# Loading Data into R

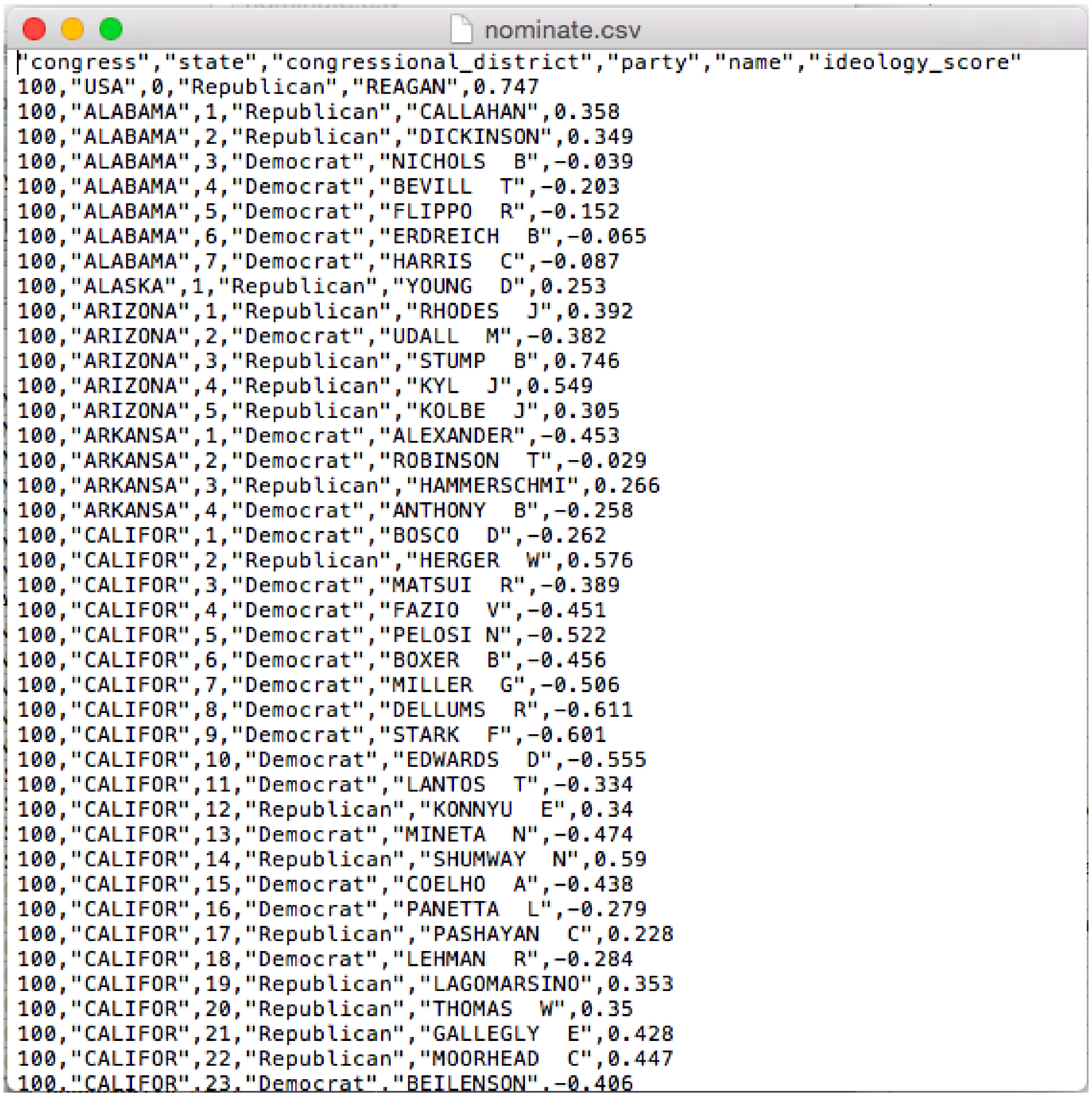
## Loading Data Sets

Rather than manually entering data using `c()` or something else, we'll want to load data in stored in a data file. For this class, these will usually be one of three types:

1. R data or `.Rds` files. This is the easiest format because it stores factors as factors and all the related information. Read with `readRDS()`.
2. comma-separated value or `.csv` files. This is a common data format. However, it stores factors as character strings, so the data file does not contain the information about factors. Read with `readr::read_csv()`.
3. Stata or `.dta` files. Another common data format because many political scientists use Stata. Read these files into R with `haven::read_dta()`.

