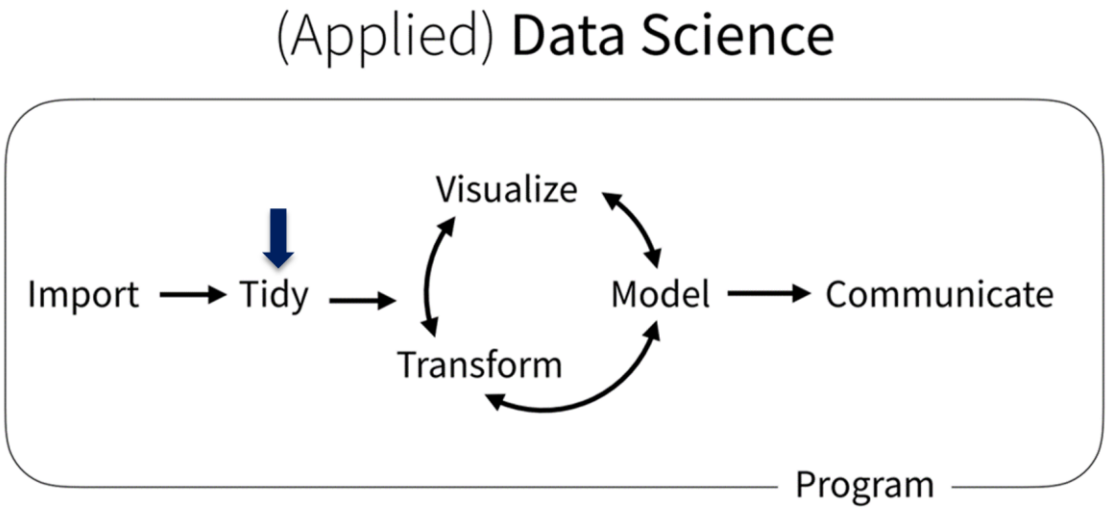
Tidy the data to the correct format

Jeff Wesner

September 20, 2019



4) Tidy the data

Most datasets are formatted for humans to read. But computers want a different format. The next steps will turn your dataset into a format that your computer can read. We refer to it as "tidy data". You can read more about it here (but you don't have to). First we need to install some packages that will make this process easy.

