Data Analytics

CS390

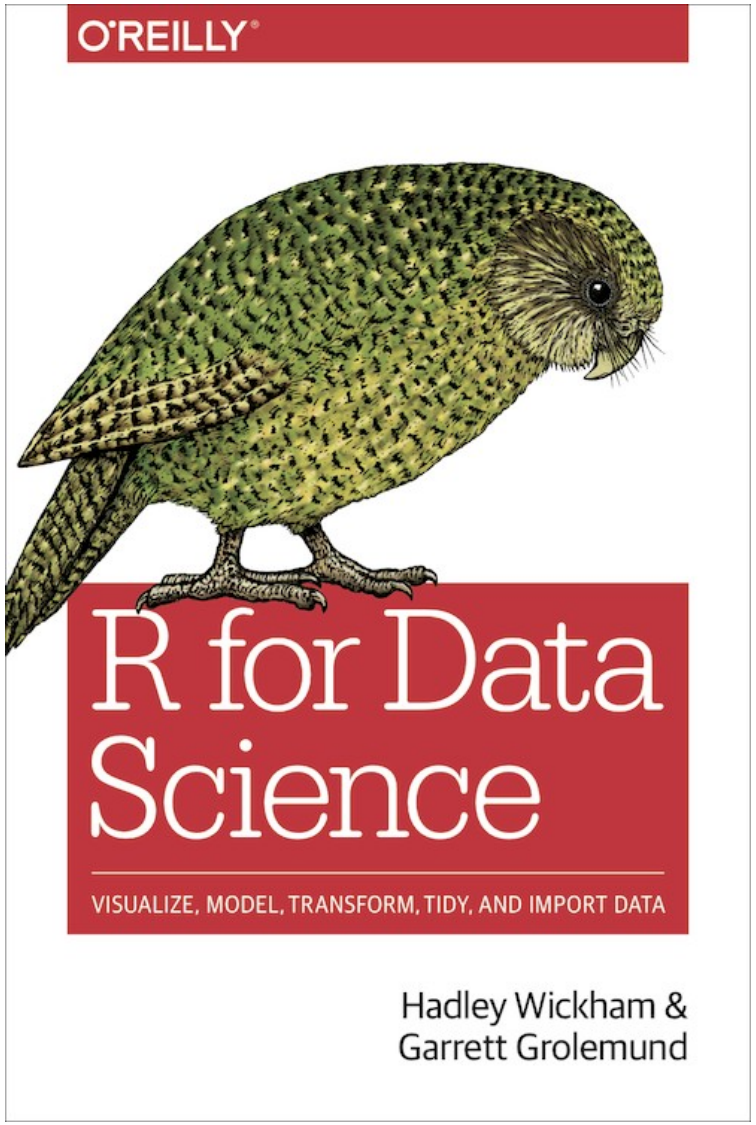
Tidy Data and Import

Week 6

Fall 2018

Oliver Bonham-Carter

Where in the Web?
Where in the Book?