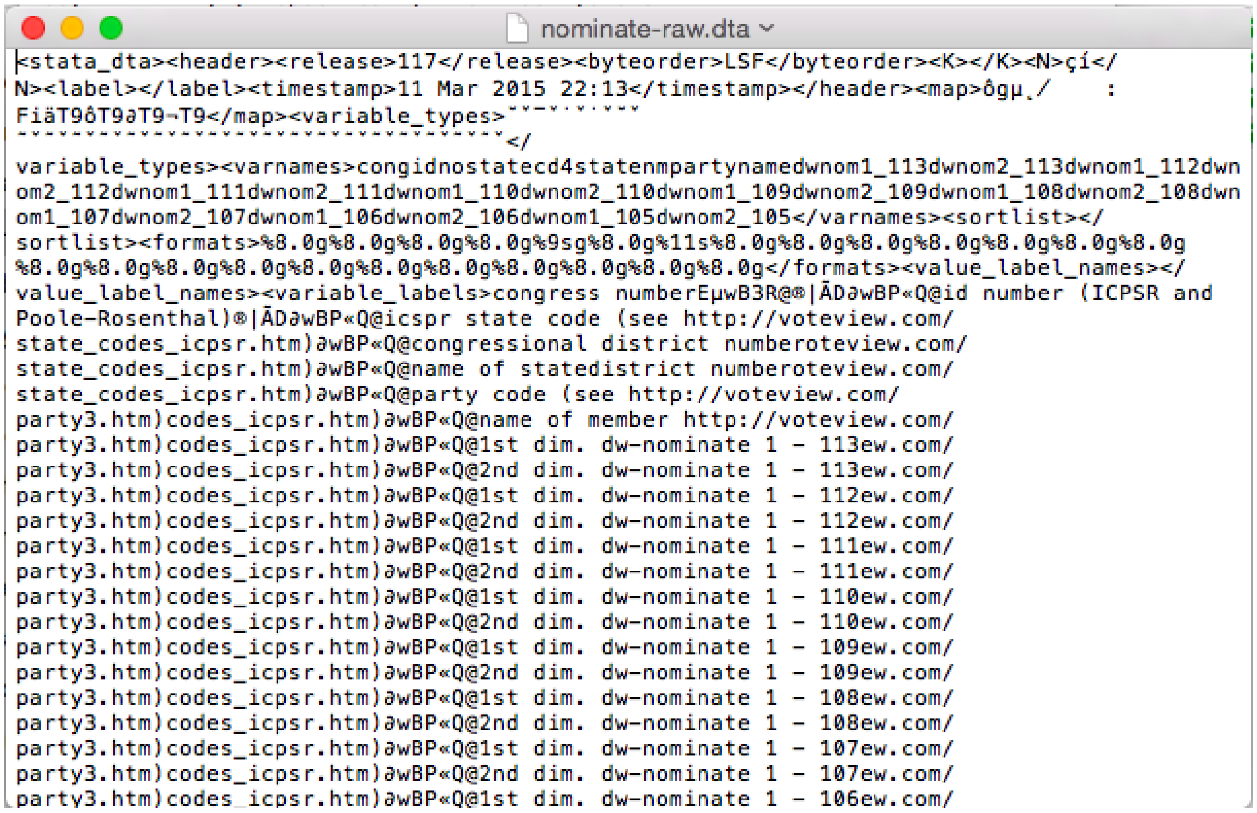
### Comma-Separated Value Format (`.csv`)

Data can be stored in a wide range of formats. One popular format, for example, is Stata's proprietary `.dta` format. I typically use (and encourage you to use) the comma-separated values `.csv` format. The `.csv` format is excellent because it is open and simple. This means that anyone can use it without acess to proprietary software. It will also be useble by anyone into the foreseeable future. We can see why `.csv` files are easy to work with if we open it up the file `nominate.csv` with a text editor. You'll see that you can read the file directly. You don't really need software, you can do it with your eyes!


