Cleaning data

Dien Giau (Richard) Bui

2/7/2022

Load library

```r
library(tidyverse) # for MacOS: library(dplyr)
library(stringr)
library(janitor)
library(tidyr)
library(skimr)
library(readxl)
```

What is cleaning data?

- Data is generally “dirty”: it has a lot of errors/problems
    - Weird column names
    - Not in tidy format
    - Wrong data type: a numeric variable but is stored as characters
    - Some typos in data: “femela” (should be “female”)
    - Duplicates: some rows are repeated
    - Missing data: some rows/columns have no data
- “Garbage in, garbage out”
- So, cleaning step makes sure we have a clean data before doing any statistical analysis
- That’s why we start this course by a lecture on cleaning data

Our data in today lecture

- The US tuition fee in different states and territories
- A picture on the data: click here
- I borrowed the data from: Tidy Tuesday project
- I downloaded data and made it messy/dirty and store it in our "data/raw/us_avg_tuition.xlsx"
- Let's clean this data
  - I noted step by step in this slide

1. Load/import data to R

- Before load/import data, let's open it by MS Excel
- Load data to R by two ways:
  - By RStudio using mouse click
  - By code

Import data

```r
Tuition = read_excel("data/raw/us_avg_tuition.xlsx")
# View(Tuition)
```

Discussion [1]

- By hover the mouse over columns, we find that the data type is imported incorrectly
  - It is numeric data, but now is imported as character
- Columns names: quite non-standard

```
names(Tuition)
```

```
##  [1] "State"   "2004-05" "2005-06" "2006-07" "2007-08" '
##  [8] "2010-11" "2011-12" "2012-13" "2013-14" "2014-15" '
```

- For example, can you try to take one column for the year “2004-2005”:

```
Tuition$2004-05
```

Discussion [2]

- One row is duplicate/repeated: can you help me to find which row?
- One row has no data (missing data)
- Some state names are in abbreviation, not with full name: can you find them?
- **Conclusion:** there are a lot of problems with data, let's clean it in next slides

2. Clean column names

```
Tuition = clean_names(Tuition)
head(Tuition[,1:3])
```

```
## # A tibble: 6 x 3
##   state    x2004_05          x2005_06
##   <chr>    <chr>             <chr>
## 1 Alabama  5682.8381203801473 5840.5497850562942
## 2 Alaska   4328.2813621964096 4632.6234493346974
## 3 Arizona  5138.4953115100307 5415.5160491299894
## 4 Arkansas 5772.3018690601893 6082.3793244626404
## 5 CA       5285.9214889123541 5527.8812896622303
## 6 Colorado 4703.7770960929247 5406.9665199590581
```

3. Data type casting

- Character is not the same as numeric, for example

```
sqrt("16") # we can't apply numeric method to a character
sqrt(16)
```

- We want to convert tuition from character type to numeric
- In R, we will use `as.numeric` function:

```
as.numeric("16")
```

```
## [1] 16
```

Convert character to numeric columns

It is a bit long to type all of the below code, but we will return on how to make it shorter in a later lecture.

```r
Tuition = Tuition %>%
  mutate(
    x2004_05 = as.numeric(x2004_05),
    x2005_06 = as.numeric(x2005_06),
    x2006_07 = as.numeric(x2006_07),
    x2007_08 = as.numeric(x2007_08),
    x2008_09 = as.numeric(x2008_09),
    x2009_10 = as.numeric(x2009_10),
    x2010_11 = as.numeric(x2010_11),
    x2011_12 = as.numeric(x2011_12),
    x2012_13 = as.numeric(x2012_13),
    x2013_14 = as.numeric(x2013_14),
    x2014_15 = as.numeric(x2014_15),
    x2015_16 = as.numeric(x2015_16)
  )
```

Note on `pipe` or `%>%`

- Traditionally, we will write function like:

```
mutate(Tuition, ...)
```

- But in the previous example, I wrote:

```
Tuition %>% mutate(...)
```

- In plain English:
  1. We take `Tuition` data
  2. Then `mutate` to add/update new columns to the data
  3. and so on, we can continue forever with pipe
- It makes our code easier to read and follow, so I will use it a lot in this class

4. Remove duplicates

- Check duplicate by `duplicated`

```
duplicated(Tuition)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
## [13] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FAI
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
## [49] FALSE FALSE FALSE FALSE
```

- Remove duplicates by `distinct`

```
Tuition = Tuition %>%
  distinct()
```

5. Remove missing data

- Missing data is everywhere
- We have several ways to deal with missing:
  - Find out data
  - Fill by a make-sense number: fill by zero, by industry average, ...
  - Or remove missing rows
- We will remove missing for convenience in our data
  - Can you guess how we can remove the missing row?

Remove missing by filter out

```
Tuition = Tuition %>%
  filter(state != "Puerto Rico")
```

6. String manipulation in `state` columns

- Recall that:
  - Some state names are in abbreviation, not with full name: “CA”, “NY”
- We use package “`stringr`” and its function “`str_replace`”

```
Tuition = Tuition %>%
  mutate(
    state = str_replace(state, "CA", "California")
  ) %>%
  mutate(
    state = str_replace(state, "NY", "New York")
  )
```

7. Wide to long data

- We can transform data from wide to long as following:

```
Tuition = Tuition %>%
  pivot_longer(-state,
               names_to = "year",
               values_to = "tuition")
head(Tuition)
```

```
## # A tibble: 6 x 3
##   state   year     tuition
##   <chr>   <chr>      <dbl>
## 1 Alabama x2004_05   5683.
## 2 Alabama x2005_06   5841.
## 3 Alabama x2006_07   5753.
## 4 Alabama x2007_08   6008.
## 5 Alabama x2008_09   6475.
## 6 Alabama x2009_10   7189.
```

Quiz

- Can you transform the `year` column to be numeric from 2004 to 2015?
- For example: `x2004_05` will become 2004, `x2005_06` will become 2005

Transform year

- Will solve in class

```
# add code here
```

Look how clean of the final data

```
summary(Tuition)
```

```
##      state                year              tuition
##  Length:600         Length:600         Min.   : 3621
##  Class :character   Class :character   1st Qu.: 6108
##  Mode  :character   Mode  :character   Median : 7607
##                                        Mean   : 7899
##                                        3rd Qu.: 9424
##                                        Max.   :15224
```

Conclusion

- So many steps to clean a data
- Need to document all process in an Rmd file like this
  - or you can copy each step's code + results to MS `docx` file, but will need more efforts
- By the end, you can run everything together to get a report: pdf, word, html format
- Final tip: save this cleaned data to a `data/process` folder

Save data

```
saveRDS(Tuition, "data/process/Tuition_clean.rds")
```