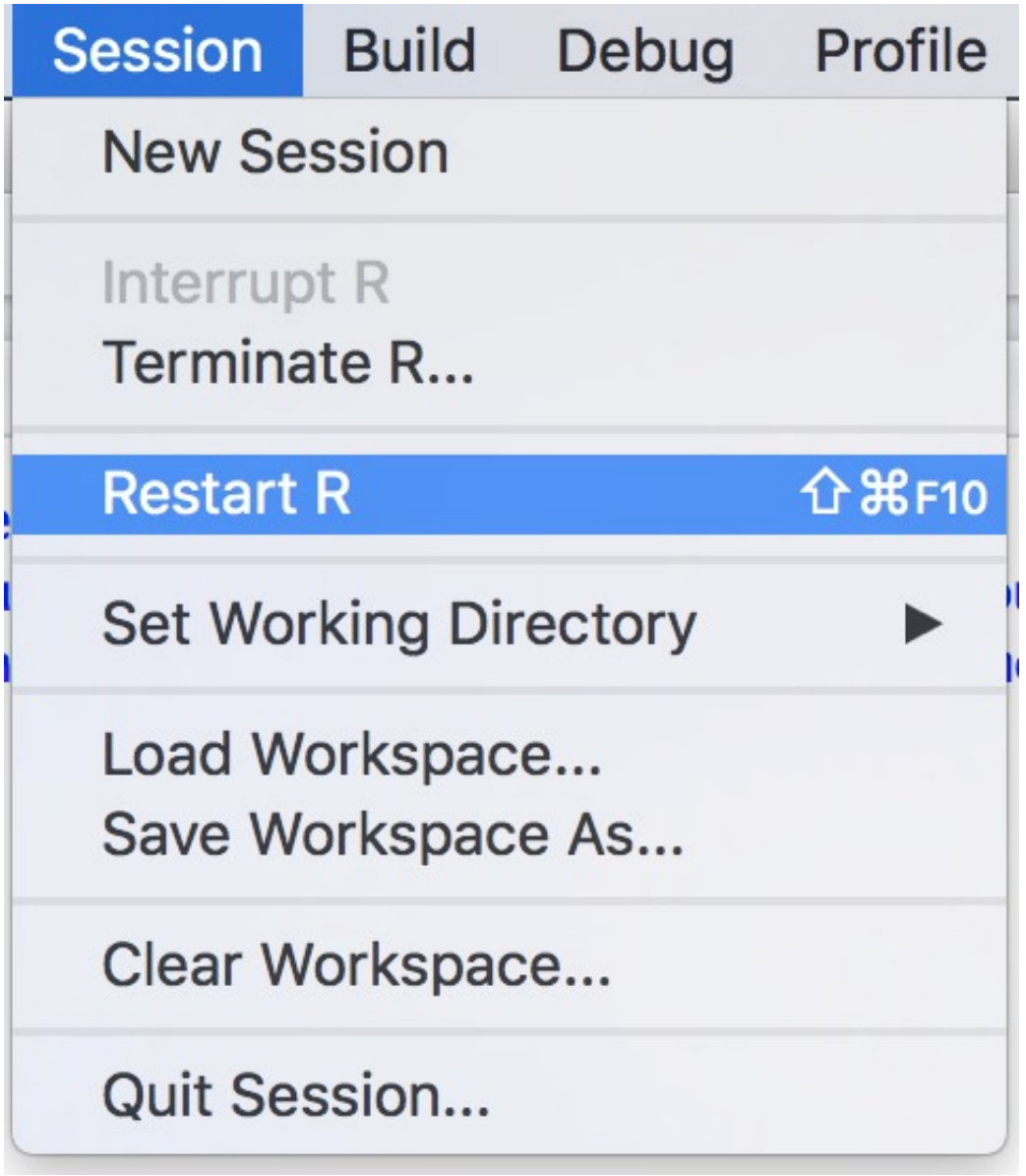


- Note the chapter differences!
- Book:
  - Chap 8
- Web:
  - Chap 11

- Tidy Data and Import

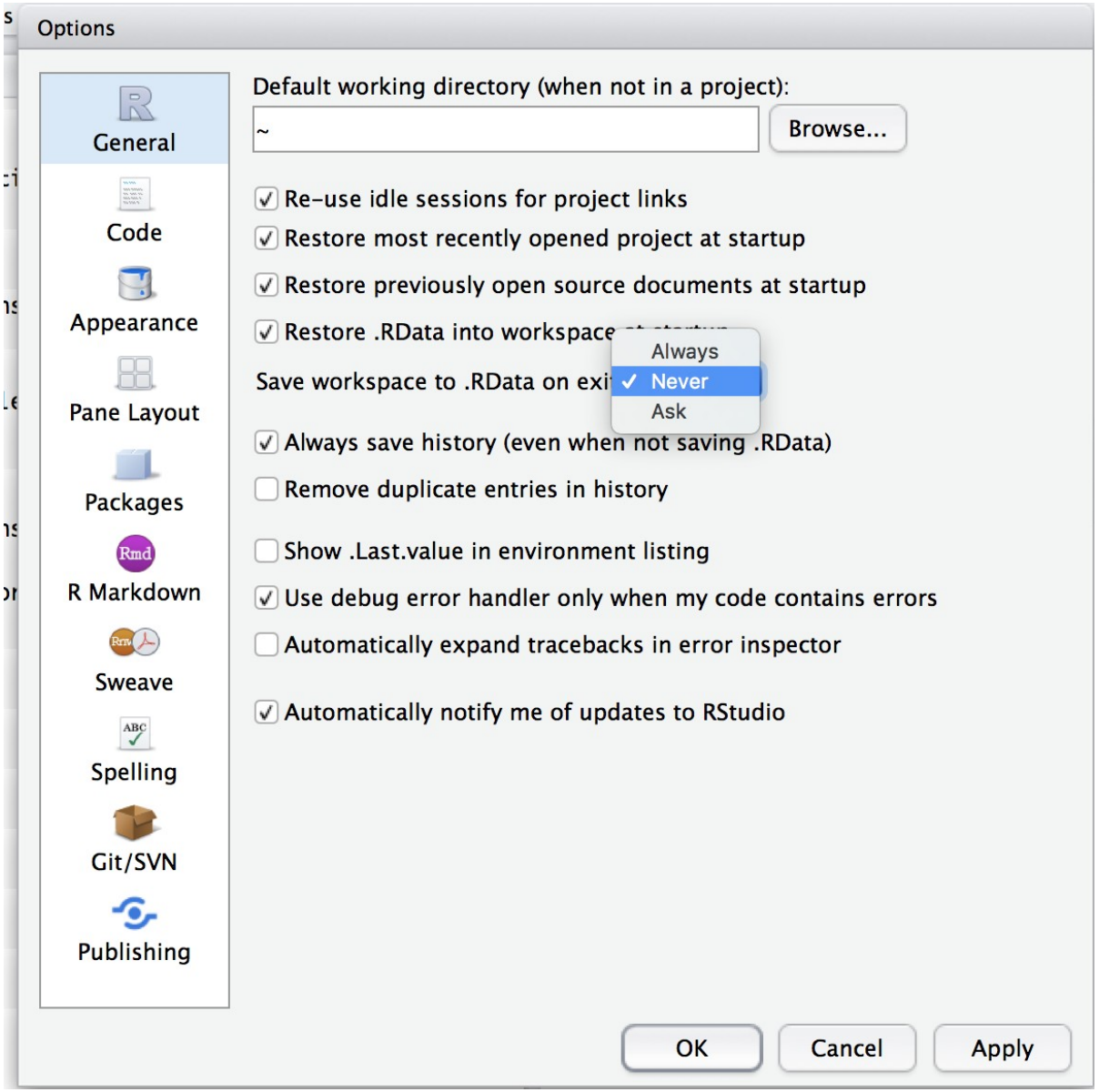
Now That We Are RStudio Programmers...

- Consider starting with a clean-slate, without a bunch of old data tables.
- Consider not saving your R-environment after each session.
- Instead, your work and code should come source files and not be text-mined from the command history.



Now That We Are RStudio Programmers...

- Consider stopping the workspace from being saved each time.
- This move will encourage you to begin writing code to be opened in RStudio.
- Better command archive for future works.