nominate.csv

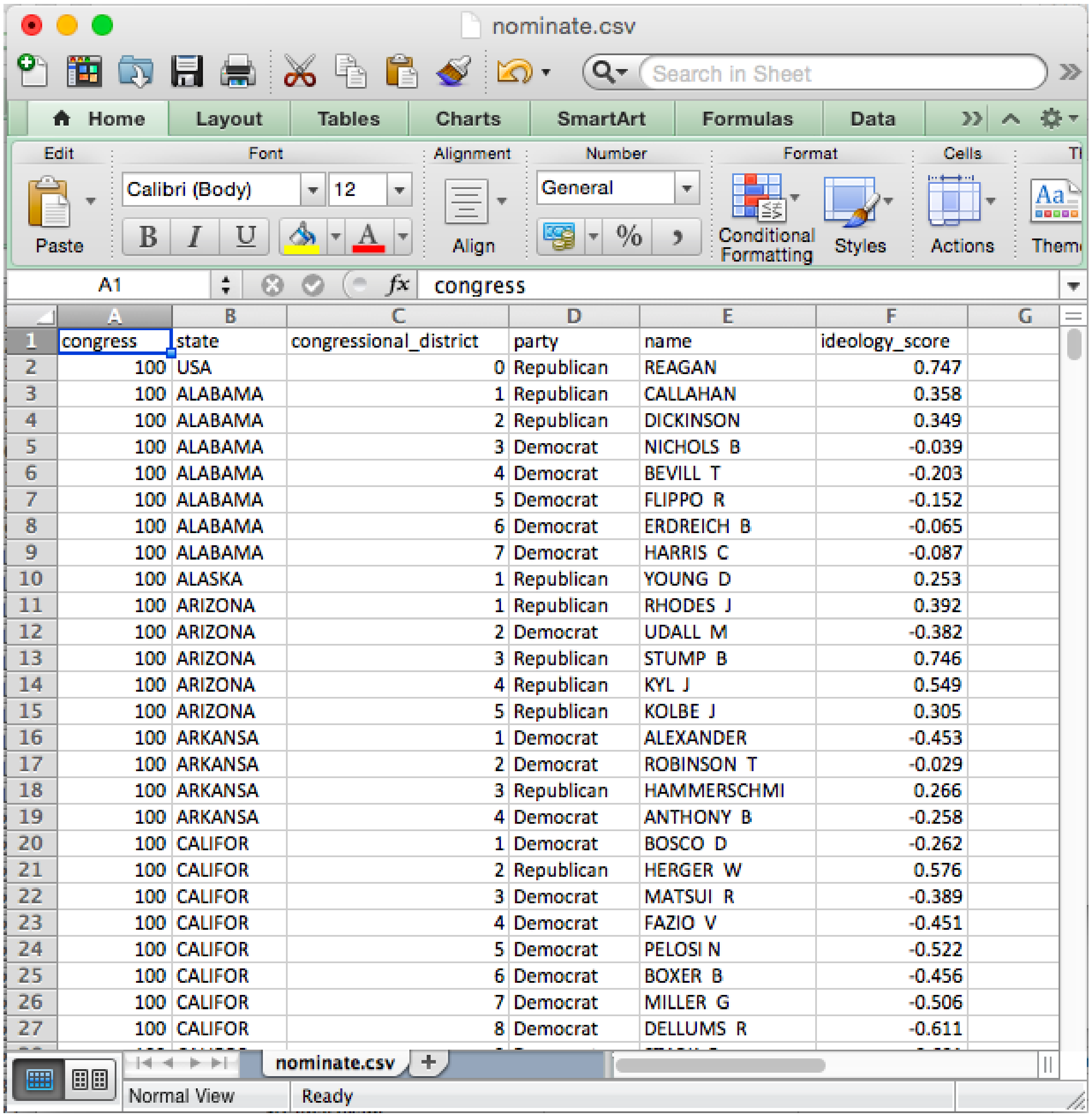
```
"congress","state","congressional_district","party","name","ideology_score"
100,"USA",0,"Republican","REAGAN",0.747
100,"ALABAMA",1,"Republican","CALLAHAN",0.358
100,"ALABAMA",2,"Republican","DICKINSON",0.349
100,"ALABAMA",3,"Democrat","NICHOLS  B",-0.039
100,"ALABAMA",4,"Democrat","BEVILL  T",-0.203
100,"ALABAMA",5,"Democrat","FLIPPO  R",-0.152
100,"ALABAMA",6,"Democrat","ERDREICH  B",-0.065
100,"ALABAMA",7,"Democrat","HARRIS  C",-0.087
100,"ALASKA",1,"Republican","YOUNG  D",0.253
100,"ARIZONA",1,"Republican","RHODES  J",0.392
100,"ARIZONA",2,"Democrat","UDALL  M",-0.382
100,"ARIZONA",3,"Republican","STUMP  B",0.746
100,"ARIZONA",4,"Republican","KYL  J",0.549
100,"ARIZONA",5,"Republican","KOLBE  J",0.305
100,"ARKANSA",1,"Democrat","ALEXANDER",-0.453
100,"ARKANSA",2,"Democrat","ROBINSON  T",-0.029
100,"ARKANSA",3,"Republican","HAMMERSCHMI",0.266
100,"ARKANSA",4,"Democrat","ANTHONY  B",-0.258
100,"CALIFOR",1,"Democrat","BOSCO  D",-0.262
100,"CALIFOR",2,"Republican","HERGER  W",0.576
100,"CALIFOR",3,"Democrat","MATSUI  R",-0.389
100,"CALIFOR",4,"Democrat","FAZIO  V",-0.451
100,"CALIFOR",5,"Democrat","PELOSI N",-0.522
100,"CALIFOR",6,"Democrat","BOXER  B",-0.456
100,"CALIFOR",7,"Democrat","MILLER  G",-0.506
100,"CALIFOR",8,"Democrat","DELLUMS  R",-0.611
100,"CALIFOR",9,"Democrat","STARK  F",-0.601
100,"CALIFOR",10,"Democrat","EDWARDS  D",-0.555
100,"CALIFOR",11,"Democrat","LANTOS  T",-0.334
100,"CALIFOR",12,"Republican","KONNYU  E",0.34
100,"CALIFOR",13,"Democrat","MINETA  N",-0.474
100,"CALIFOR",14,"Republican","SHUMWAY  N",0.59
100,"CALIFOR",15,"Democrat","COELHO  A",-0.438
100,"CALIFOR",16,"Democrat","PANETTA  L",-0.279
100,"CALIFOR",17,"Republican","PASHAYAN  C",0.228
100,"CALIFOR",18,"Democrat","LEHMAN  R",-0.284
100,"CALIFOR",19,"Republican","LAGOMARSINO",0.353
100,"CALIFOR",20,"Republican","THOMAS  W",0.35
100,"CALIFOR",21,"Republican","GALLEGLY  E",0.428
100,"CALIFOR",22,"Republican","MOORHEAD  C",0.447
100,"CALIFOR",23,"Democrat","BEILENSON",-0.406
```