Open a script

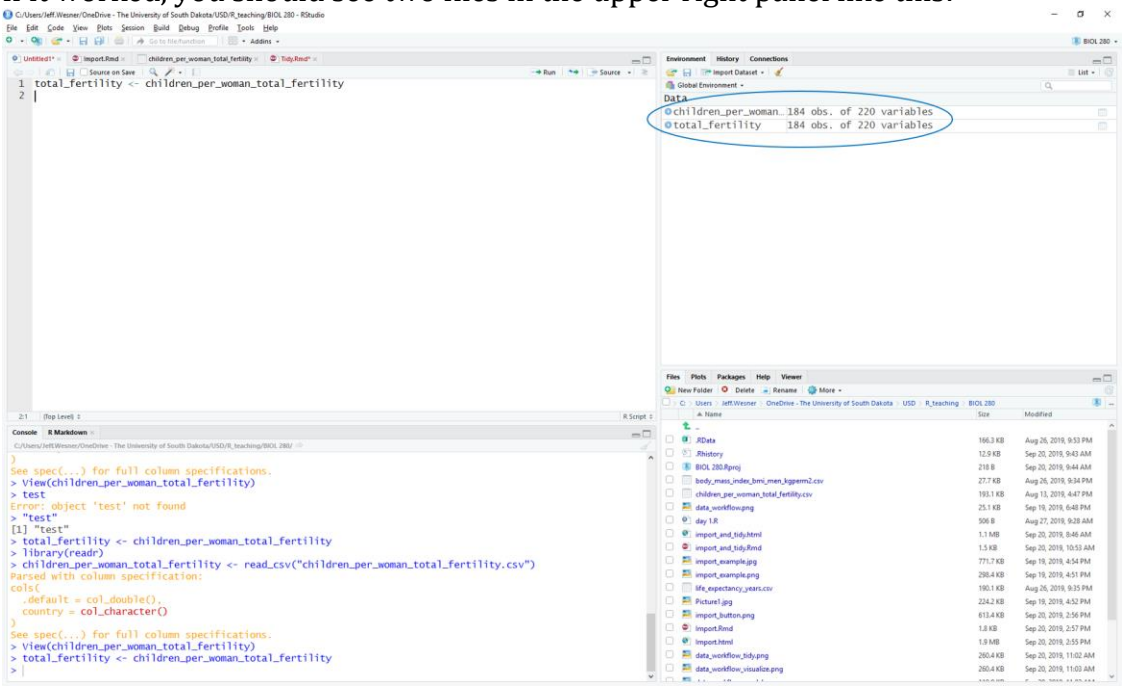
File -> New File -> R Script

You should see a blank file with a flashing prompt. Script's are your friend. It's where you tell the computer what to do. For example, you might want to give your dataset a shorter name, since you'll have to type it over and over again. Paste the code below in your script and click *Run*.

```
total_fertility <- children_per_woman_total_fertility
```

In words, this says "create a new dataset called total_fertility from the other dataset called children_per_woman_total_fertility". The arrow is made with two symbols, a *lesser than* < and a *minus* -, which gives <-.

If it worked, you should see two files in the upper right panel like this:



Install some packages

"Packages" are simply shortcuts for coding that other people have developed and released to the world. R has a huge library of external pakckages. We're going to use two of them - *tidyverse* and *janitor*. Getting these packages involves two steps:

1) Install the packages to your computer. You only have to do this once. Paste the code below in your script and run it.

```
install.packages("tidyverse")
install.packages("janitor")
```

2) Tell R that you want to use the package in this session by typing *library(INSERT PACKAGE NAME)* (replace with the actual package name). Try typing the code below and clicking Run.

```
library(tidyverse)

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

## v ggplot2 3.2.0          v purrr   0.3.2
## v tibble  2.1.3          v dplyr   0.8.3
## v tidyr   0.8.3.9000     v stringr 1.4.0
## v readr   1.3.1          v forcats 0.4.0

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

```
library(janitor)

##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Tidy the data

Your dataset has a column for country and a bunch of other columns for years. The numbers represent the number of children born per woman for a given combination of country and year. You can do a lot with this. For example, the code below will calculate the mean, median, and standard deviation of fertility in the year 1801. The dollar sign just indicates that you want to perform an operation on a certain column within a dataset:

```
mean(total_fertility$'1801') #find the column called 1801 in the dataset
called "total_fertility" and caculate the mean
median(total_fertility$'1801') #find the column called 1801 in the dataset
called "total_fertility" and caculate the mean)
sd(total_fertility$'1801') #find the column called 1801 in the dataset called
"total_fertility" and caculate the mean
```

You can also plot the data like this:

```
plot(total_fertility$'1801')
```

How would you interpret this plot?

The functions above are extremely useful for quick checks of your data. But they quickly become tedious with larger datasets. For example, you have 200 years worth of fertility data. We need a better way to organize it and automate some processes. You don't want to compute a mean 200 times by hand!

The code below will make the dataset "long" instead of "wide" so that we have only three columns (country, year, children_per_woman), instead of 184 columns.

```
total_fertility_long <- total_fertility %>%
  gather(key = year, value = "children_per_woman", -country)
```

In words, this says "create a new file called total_fertility_long. In that file take the data from total_fertility *and then* gather all the columns and stack them, but ignore the country column in this operation". More generally, this procedure does this:

gather() is one function that the *tidyverse* packages does. *Tidyverse* provides a lot of useful shortcuts for arranging your data. For example, you could also filter the dataset so that it only contains data from 1800, 1900, and 2000:

```
total_fertility %>%
  gather(key = year, value = "children_per_woman", -country) %>%
  filter(year == 1800|year == 1900| year == 2000)
```

We will primarily use just two functions - *gather()* and *filter()*. But there are lots of extras to explore if you're interested here - https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf