

Probability - Tidyverse 1

Xiaolong Yang

University of Tokyo

June 24, 2022

Today's Game Plan

- 1 column-wise operations with `dplyr` package
 - `across()`
 - `where()`
- 2 writing mathematics in `R`markdown
- 3 writing code in `R`markdown

i Today's in-class assignment: `enigma`

Section 1

Column-wise operations

Recap: dplyr as a grammar of data manipulation

- `mutate()` adds new variables that are functions of existing variables
- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarize()` reduces multiple values down to a single summary.
- `arrange()` changes the ordering of the rows.
- `group_by()` perform operations by group

Column-wise operations with dplyr

Apply the same function to multiple variables/columns?

- tedious to apply the same operation across columns
- solution: `across()` + `where()` + other dplyr functions

Column-wise operations with dplyr

- FLVoters data set as a heuristic

```
glimpse(FLVoters)
```

```
Rows: 10,000
```

```
Columns: 6
```

```
$ surname <chr> "PIEDRA", "LYNCH", "CHESTER", "LATHROP", "HUMM  
$ county  <int> 115, 115, 115, 115, 115, 115, 115, 115, 1, 1,  
$ VTD     <int> 66, 13, 103, 80, 8, 55, 84, 48, 41, 39, 26, 45  
$ age     <int> 58, 51, 63, 54, 77, 49, 77, 34, 56, 60, 44, 45  
$ gender  <chr> "f", "m", "m", "m", "f", "m", "f", "f", "f", "  
$ race    <chr> "white", "white", NA, "white", "white", "white"
```

Tedious-and-error-prone-but-works approach

- calculate the mean to all the numeric variables

```
FLVoters %>% summarise(  
  mean_country = mean(county, na.rm = TRUE),  
  mean_VTD = mean(VTD, na.rm = TRUE),  
  mean_age = mean(age, na.rm = TRUE)  
)
```

```
  mean_country mean_VTD mean_age  
1      70.6237 232.0697 52.60979
```

Easier-and-works-better approach

- same goal, different operation

```
FLVoters %>% summarise(across(county:age, mean, na.rm = TRUE))
```

	county	VTD	age
1	70.6237	232.0697	52.60979

Basic usage of across() function

- .cols selects the columns to operate on
 - think of select()
 - by position, name, type
- .fns function(s) to apply to each column
 - n_distinct()
 - min()
 - max()
 - sum()
 - sd()
 - ...

```
across(.cols = columns,  
      .fns = function)
```

A list of functions for .fns

- name a list of functions
- supply the named list in the fns argument
- example: min() and max() at once

```
min_max <- list(  
  min = ~min(.x, na.rm = TRUE),  
  max = ~max(.x, na.rm = TRUE)  
)
```

```
FLVoters %>%  
  summarise(across(county:age, min_max)) %>%  
  glimpse()
```

A list of functions for .fns

Rows: 16

Columns: 8

Groups: gender [3]

```
$ gender      <chr> "f", "f", "f", "f", "f", "f", "f", "m", "m"
$ race        <chr> "asian", "black", "hispanic", "native", "ot
$ county_min  <int> 1, 1, 1, 9, 1, 1, 1, 1, 1, 1, 9, 1, 1, 1, 1
$ county_max  <int> 129, 131, 127, 101, 131, 133, 131, 127, 129
$ VTD_min     <int> 2, 1, 1, 4, 3, 1, 1, 3, 1, 1, 4, 1, 1, 1, 3
$ VTD_max     <int> 1294, 1404, 1432, 1062, 1391, 1433, 1251, 1
$ age_min     <int> 20, 19, 19, 19, 19, 19, 19, 19, 19, 19, 25
$ age_max     <int> 80, 99, 95, 91, 92, 99, 99, 82, 92, 96, 90
```

across() with where()

- where() as a selection helper
 - ① takes a function
 - ② returns all variables when function = TRUE

```
FLVoters %>%  
  select(where(is.numeric)) %>%  
  head()
```

	county	VTD	age
1	115	66	58
2	115	13	51
3	115	103	63
4	115	80	54
5	115	8	77
6	115	55	49

across() with where()

- where() as a selection helper
 - ① takes a function
 - ② returns all variables when function = TRUE

```
FLVoters %>%  
  summarize(across(where(is.numeric), mean, na.rm = TRUE)) %>%  
  glimpse()
```