I tried the same thing for a similar `.dta` file. You can see that it looks like nonsense. You'll definitely need Stata (or other speciallized software) to work with this file.



```
<stata_dta><header><release>117</release><byteorder>LSF</byteorder><K></K><N>çí</
N><label></label><timestamp>11 Mar 2015 22:13</timestamp></header><map>ôgµ¸/    :
FiäT9ôT9∂T9¬T9</map><variable_types>ˇˇ˜ˇ˙ˇ˙ˇˇˇ
ˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇˇ</
variable_types><varnames>congidnostatecd4statenmpartynamedwnom1_113dwnom2_113dwnom1_112dwn
om2_112dwnom1_111dwnom2_111dwnom1_110dwnom2_110dwnom1_109dwnom2_109dwnom1_108dwnom2_108dwn
om1_107dwnom2_107dwnom1_106dwnom2_106dwnom1_105dwnom2_105</varnames><sortlist></
sortlist><formats>%8.0g%8.0g%8.0g%8.0g%9sg%8.0g%11s%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g
%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g%8.0g</formats><value_label_names></
value_label_names><variable_labels>congress numberEµwB3R@®|ĀD∂wBP«Q@id number (ICPSR and
Poole-Rosenthal)®|ĀD∂wBP«Q@icspr state code (see http://voteview.com/
state_codes_icpsr.htm)∂wBP«Q@congressional district numberoteview.com/
state_codes_icpsr.htm)∂wBP«Q@name of statedistrict numberoteview.com/
state_codes_icpsr.htm)∂wBP«Q@party code (see http://voteview.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@name of member http://voteview.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 113ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@2nd dim. dw-nominate 1 - 113ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 112ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@2nd dim. dw-nominate 1 - 112ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 111ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@2nd dim. dw-nominate 1 - 111ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 110ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@2nd dim. dw-nominate 1 - 110ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 109ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@2nd dim. dw-nominate 1 - 109ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 108ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@2nd dim. dw-nominate 1 - 108ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 107ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@2nd dim. dw-nominate 1 - 107ew.com/
party3.htm)codes_icpsr.htm)∂wBP«Q@1st dim. dw-nominate 1 - 106ew.com/
```

