     NA     NA     NA
## 10 Virginia   Yadkin       3174   3216   3528   3323   3374   3323   4689   4547
## # ... with 11 more variables: `1820` <dbl>, `1821` <dbl>, `1822` <dbl>,
## #   `1823` <dbl>, `1824` <dbl>, `1825` <dbl>, `1826` <dbl>, `1827` <dbl>,
## #   `1828` <dbl>, `1829` <dbl>, `1830` <dbl>
```

By looking at the data we can see that we got back to where we started.

Turning long data into wide is often useful when you want to create a tabular representation of data. (And once you have a data frame that can be a table, the `knitr::kable()` function is quite nice.) And some algorithms, such as clustering algorithms, expect wide data rather than tidy data.

For the exercise, we will use summary statistics of the number of white and black members in the Methodists by year.

```
methodists_by_year_race <- methodists %>%
  group_by(year) %>%
  summarize(white = sum(members_white, na.rm = TRUE),
            black = sum(members_black, na.rm = TRUE),
            indian = sum(members_indian, na.rm = TRUE))
methodists_by_year_race
```

```
## # A tibble: 49 x 4
##     year white black indian
##    <int> <int> <int>  <int>
##  1  1786 18291  2890      0
##  2  1787 21949  3883      0
##  3  1788 30557  7991      0
##  4  1789 34425  8840      0
##  5  1790 45983 11682      0
##  6  1791 50580 13098      0
##  7  1792 52079 13871      0
##  8  1793 51486 14420      0
##  9  1794 52794 13906      0
## 10  1795 48121 12171      0
## # ... with 39 more rows
```

(5) The data in `methodists_by_year_race` could be tidier still. While `white`, `black`, and `indian` are variables, it is perhaps better to think of them as two different variables. One variable would be `race`, containing the racial descriptions that the Methodists used, and another would be `members`, containing

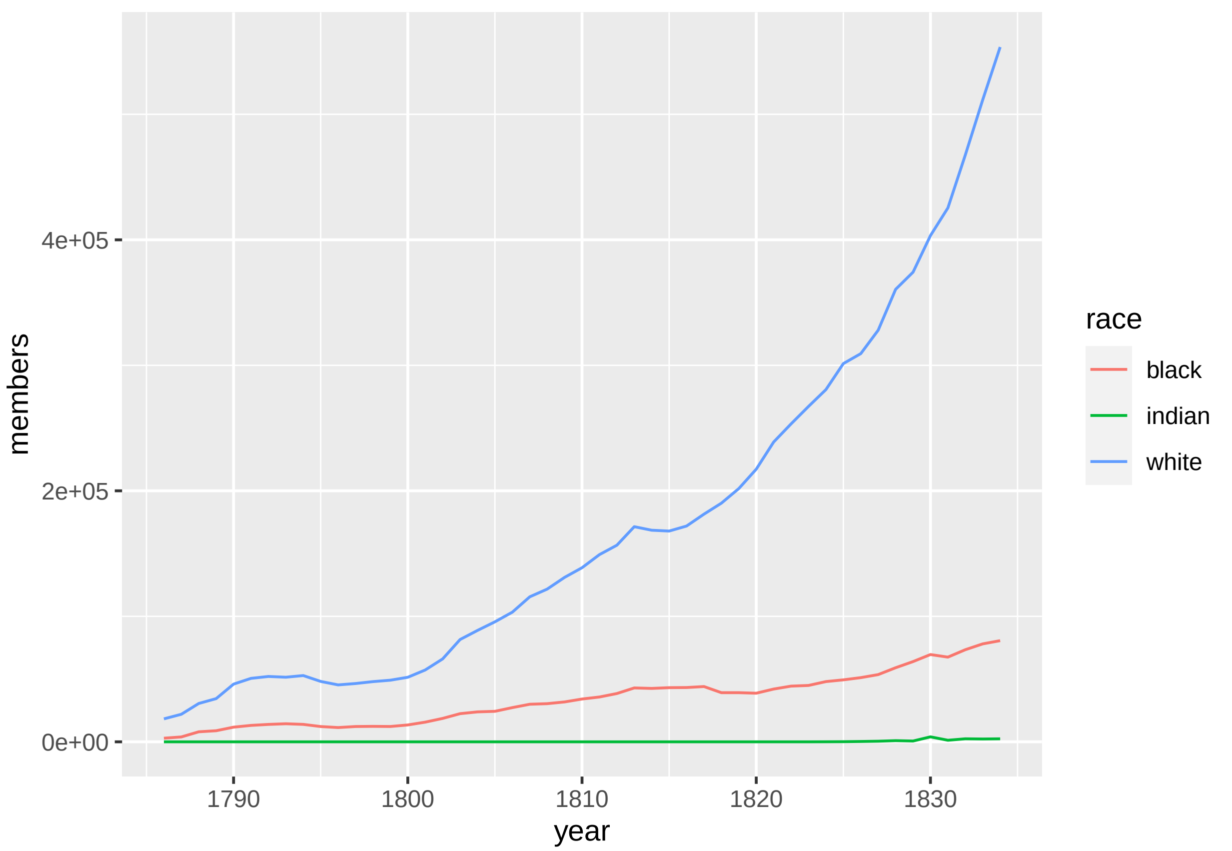
the number of members. Using the `gather()` function, create that data frame.