Rows: 1

Columns: 3

```
$ county <dbl> 70.6237  
$ VTD    <dbl> 232.0697  
$ age    <dbl> 52.60979
```

across() in conjunction with dplyr verbs

- `across()` work with most other verbs (besides `summarize()`) e.g.,
 - `mutate()`
 - `group_by()`
 - `count()`
 - `distinct()`
 - ...

```
FLVoters %>%  
  na.omit() %>%  
  group_by(gender) %>%  
  summarise(  
    across(where(is.numeric), min_max),  
    across(where(is.numeric), mean),  
    across(where(is.character), tolower)) %>%  
  glimpse()
```

across() in conjunction with dplyr verbs

Rows: 9,113

Columns: 12

Groups: gender [2]

```
$ gender      <chr> "f", "f", "f", "f", "f", "f", "f", "f", "f"
$ county_min  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
$ county_max  <dbl> 133, 133, 133, 133, 133, 133, 133, 133, 133, 133, 133, 133, 133, 133
$ VTD_min     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
$ VTD_max     <dbl> 1433, 1433, 1433, 1433, 1433, 1433, 1433, 1433, 1433, 1433, 1433, 1433, 1433, 1433
$ age_min     <dbl> 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19
$ age_max     <dbl> 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99
$ county      <dbl> 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437, 70.2437
$ VTD         <dbl> 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661, 241.8661
$ age         <dbl> 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159, 52.81159
$ surname     <chr> "piedra", "hummel", "homan", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer", "heschmeyer"
$ race        <chr> "white", "white", "white", "white", "white", "white", "white", "white", "white", "white", "white", "white", "white", "white"
```

Short Summary: why `across()`?

- **flexible**: complex column-wise operations
 - works great with `summarise()` with the help of `where()`
- **light**: reduces repetition of functions

Section 2

Writing Maths in Rmarkdown

Mathematical modes in Rmarkdown

- 1 *Inline* by $\$ \dots \$$
 - to write maths as part of a paragraph
- 2 *Display* by $\$ \$ \dots \$ \$$
 - independent expressions that are put on separate lines

LaTeX for typesetting mathematics

- widely adopted approach to write technical documents in political science and across scientific disciplines
- Cheat sheet for mathematical notations: **link**

"Focus on writing, not typesetting."

Example

Inline maths

Define $P(M_i)$ as the probability that a randomly chosen message was assigned to machine i . Let U denote the event that some machine failed to decode a message. Then, we are interested in determining machine i for which $P(M_i \mid U)$ is the greatest. Applying Bayes' rule, we have,

Display maths

$$P(M_i \mid U) = \frac{P(U \mid M_i) P(M_i)}{P(U)} = \frac{P(U \mid M_i) P(M_i)}{\sum_{j=1}^5 P(U \mid M_j) P(M_j)}$$

Example

Inline maths

Define $P(M_i)$ as the probability that a randomly chosen message was assigned to machine i . Let U denote the event that some machine failed to decode a message. Then, we are interested in determining machine i for which $P(M_i | U)$ is the greatest. Applying Bayes' rule, we have,

Display maths

$$P(M_i | U) = \frac{P(U | M_i)P(M_i)}{P(U)} = \frac{P(U | M_i)P(M_i)}{\sum_{j=1}^5 P(U | M_j)P(M_j)}$$

Section 3

Writing code in Rmarkdown

Two modes of code in Rmarkdown

- An Rmarkdown document = **prose** + **code**
- Code in an Rmarkdown document = **code chunks** + **inline code**

Display code : wrap code with ````${code}```` and `````

💡 Code chunk is customizable by setting the arguments in `{}`. See [link](#) for more.

Inline code: wrap code with ```

Please use ``library("tidyverse")`` to load the ``tidyverse`` package.

→ Please use `library("tidyverse")` to load the tidyverse package.

What we learnt

- column-wise operations with `across()` and `where()`
- writing maths (`inline & display`) in Rmarkdown
- writing codes (`inline & display`) in Rmarkdown

Future Game Plan

- write your functions
- `purrr` package

- Quantitative Social Science: An Introduction in tidyverse
- R for Data Science
- Advanced R