Also, `.csv` files are easy to support, so they work in almost all data analysis software. For example, we can open up `nominate.csv` in Excel. You can see that we have six variables in the columns and many cases in the rows (we don't know how many because they overflow the screen). In this case, each row represents a particular Congressman or Congresswoman from a particular Congress (and Presidents are also included). The second row, for example, is for Rep. Callahan (R) from the 1st Congressional District of Alabama. During the 100th Congress, Rep. Calahan has a ideology score of 0.358, which means he's conservative, but not as conservative as Pres. Reagan, who has a score of 0.747. We'll work with these data a lot thoughout the semester, so we'll have plenty of time for closer examination.



| congress | state | congressional_district | party | name | ideology_score |
|---|---|---|---|---|---|
| 100 | USA | 0 | Republican | REAGAN | 0.747 |
| 100 | ALABAMA | 1 | Republican | CALLAHAN | 0.358 |
| 100 | ALABAMA | 2 | Republican | DICKINSON | 0.349 |
| 100 | ALABAMA | 3 | Democrat | NICHOLS B | -0.039 |
| 100 | ALABAMA | 4 | Democrat | BEVILL T | -0.203 |
| 100 | ALABAMA | 5 | Democrat | FLIPPO R | -0.152 |
| 100 | ALABAMA | 6 | Democrat | ERDREICH B | -0.065 |
| 100 | ALABAMA | 7 | Democrat | HARRIS C | -0.087 |
| 100 | ALASKA | 1 | Republican | YOUNG D | 0.253 |
| 100 | ARIZONA | 1 | Republican | RHODES J | 0.392 |
| 100 | ARIZONA | 2 | Democrat | UDALL M | -0.382 |
| 100 | ARIZONA | 3 | Republican | STUMP B | 0.746 |
| 100 | ARIZONA | 4 | Republican | KYL J | 0.549 |
| 100 | ARIZONA | 5 | Republican | KOLBE J | 0.305 |
| 100 | ARKANSA | 1 | Democrat | ALEXANDER | -0.453 |
| 100 | ARKANSA | 2 | Democrat | ROBINSON T | -0.029 |
| 100 | ARKANSA | 3 | Republican | HAMMERSCHMI | 0.266 |
| 100 | ARKANSA | 4 | Democrat | ANTHONY B | -0.258 |
| 100 | CALIFOR | 1 | Democrat | BOSCO D | -0.262 |
| 100 | CALIFOR | 2 | Republican | HERGER W | 0.576 |
| 100 | CALIFOR | 3 | Democrat | MATSUI R | -0.389 |
| 100 | CALIFOR | 4 | Democrat | FAZIO V | -0.451 |
| 100 | CALIFOR | 5 | Democrat | PELOSI N | -0.522 |
| 100 | CALIFOR | 6 | Democrat | BOXER B | -0.456 |
| 100 | CALIFOR | 7 | Democrat | MILLER G | -0.506 |
| 100 | CALIFOR | 8 | Democrat | DELLUMS R | -0.611 |

## Setting the Working Directory

Each project you work on should have it's own directory (i.e., folder). For example, I have a folder for each class I teach and each paper I write. I have code and data for each class I teach and each paper I write. At the beginning of my R session, I simply set the working directory to the directory for the class or paper that I want to work on. To set the working directory in RStudio, click "Session," "Set Working Directory," "Choose Directory." Then naviagte to and choose the directory of the project (e.g., class or paper) you want to work on.

