

# Statistics with Recitation: TA Session

Danny Po-Hsien Kang (康柏賢)

September 16, 2025

# Today's agenda

## 1 R Guides & Tips

- Open a script
- Project
- Folder Organization
- Coding styles

## 2 Summary Statistics Table

- `table()`
- `addmargins()`

## 3 Remove everything in the environment

- `rm(list = ls())`

## 4 Draw Graphs

- `geom_bar()`
- `labs()`
- `scale_x_continuous()`, `scale_y_continuous()`

# Today's Dataset

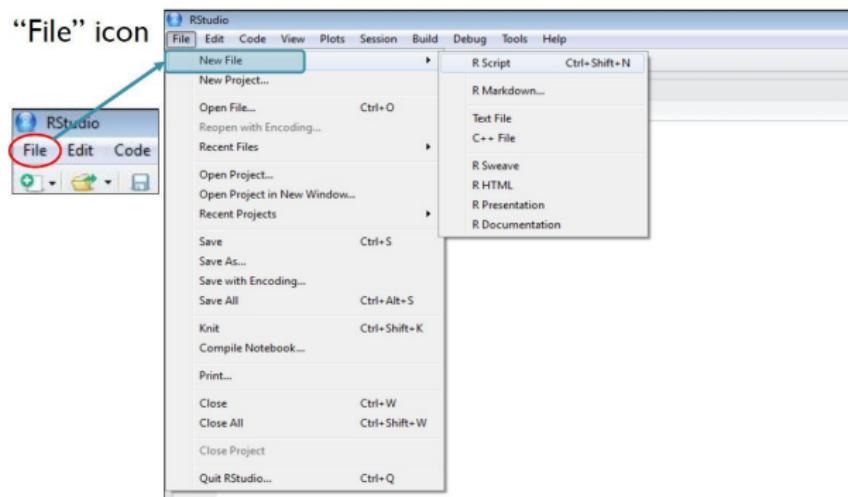
- Please download "loan\_full\_schema.csv" by the link on TA's website. This is a bigger version of "loan50.csv", which we used last week.
- Please import the following dataset into your RStudio.
  - loan\_full\_schema.csv
  - county.csv

## About the Quiz

- Quiz 1 will be held on 9/23 (Tue.).
- It will cover Chapter 1-2 (i.e. Main Course week 1-3).
- Quiz 1 are handwritten questions.

# Script in RStudio

- A **script** is a text file that you can write your code in.



# Why should we open a script?

- Main Reason: We open an R script so we can save it!
- Save your code in the script such that it is executable and reproducible.

The screenshot shows the RStudio interface with the following components:

- File menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Editor pane:** Displays an R script named "2025FAI\_Statistics\_TA\_session\_wk3.R". The script contains code for importing datasets, creating a contingency table, and generating a scatter plot.
- Environment pane:** Shows the global environment with objects like `bar`, `bar_graph`, `bar_graph_withlabel`, `county`, `loan`, `scatter_plot`, `scatter_plot_scale`, `table_withsum`, `values`, and `table`.
- Plots pane:** Displays a scatter plot of `homeshowership` versus `homeshowership`, showing a dense cloud of points.
- Console pane:** Shows the command history for running the script.

```
# 2025FAI_Statistics_TA_session_wk3.R
# clear the memory, import package for graphing
rm(list = ls())
library(ggplot2)

# 1. import the dataset
loan <- read.csv("data/loans_full_schema.csv")
county <- read.csv("data/county.csv")
# 2. Create contingency table object: 'table()'
# using 'table' function to easily build contingency table within dataset
table <- table(loan$application_type,
               loan$homeshowership)
print(table)
class(table)
# we can also specify the column name in the function arguments
# by giving a vector of names to parameter 'dim', which means "dimension names"
table <- table(loan$application_type,
               loan$homeshowership,
               dim = c("application type",
                      "home ownership"))
print(table)

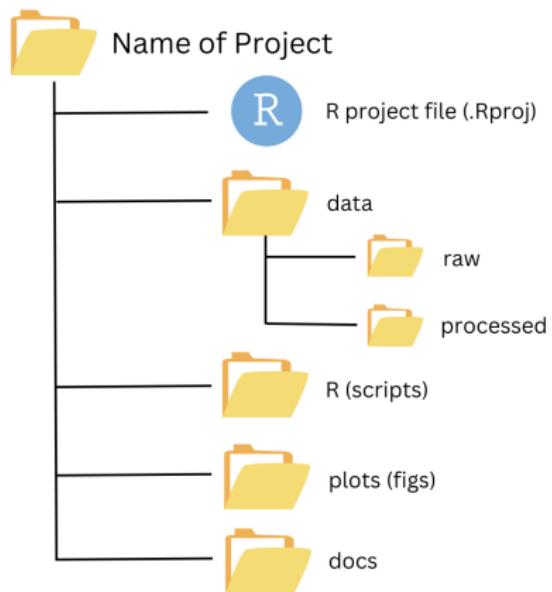
## [1] "Top Level" 
Console Terminal Background Jobs
R 4.4.1 - C:\Users\danny\Dropbox\Graduate\统计学\2025FAI\2025R code\2025FAI_Statistics_TA_session_wk3.R
> print(scatter_plot)
> scatter_plot_scale <- scatter_plot +
```

# RStudio project

- **RStudio project:** A tool to organize all files related to a task.
  - Sets the root folder as the wd for all R sessions within the project.
  - No more `setwd()`!
  - Keep history, options ... etc.
  - More advantages. e.g. portable path: `here()` function.
- Open a project by (File → New Project) at the top left corner of RStudio.
- When should I open a RStudio project?
  - If the task lasts for multiple periods, or with multiple files.
- Recommend to open a RStudio project for the TA Session Practices!

