

# Uncertainty - Tidyverse 3

## For loop variations and wrap-up

Xiaolong Yang

Introduction to Quantitative Social Science

University of Tokyo

July 7, 2022

# Today's Game Plan

- **For loop:** variations
- Tidyverse recap

**i** Today's in-class assignment: nazi-election

# For loop variations

- ① Modifying an existing object, instead of creating a new object
- ② Looping over names or values, instead of indices

# Modifying an existing object

## For loop to modify an existing object

```
df$a <- (df$a - min(df$a, na.rm = TRUE)) /  
(max(df$a, na.rm = TRUE) - min(df$a, na.rm = TRUE))  
df$b <- (df$b - min(df$b, na.rm = TRUE)) /  
(max(df$b, na.rm = TRUE) - min(df$b, na.rm = TRUE))  
df$c <- (df$c - min(df$c, na.rm = TRUE)) /  
(max(df$c, na.rm = TRUE) - min(df$c, na.rm = TRUE))  
df$d <- (df$d - min(df$d, na.rm = TRUE)) /  
(max(df$d, na.rm = TRUE) - min(df$d, na.rm = TRUE))
```

# Modifying an existing object

## For loop to modify an existing object

```
rescale01 <- function(x) {  
  rng <- range(x, na.rm = TRUE)  
  (x - rng[1]) / (rng[2] - rng[1])  
}  
  
df$a <- rescale01(df$a)  
df$b <- rescale01(df$b)  
df$c <- rescale01(df$c)  
df$d <- rescale01(df$d)
```

# Modifying an existing object

```
for (i in seq_along(df)) {  
  df[[i]] <- rescale01(df[[i]])  
}
```

- ① The output: no need to create new vector - **input as output**
- ② The sequence: iterate over each column with `seq_along(df)`
- ③ The body: apply `rescale01()`

# Looping patterns

- ➊ loop over the numeric indices

- `for (i in seq_along(vec))`

```
for (i in seq_along(vec) {  
  name <- names(vec)[[i]]  
  value <- x[[i]]  
}
```

# Looping patterns

- ① loop over the numeric indices
  - `for (i in seq_along(vec))`
- ② Loop over the elements
  - `for (x in vec)`

# Looping patterns

- ➊ loop over the numeric indices
  - `for (i in seq_along(vec))`
- ➋ Loop over the elements
  - `for (x in vec)`
- ➌ Loop over the names
  - `for (nm in names(vec))`

# Tidyverse recap

R programming → data analysis

- baseR as the basic syntax
- tidyverse as a dialect or specific syntax
  - philosophy: “*facilitate a conversation between a human and a computer about data*”



Figure 1: Tidyverse packages

# Component packages of tidyverse

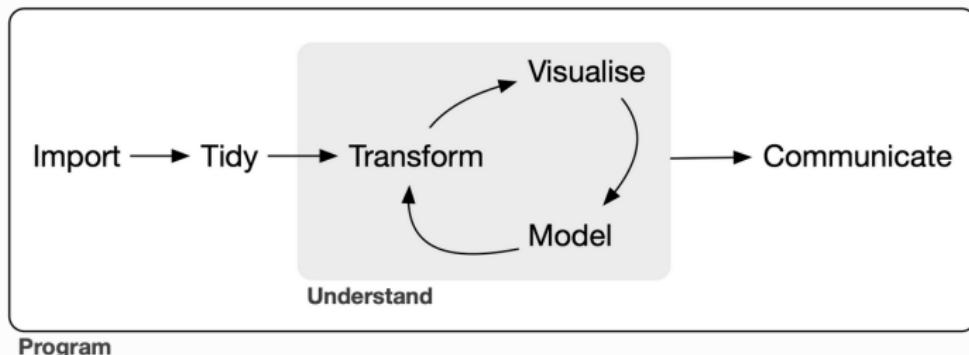


Figure 2: Data analysis workflow; R4DS

# Tidy: dplyr

- Every column is variable
- Every row is an observation
- Every cell is a single value

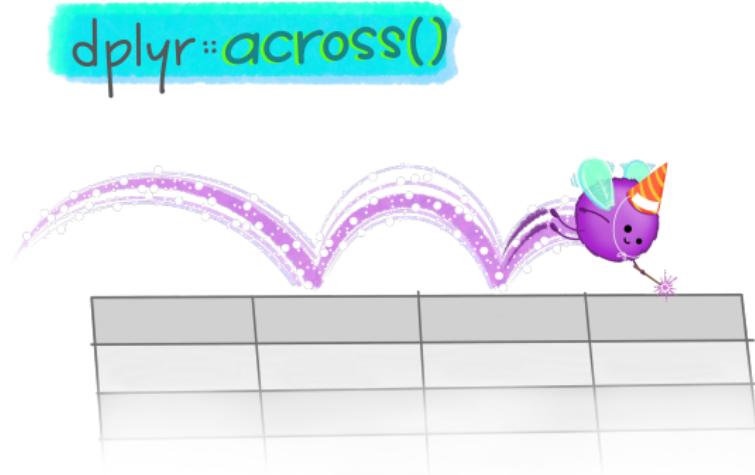


# Tidy: dplyr: mutate()



Figure 3: mutate()

## Tidy: dplyr: across()



@allison\_horst

Figure 4: `across()`

# Tidy: dplyr: case\_when()



Figure 5: case\_when()

# Tidy: dplyr: filter()



Figure 6: filter()

## Tidy: dplyr: group\_by()



Figure 7: group\_by()

## Transform: dplyr with stringr



Figure 8: stringr

# Transform: dplyr with lubridate



Figure 9: lubridate

# Visualize: ggplot2



Figure 10: ggplot2

# Model: modelr

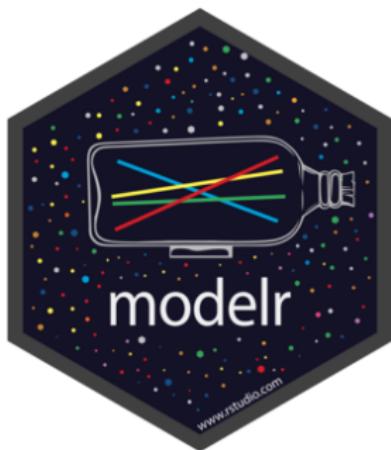


Figure 11: modelr

# Programming: purrr and magrittr



purrr



magrittr

# Communication: RMarkdown



# Other useful resource

## Debugging in R

- Advanced R: Debugging
- Debugging with the RStudio IDE

## Looping patterns: advanced

- Unknown output length
- Unknown sequence length

# Summary

## What we learnt

- for loop variations
- tidyverse packages and major functions covered in the **QSS Tidyverse** textbook

# Reference

- Quantitative Social Science: An Introduction in tidyverse
- R for Data Science
- Advanced R
- Artwork by @allison\_horst