But what should a project folder look like? It should be very well organized. I recommend you have two for this class, one for the class material (e.g., notes, data, code, etc.) and another for the paper you'll write. I recommend something like the following structure for the class directory.

```
class-folder
  |--data
  |--notes
  |--R
  |--readings
```

You can name `class-folder` anything you like, but I recommend not using spaces. You can include other subdirectories as you like, but the important point is to have a `data` subdirectory in which you keep all the data that you load into R. The directory `R` holds all your R code for the class, named informatively (e.g., `intro-to-R.R`, `histograms.R`). You can keep any notes in the `notes` directory and any readings in the `readings` directory.

And I recommend something like the following for your papers.

```
project-folder
  |--data
  |--R
  |--doc
    |--figs
  |--lit
```

Again, name `project-folder` something short and descriptive, avoiding spaces. As before, the `data` directory holds the data for your project, the `R` directory holds all the R scripts. The `doc` directory holds the paper, perhaps in Microsoft Word or whatever you use to write papers. I recommend keeping the figures in a separate directory `figs` inside the `doc` directory. You might use a directory `lit` to keep up with any literature you reference as part of your project.

### Reading the Data

Once you've set up your project directory and saved the data to the `data` subdirectory, loading data is a easy, you just need to tell `read_csv()` (in the package `readr`), `readRDS()` (pre-loaded as part of base R), or `read_dta()` (in the package `haven`) where to look for the data. Suppose that we've saved a data set as `cool-data.csv` to `project-folder/data`.

If the data set were in the working directory (`project-folder` in this example), then we could just use `readr::read_csv("cool-data.csv")`. But the data set is not in the working directory, it is in the subdirectory `data`. We just need to include the subdirectory path in the filename, so that `readr::read_csv("data/cool-data.csv")`.

Because we'll set the working directory to the folder dedicated to the project we're working on, and we'll always save data to the subdirectory `data`. This will be really easy.

To practice, go ahead and create a directory for this class. Then download the data set `nominate.csv` from URL and save it as `nominate.csv` in the `data` subdirectory. Open RStudio and set the working directory to your class folder. Now let's read those data into R.

```r
# Before this will work:
# 1.) Save the file nominate.csv to your data subdirectory in your class folder.
# 2.) set the working directory to the class folder.

# load packages
library(readr)  # for read_csv() function
```

```
## Warning: package 'readr' was built under R version 3.2.5
```

```r
library(dplyr)  # for glimpse() function
```

```
## Warning: package 'dplyr' was built under R version 3.2.5
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# read data
nominate <- read_csv("data/nominate.csv")
```

```
## Parsed with column specification:
## cols(
##   congress = col_integer(),
##   state = col_character(),
##   congressional_district = col_integer(),
##   party = col_character(),
##   name = col_character(),
##   ideology_score = col_double()
## )
```

```r
# check that data read properly
glimpse(nominate)  # quick look at the data
```

```
## Observations: 6,159
## Variables: 6
## $ congress               (int) 100, 100, 100, 100, 100, 100, 100, 100,...
## $ state                  (chr) "ALABAMA", "ALABAMA", "ALABAMA", "ALABA...
## $ congressional_district (int) 1, 2, 3, 4, 5, 6, 7, 1, 1, 2, 3, 4, 5, ...
## $ party                  (chr) "Republican", "Republican", "Democrat",...
## $ name                   (chr) "CALLAHAN", "DICKINSON", "NICHOLS  B", ...
## $ ideology_score         (dbl) 0.358, 0.349, -0.039, -0.203, -0.152, -...
```

If you like, you can save the above as an R script `reading-data.R` to your `R` directory for future reference.

**`rio`** Loading data into R is a little bit tricky and tedious. One reason is finding a function to handle the data format. If the data is `.Rds`, `.csv`, or `.dta.` formats, we already know what to do. But what if the data is in a format such as `.tsv` (tab separated), `.xlsx` (Microsoft Excel), `.ods` (OpenDocument spreadsheet), or any number of other formats?

The R package `rio` contains the fuction `import()` that automatically adapts to the different formats according to the filename extension. It's just one function—you simply need to point it to the data set.

```
# Before this will work:
# 1.) Save the files nominate.csv, nominate.rds, and nominate.dta to your data subdirectory in your cla
# 2.) set the working directory to the class folder.

# load packages
library(rio)  # for generic import() function
```

```
## Warning: package 'rio' was built under R version 3.2.5
```

```
# read same data stored in three different formats
nominate <- import("data/nominate.csv")
nominate <- import("data/nominate.rds")
nominate <- import("data/nominate.dta")
```

### Data Frames

Almost the statistical computation we do in this class revolves around data sets. In R, it usually makes sense to store data sets as specific objects known as data frames. Data frames are simply a set of vectors that all contain the same number of elements. These might be numeric, character, factor, or logical vectors, or some mixture of types.

When you read a data set into R using `readr::read_csv`, `readRDS()`, `read_dta()`, or some other method, it creates a data frame. A data frame is a special R object that holds a set of vectors that all have the name number of elements. If you think of the data set as an Excel spreadsheet, then you can think of the columns of the spreadsheet as the vectors held by the data frame. These vectors or variables can be numeric, character, factor, or logical. As a reminder, here are the variable types:

- `numeric`: numbers, such as 1.1, 2.4, and 3.4. Sometimes numeric variables are subdivided into `integer` (whole numbers, e.g., 1, 2, 3, etc.) and `double` (fractions, e.g., 1.47, 3.35462, etc.).
- `character`: text strings, such as `"Republican"` or `"Argentina (2001)"`.
- `factor`: cateogories, such as `"Very Liberal"`, `"Weak Republican"`, or `"Female"`. Similar to `character`, except the entire set of possible levels is defined. A `factor` variable may be ordered or unordered.
- `logical`: true or false, such as `TRUE` or `FALSE`.

For the `.csv` files we will usually use, R cannot distinguish between `character` and `factor` variables. By default, `readr::read_csv()` will load these as `character` variables–there's no way for R to know the entire set of levels from the `.csv` file anyway. Sometimes, though, it will be useful to work with `factor` variables. This is straightforward to change.

**Working with Variables in Data Frames** A data frame holds the variables, but it also hides the vectors. For example, the data frame `nominate`, which we loaded above, has a numeric variable `ideology_score`, but if we try to sum it, we get an error.

```
sum(ideology_score)  # fails because the variable ideology_score is hidden in a data frame
```

```
## Error in eval(expr, envir, enclos): object 'ideology_score' not found
```

We've loaded the data set, but R can't seem to find the variable. That's because the variable `ideology_score` is hidden in the data frame `nominate`.

In order to access variables in data frames, we need to do one of two things.

1. Use the `$` operator.
2. Use the `data` argument.

Some functions, such as `exp()` are designed to work with *vectors*, not data frames. This will be the case for most functions we use (with the notable exceptions of plotting in with `ggplot()` and estimate liner model with `lm()`). To use the functions on variables stored in data frames, we need to use the `$` operator. Suppose we have a data set loaded and given to the object `my_data`. If `my_data` contains the variable of interest `my_variable`, then we can access `my_variable` using the syntax `my_data$my_varible`. That is, the syntax `data$var` means "get the variable `var` from the data set `data`." We'll use this often, so make sure it's clear.

```r
sum(nominate$ideology_score) # example of the $ operator
```

```
## [1] 535.583
```

