# Slide 02: Dataframe (or Table)

Richard Dien Giau Bui

# Load packages

- Again, first load required packages:

```r
library(tidyverse)
library(skimr)
library(readxl)
library(janitor)
```

# Dataframe

- In R, we often put data in a dataframe (or data.frame)
- In short, it is a table, like Excel table
- So, we will learn how to work with a dataframe in this slide

# How to make a dataframe in R?

1. Put columns/variables together
2. Import

# 1. Put variables together

```r
id = c(1:5)
score = c(100, 80, 97, 76, 50)

Data = data.frame(id=id, score=score)
Data
```

```
  id score
1  1   100
2  2    80
3  3    97
4  4    76
5  5    50
```

# 2. Import data

- First, check the working directory

```r
getwd()
```

```
[1] "B:/_YZU/01_teaching/cm002/prepare_slides"
```

- You can change to correct folder by:

```r
setwd()
```

# 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

# Some questions

- How many columns & rows in data?
- What columns we have in the data?
- Show the head of data (first few rows)
- Take the first column of data

# How many columns & rows in data?

```
nrow(Tuition)
```

```
[1] 52
```

```
ncol(Tuition)
```

```
[1] 13
```

```
dim(Tuition)
```

```
[1] 52 13
```

# What columns we have in the data?

```r
names(Tuition)
```

```
[1] "State"   "2004-05" "2005-06" "2006-07" "2007-08" "2008-09" "2009-10"
[8] "2010-11" "2011-12" "2012-13" "2013-14" "2014-15" "2015-16"
```

# Head of data

```r
head(Tuition)
```

```
# A tibble: 6 × 13
  State    `2004-05` `2005-06` `2006-07` `2007-08` `2008-09` `2009-10` `2010-
11`
  <chr>    <chr>     <chr>     <chr>     <chr>     <chr>     <chr>     <chr>
1 Alabama  5682.838… 5840.549… 5753.496… 6008.168… 6475.091… 7188.954…
8071.133…
2 Alaska   4328.281… 4632.623… 4918.500… 5069.822… 5075.482… 5454.606…
5759.152…
3 Arizona  5138.495… 5415.516… 5481.419… 5681.637… 6058.463… 7263.204…
8839.604…
4 Arkansas 5772.301… 6082.379… 6231.977… 6414.900… 6416.503… 6627.092…
6900.912…
5 CA       5285.921… 5527.881… 5334.825… 5672.472… 5897.888… 7258.771…
8193.738…
6 Colorado 4703.777… 5406.966… 5596.348… 6227.001… 6284.137… 6948.472…
7748.200
```

# Take one column of data

- The function will be: `Data$Column`
- Remember to use **TAB** key after the `$` to easy your life

```
Tuition$State
```

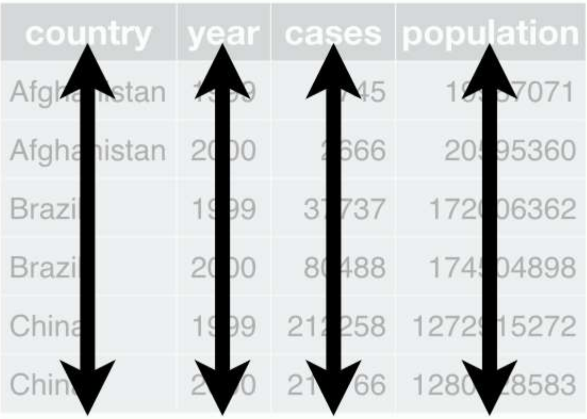
```
 [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"      
 [5] "CA"             "Colorado"       "Connecticut"    "Delaware"      
 [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"         
[13] "Illinois"       "Indiana"        "Iowa"           "Kansas"        
[17] "Kentucky"       "Louisiana"      "Louisiana"      "Maine"         
[21] "Maryland"       "Massachusetts"  "Michigan"       "Minnesota"     
[25] "Mississippi"    "Missouri"       "Montana"        "Nebraska"      
[29] "Nevada"         "New Hampshire"  "New Jersey"     "New Mexico"    
[33] "NY"             "North Carolina" "North Dakota"   "Ohio"          
[37] "Oklahoma"       "Oregon"         "Pennsylvania"   "Puerto Rico"   
[41] "Rhode Island"   "South Carolina" "South Dakota"   "Tennessee"     
[45] "Texas"          "Utah"           "Vermont"        "Virginia"      
[49] "Washington"     "West Virginia"  "Wisconsin"      "Wyoming"       
```