Entering Data as a Table

Your own data typed in:

```
read_csv("a,b,c
1,2,3
4,5,6")
```

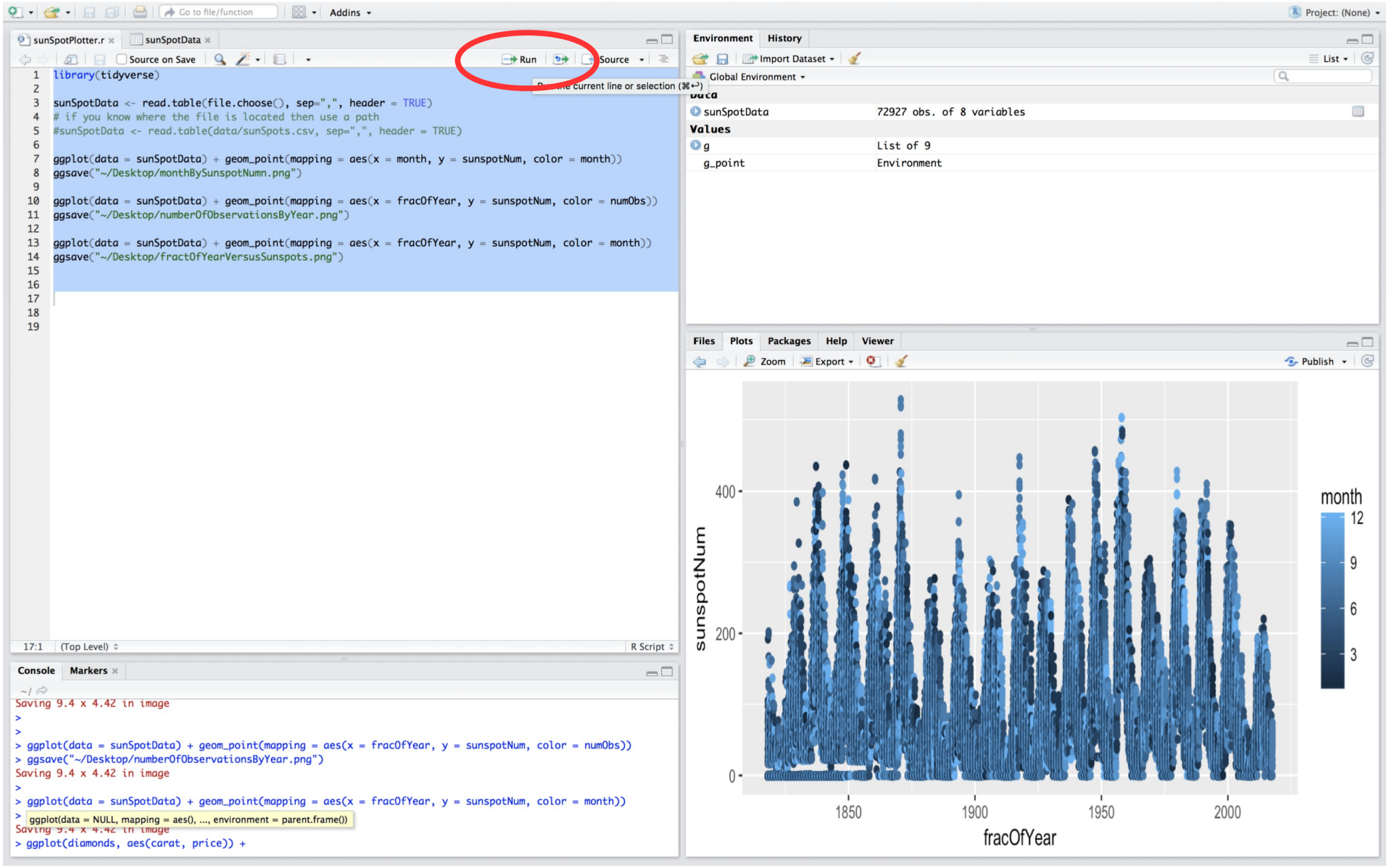
```
> read_csv("a,b,c
+ 1,2,3
+ 4,5,6")
# A tibble: 2 x 3
      a     b     c
  <int> <int> <int>
1     1     2     3
2     4     5     6
```

```
read_csv("a,b,c \n 1,2,3\n4,5,6")
read_csv("1,2,3\n4,5,6", col_names = FALSE)
read_csv("1,2,3\n4,5,6", col_names = c("a", "b", "c"))
```

Loading Data and Saving Plots

```
#Rscript:

library(tidyverse)

sunSpotData1 <- read.table(file.choose(), sep=",",
header = TRUE)

#sunSpotData2 <- read.table(data/sunSpots.csv,
sep=",", header = TRUE)

sunSpotData3 <- read_csv("PATH/sunSpots.csv")

ggplot(data = sunSpotData1) + geom_point(mapping =
aes(x = fracOfYear, y = sunspotNum, color = month))

#save the plot to file

ggsave("~/Desktop/fractOfYearVersusSunspots.png")
```

Save only good code and then have it to run later.



How Do We Deal With Messy Data?

- We may try to use a data table only to find:
  - There are numbers mixed with characters
  - Different types of entries are mixed in a column
  - Mixed makes things messy.

The Organization of Data

```
#Naturally tidy data:

data_frame(x = 1:5, y = 1, z = x ^ 2 + y)


library(tidyverse)

# The same data displayed in multiple ways; each data set
below organizes the values in a different way

table1 # country  year  cases population

table2 # country  year  type   count

table3 # country  year rate

table4a # country `1999` `2000`

table4b # country `1999` `2000`
```

What are the qualities that make data tidy?!

Tidy Data

- What does tidy data look like?
  - A column should be of all same types and description
- There are three interrelated rules which make a data set *tidy*:
  - Each variable must have its own column.
  - Each observation must have its own row.
  - Each value must have its own cell.

Tidy Data

- Be tidy: it matters how your data is arranged
- Trends could be *missed due to mess*
- Code is easiest to implement when data from a column is same

Figure 9-1 shows the rules visually.

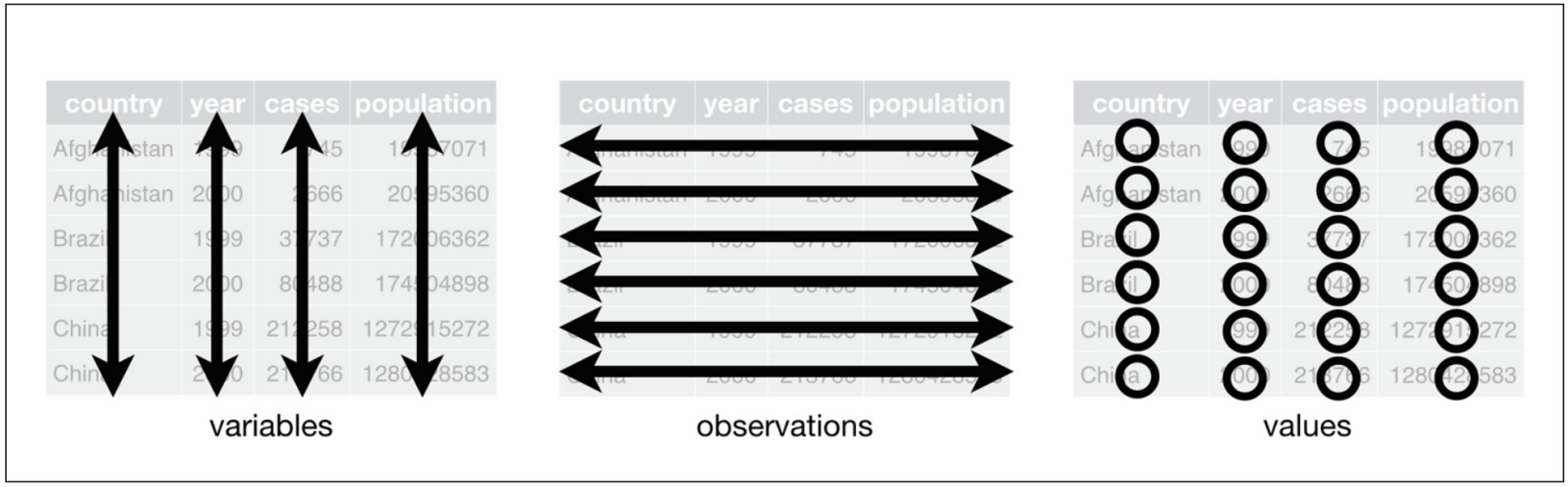


Figure 9-1. The following three rules make a dataset tidy: variables are in columns, observations are in rows, and values are in cells

Which Table is Most Tidy?

View(table1)

- There are three interrelated rules which make a data set tidy:
  - Each variable must have its own column.
  - Each observation must have its own row.
  - Each value must have its own cell.
- Table 1 is the most tidy for for data-organization

```
> table1
# A tibble: 6 x 4
      country  year  cases population
        <chr> <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3      Brazil  1999  37737  172006362
4      Brazil  2000  80488  174504898
5       China  1999 212258 1272915272
6       China  2000 213766 1280428583
```

All same types and descriptions in columns, but it seems that two sets are mixed

Not Tidy!!