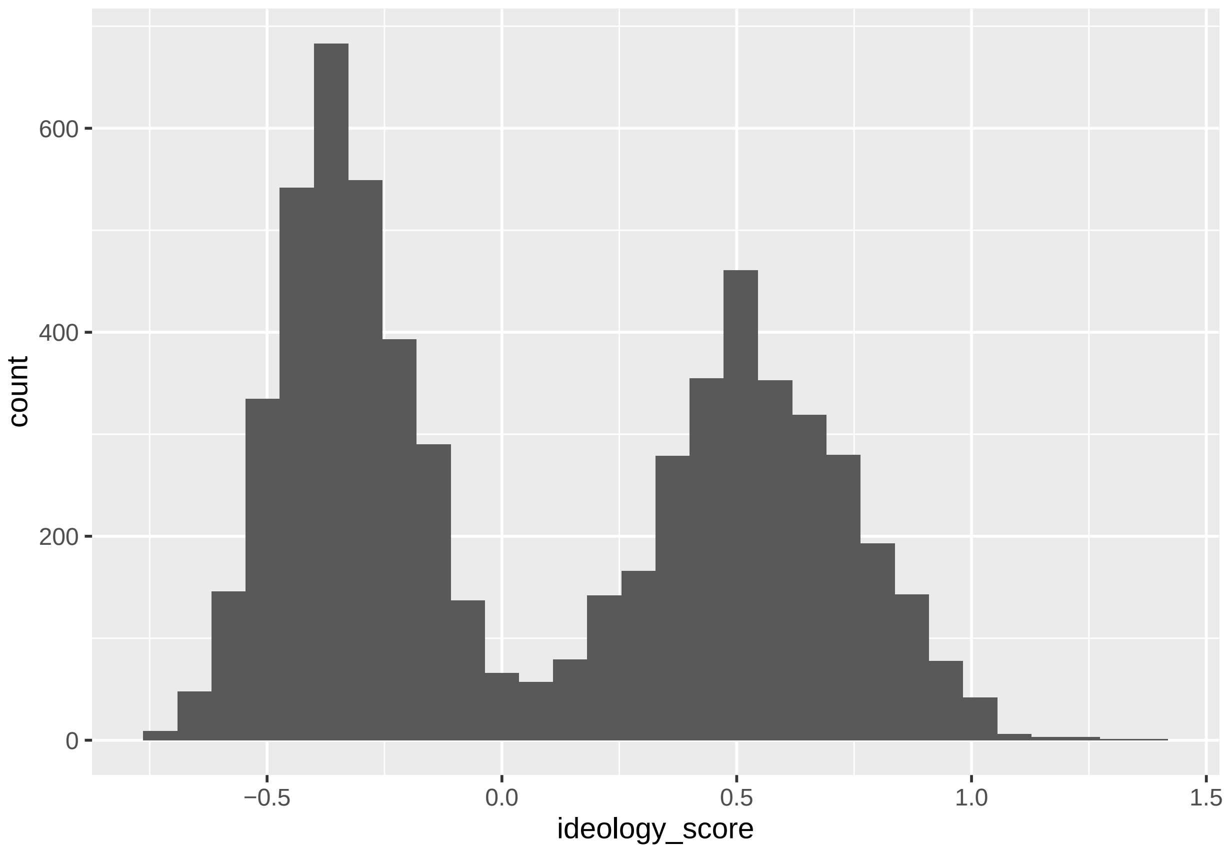
But some functions are designed to work with data frames. Most of the functions we will use are like this. For example, the `qplot()` function in the `ggplot2` package is designed to work with data sets. If you open the help file for `qplot()` (i.e., `help(qplot)` after `library(ggplot2)`), you'll see that one of the arugments is `data`. If you use this argument to point `qplot()` to the data frame, it will know where to find your variables.

```r
# load ggplot2 package, which contains the qplot function
library(ggplot2)
```

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

```r
# example of a function with a data argument
qplot(ideology_score, data = nominate)  # using the data argument
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Many of the functions we use take a data argument. If they do not, though, we'll need to use the `$` operator. Because we'll almost always use data stored in data frames, you need to be sure to use one approach or the other. If the function has a data argument, use it. In many cases, though, we'll need to use the `$` operator.

## How We'll Always Use R

1. Start RStudio.
2. Set the working directory to whatever project you want to work on.
3. Open a new R script to do something new *OR* open a previously saved script to continue making progress.