# Tidy dataframe

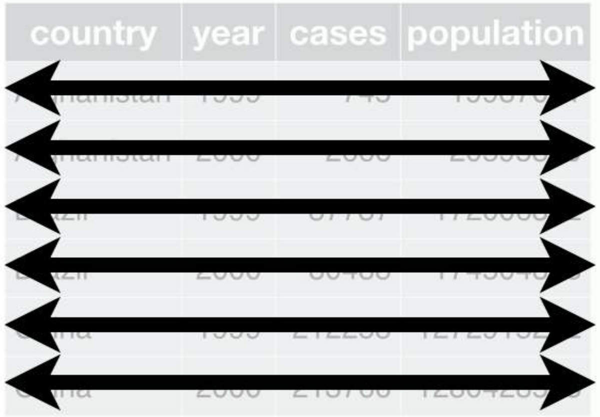
- All dataframes are often dirty, but some are clean
  - Like all ladies are beautiful, but some are more beautiful
- > “Happy families are all alike; every unhappy family is unhappy in its own way.” –– Leo Tolstoy
- > “Tidy datasets are all alike, but every messy dataset is messy in its own way.” –– Hadley Wickham

# How (all) tidy data look like?

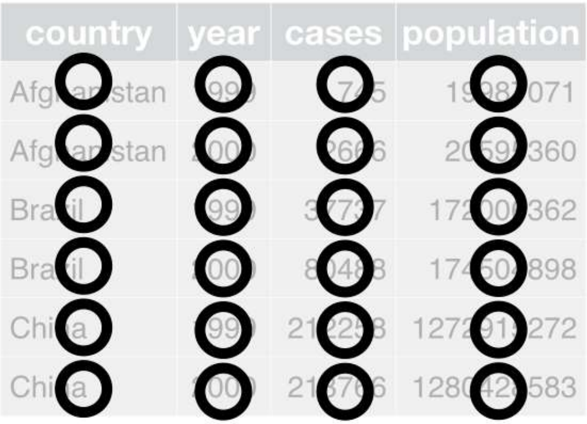
- First row is columns/variable names
- Each row is one observation
- Each cell is a value



variables



observations



values

# Examples of (specific) dirty data

- Messy variable names
- Wrong data type: a numeric variable but is stored as characters
- Not in tidy format
- Some typos in data: `"femela"` (should be `"female"`)
- Duplicates: some rows are repeated
- Missing data: some rows/columns have no data

# Why not just dirty 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

# Your turn

- Can you open the data and help me to find which part of data looks dirty?

# 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" "2008-09" "2009-10"
[8] "2010-11" "2011-12" "2012-13" "2013-14" "2014-15" "2015-16"
```

- 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

# Clean 01. Clean column names

- You can rename one by one by `rename` function:

```
rename(Tuition, new_name = old_name)
```

- Or do automatically by `clean_names`

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

```
# A tibble: 6 × 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
```

# Clean 02. Data type casting

- Character is not the same as numeric, for example

```r
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:

```r
as.numeric("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.

# 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

# Clean 03. Remove duplicates

- Check duplicate by `duplicated`

```r
duplicated(Tuition)
```

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

- Remove duplicates by `distinct`

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

# Clean 04. 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

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

# Clean 05. 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`”

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

# Clean 06. Wide to long data

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

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

```
# A tibble: 6 × 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.
```

# Your turn

- 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

```
skim(Tuition)
```

Data summary

| Name | Tuition |
|---|---|
| Number of rows | 600 |
| Number of columns | 3 |
| ________________________ | |
| Column type frequency: | |
| character | 2 |
| numeric | 1 |

| | |
|---|---|
| ________________________________ | |
| Group variables | None |

### Variable type: character

| skim_variable | n_missing | complete_rate | min | max | e |
|---|---|---|---|---|---|
| state | 0 | 1 | 4 | 14 | |
| year | 0 | 1 | 8 | 8 | |

### Variable type: numeric

| skim_variable | n_missing | complete_rate | mean | |
|---|---|---|---|---|
| tuition | 0 | 1 | 7899.26 | 2404 |

# 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

- R provides an `rds` data format to save processed dataframe
- Today, we save it to `data/process` folder

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

# Any questions

- Any questions?

# If no, let summarize some functions we learn today

| Function | Use when: |
|---|---|
| `nrow`, `ncol`, `dim`, `names`, `head` , `View` | take a look at data |
| `Data$Column` | take a column from a data |
| `mutate` | make a new column or modify the existing column |
| `filter` | filter or subset rows based on a condition |

| Function | Use when: |
|---|---|
| `distinct` | remove duplications in data based on selected columns |
| `as.numeric` | convert a column to numericc |
| `clean_names` | to make variable names become clean |
| `rename` | rename manually: new = old |
| `str_replace` | to replace string by a pattern |
| `pivot_longer` | transform data from wide format to long format |

# Next lecture

- Will be a new data
- And you practice to clean by yourself