- View(table2)
- Not tidy
- The Cases are easily confused

```
> table2
# A tibble: 12 x 4
       country  year       type      count
         <chr> <int>      <chr>      <int>
 1 Afghanistan  1999      cases        745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000      cases       2666
 4 Afghanistan  2000 population   20595360
 5      Brazil  1999      cases      37737
 6      Brazil  1999 population  172006362
 7      Brazil  2000      cases      80488
 8      Brazil  2000 population  174504898
 9       China  1999      cases     212258
10       China  1999 population 1272915272
11       China  2000      cases     213766
12       China  2000 population 1280428583
```

Use Tibble to find out how many different *years* there are mixed together

- Use the functions that we already know
  - #**Mutate()** to Compute rate per 10,000

```
table1 %>%  mutate(rate = cases / population * 10000)
```

  - #**Quick Computations** of cases per year

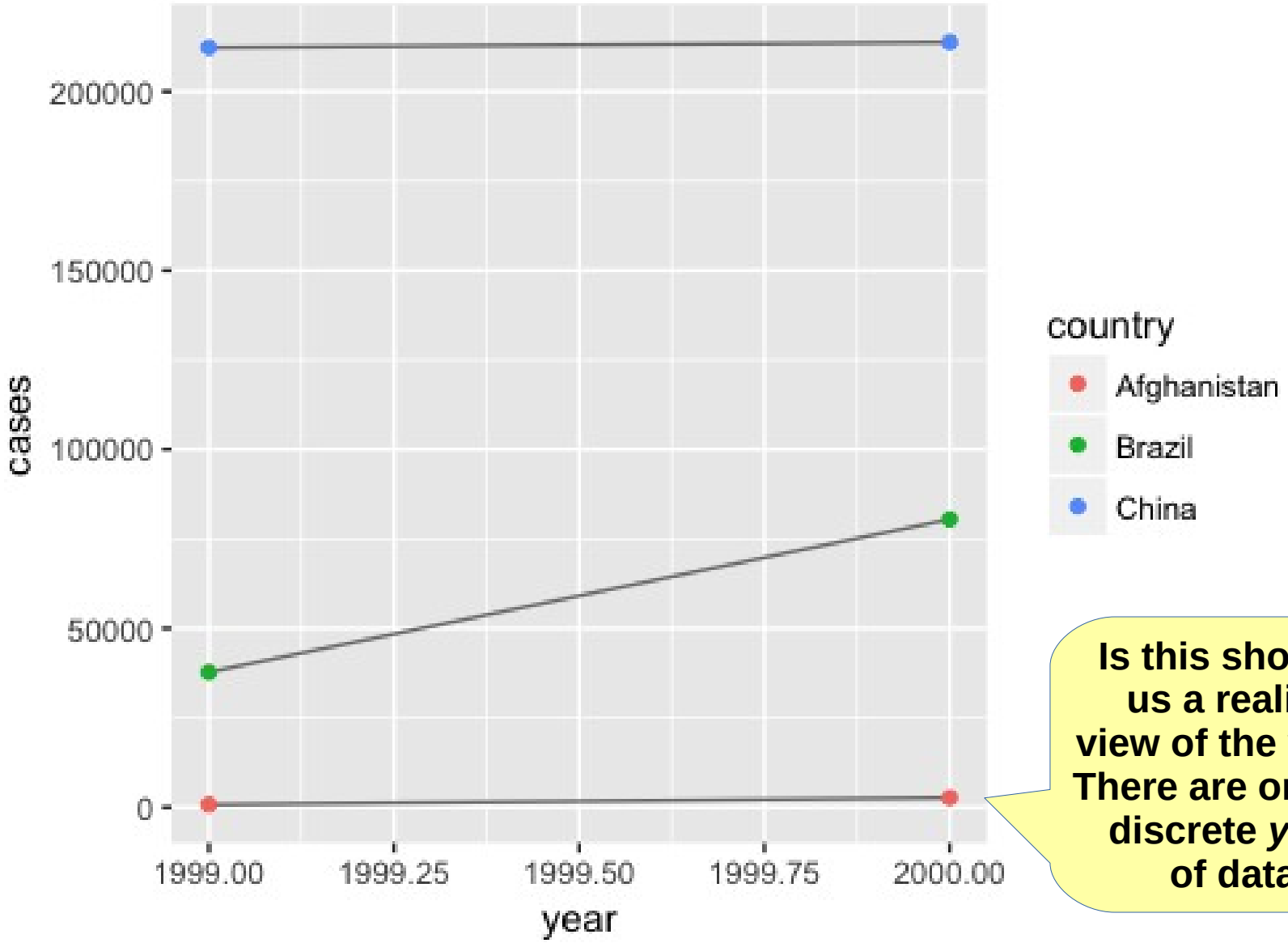
```
table1 %>% count(year, wt = cases)
# gives:
# A tibble: 2 x 2
# year      n
# <int>  <int>
# 1  1999 250740
# 2  2000 296920
```

1999: 745 + 37737 + 212258
2000: 213766 + 80488 + 2666

Implement Some Code

```
#Tidy tibble tables make using code convenient.

# Visualize changes over time on table1

library(ggplot2)

ggplot(table1, aes(year, cases)) + geom_line(aes(group = country),colour = "grey50") + geom_point(aes(colour = country))
```

Output From The Code



Is this showing us a realistic view of the years? There are only two discrete *years* of data.

Bad Organization, Bad Luck!!

- **We can apply code to data when in the right format (integers, strings, etc.)**
- **What happens when the data is badly stored; messy, and without any organization??!**

Gather(): Table4a

- *Gather()* takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed.
- Use *gather()* when you notice that you have columns that are not variables.

These variables could be better ordered as elements of “Year”

```
> table4a
# A tibble: 3 x 3
      country `1999` `2000`
*       <chr>  <int>  <int>
1 Afghanistan    745   2666
2      Brazil  37737  80488
3       China 212258 213766
```

Gather(): Table4a

Table4a is atrocious!

assemble elements and park them under "year" header

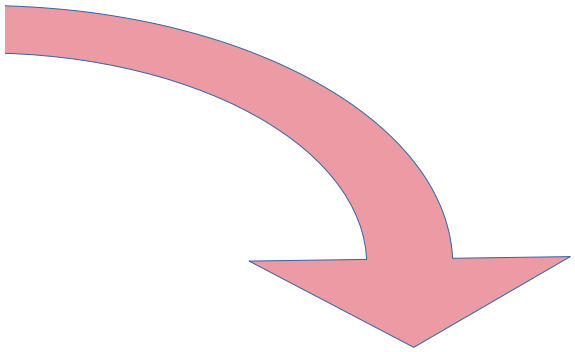
```
newTable <-
   table4a %>%
   gather(
     `1999`, `2000`,
      key = "year",
     value = "cases")
```

These variables could be better ordered as elements of "Year"

```
> table4a
# A tibble: 3 x 3
      country `1999` `2000`
*       <chr>  <int>  <int>
1 Afghanistan    745   2666
2      Brazil  37737  80488
3       China 212258 213766
```

Reordering of Data: table4a

	country	1999	2000
1	Afghanistan	745	2666
2	Brazil	37737	80488
3	China	212258	213766



```
newTable <-

  table4a %>%

  gather(
      `1999`,  `2000`,
       key = "year",
        value =  "cases")
```

```
> table4a %>% gather(`1999`, `2000`,
key = `year`, value = `cases`)
# A tibble: 6 x 3
      country  year  cases
        <chr> <chr>  <int>
1 Afghanistan  1999    745
2      Brazil  1999  37737
3       China  1999 212258
4 Afghanistan  2000   2666
5      Brazil  2000  80488
6       China  2000 213766
```

How did we do that?

```
newTable <-

   table4a %>%

   gather(
    `1999`,  `2000`,
    key = "year",
    value =
         "cases")
```

Here's how:
Reorganize the data
in the columns

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

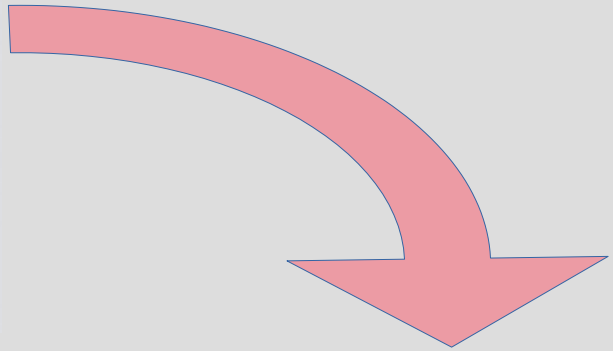
country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Figure 12.2: Gathering `table4` into a tidy form.

Reordering of Data: table4b

	country	1999	2000
1	Afghanistan	19987071	20595360
2	Brazil	172006362	174504898
3	China	1272915272	1280428583



```
newTable <-

table4b  %>%

gather( `1999`,  `2000`,
  key = "year",
  value = "population")
```

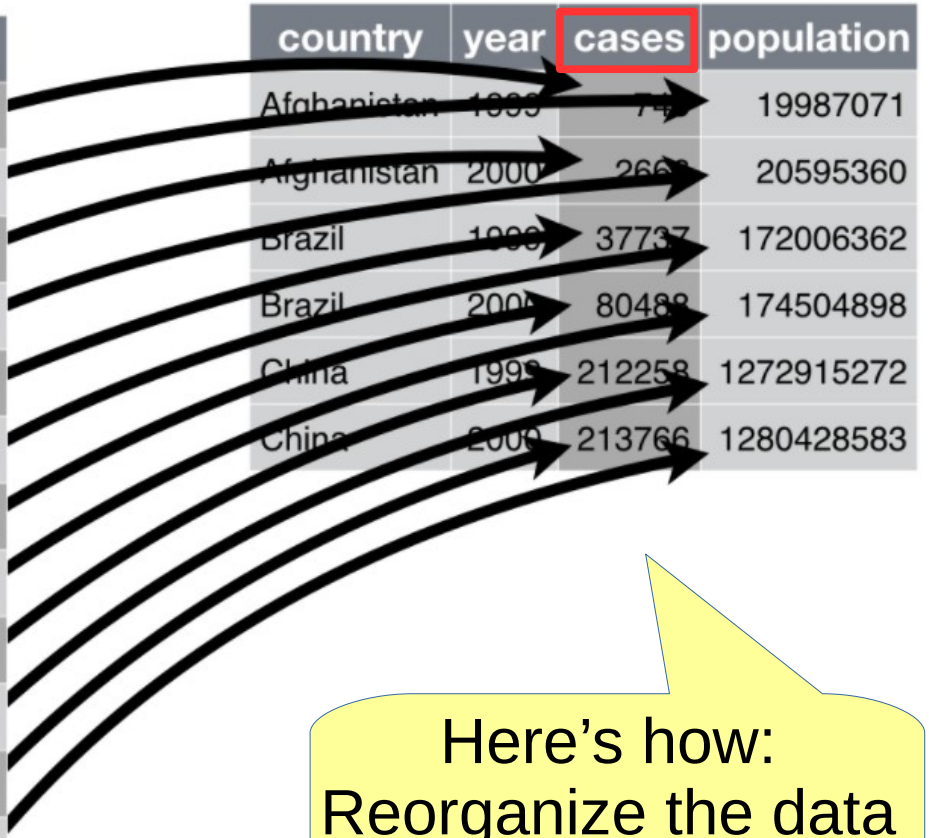
```
> table4b %>% gather(`1999`, `2000`,
key = `year`, value = `population`)
# A tibble: 6 x 3
      country  year population
        <chr> <chr>      <int>
1 Afghanistan  1999   19987071
2      Brazil  1999  172006362
3       China  1999 1272915272
4 Afghanistan  2000   20595360
5      Brazil  2000  174504898
6       China  2000 1280428583
```

Spreading(): table2

- Dealing with mixed values in the same column

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2



Gather(): table2

	country	year	type	count
1	Afghanistan	1999	cases	745
2	Afghanistan	1999	population	19987071
3	Afghanistan	2000	cases	2666
4	Afghanistan	2000	population	20595360
5	Brazil	1999	cases	37737
6	Brazil	1999	population	172006362
7	Brazil	2000	cases	80488

ng 1 to 8 of 12 entries

spread(table2,
key = type,
value = count)

```
> spread(table2, key = type,
value = count)
# A tibble: 6 x 4
      country  year  cases
*       <chr> <int>  <int>
1 Afghanistan  1999    745
2 Afghanistan  2000   2666
3      Brazil  1999  37737
4      Brazil  2000  80488
5       China  1999 212258
6       China  2000 213766
# ... with 1 more variables:
#   population <int>
```