```
temp <- methodists_by_year_race %>%
  gather(race, members, -year)

temp
```

```
## # A tibble: 147 x 3
##     year race  members
##    <int> <chr>   <int>
##  1  1786 white   18291
##  2  1787 white   21949
##  3  1788 white   30557
##  4  1789 white   34425
##  5  1790 white   45983
##  6  1791 white   50580
##  7  1792 white   52079
##  8  1793 white   51486
##  9  1794 white   52794
## 10  1795 white   48121
## # ... with 137 more rows
```

(6) Use the data frame you created in the previous step to create a line plot of membership over time, mapping the `race` column to the `color` aesthetic.

```
ggplot(temp,
       aes(x = year, y = members, color = race)) +
  geom_line()
```



(7) Now use that newly tidied data frame to create a wide data frame, where the years are the column headers and the racial descriptions are the rows.

```
temp2 <- temp %>%
  spread(year, members)

temp2
```

```
## # A tibble: 3 x 50
##   race   `1786` `1787` `1788` `1789` `1790` `1791` `1792` `1793` `1794` `1795`
##   <chr>   <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>
## 1 black    2890   3883   7991   8840  11682  13098  13871  14420  13906  12171
## 2 indian      0      0      0      0      0      0      0      0      0      0
## 3 white   18291  21949  30557  34425  45983  50580  52079  51486  52794  48121
## # ... with 39 more variables: `1796` <int>, `1797` <int>, `1798` <int>,
## #   `1799` <int>, `1800` <int>, `1801` <int>, `1802` <int>, `1803` <int>,
## #   `1804` <int>, `1805` <int>, `1806` <int>, `1807` <int>, `1808` <int>,
## #   `1809` <int>, `1810` <int>, `1811` <int>, `1812` <int>, `1813` <int>,
## #   `1814` <int>, `1815` <int>, `1816` <int>, `1817` <int>, `1818` <int>,
## #   `1819` <int>, `1820` <int>, `1821` <int>, `1822` <int>, `1823` <int>,
## #   `1824` <int>, `1825` <int>, `1826` <int>, `1827` <int>, `1828` <int>, ...
```

(8) Now use the same tidied data to create a wide data frame where the racial descriptions are column headers and the years are rows.

```
temp3 <- temp %>%
  spread(race, members)
temp3
```

```
## # A tibble: 49 x 4
##     year black indian white
##    <int> <int>  <int> <int>
##  1  1786  2890      0 18291
##  2  1787  3883      0 21949
##  3  1788  7991      0 30557
##  4  1789  8840      0 34425
##  5  1790 11682      0 45983
##  6  1791 13098      0 50580
##  7  1792 13871      0 52079
##  8  1793 14420      0 51486
##  9  1794 13906      0 52794
## 10  1795 12171      0 48121
## # ... with 39 more rows
```