# Folder Organization

- After opening a project, it is important to have well-organized folders.
- Different types of things (scripts, data, output, etc.) should be saved in different folders!



# Coding styles: Naming

- Only lowercase letters, numbers, and `_` within a name.
- The name should be readable!<sup>1</sup>

```
# Good
lesson_one
lesson_1
```

```
# Bad
LessonOne
lessonone
one
a <- 1
```

---

<sup>1</sup>See [this website](#) for more on coding styles.

# Coding styles: Spacing

- Always put a space after a comma, never before.
- Do not put spaces inside or outside parentheses.

```
# Good
x[, 1]
mean(x, na.rm = TRUE)

# Bad
x[,1]
x[ ,1]
x[ , 1]
mean (x, na.rm = TRUE)
mean( x, na.rm = TRUE )
```

# Coding styles: Function Indents

- Indent the argument name with a single indent (two spaces).
- Don't put + or , in a new line!

```
# Good
long_function_name <- function(
  a = "a long argument",
  b = "another argument",
  c = "another long argument"
) {
}

# Bad
long_function_name <- function(a = "a long argument",
  b = "another argument",
  c = "another long argument"
) {
}

long_function_name <- function(a = "a long argument", b = "ano
```

# Create Contingency Table: table()

application type	home ownership		
	Rent	Mortgage	Own
individual	3496	3839	1170
joint	362	950	183

Table: Homeownership by application type

# Create Contingency Table: table()

- **Syntax:**

```
# dnn = dimension names  
A <- table(..., dnn = ...)
```

- **Example:**

```
loan <- read.csv("loans_full_schema.csv")  
table <- table(loan$application_type,  
                loan$homeownership,  
                dnn = c("application type",  
                       "home ownership"  
                      ))  
                )
```

# Add Sum Row and Sum Column: addmargins()

application_type	homeownership				Total
	Rent	Mortgage	Own		
individual	3496	3839	1170		8505
joint	362	950	183		1495
Total	3858	4789	1353		10 000

Table: Homeownership by application type, with sum row and sum column

# Add Sum Row and Sum Column: addmargins()

- **Syntax:**

```
A <- addmargins(table)
```

- **Example:**

```
table_withsum <- addmargins(table)
```

# Remove Everything: rm(list = ls())

- `rm(list = ls())` removes everything inside the global environment.
  - Useful when you want to re-run the whole code.
- `ls()` returns a character vector of object names in the current environment.

```
x <- 1
y <- "hi"
ls()
# [1] "x" "y"
```

# Barplot: geom\_bar()

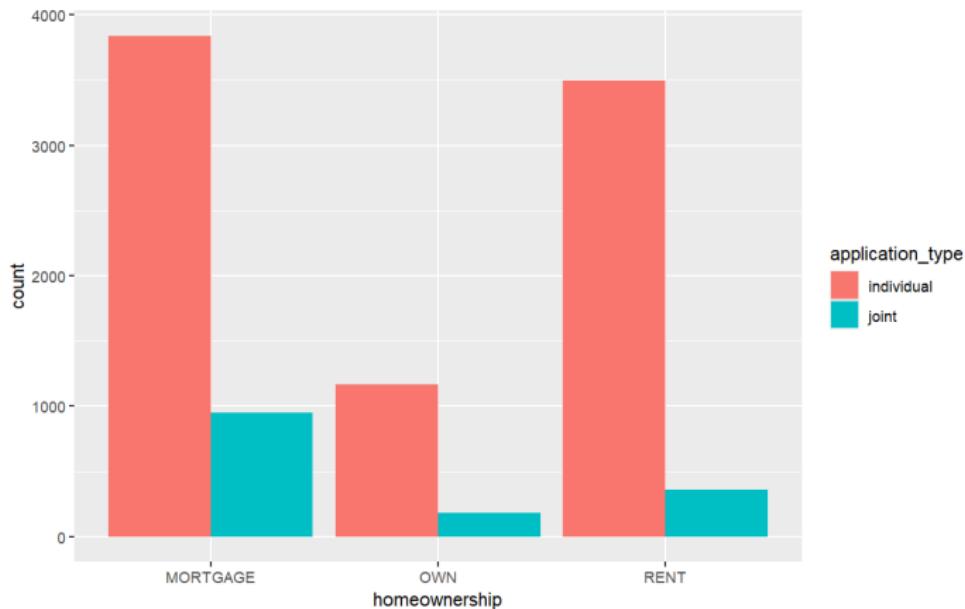


Figure: Bar plot of homeownership by application type.

# Barplot: geom\_bar()

- **Syntax:**

```
ggplot(loan, aes(x = ..., fill = ...)) +  
  geom_bar(position = ...)
```

- **Example:**

```
ggplot(loan, aes(x = homeownership, fill = application_type)) +  
  geom_bar(position = "dodge") # stack, fill, ...
```

# Add Title and Axis Labels: labs()

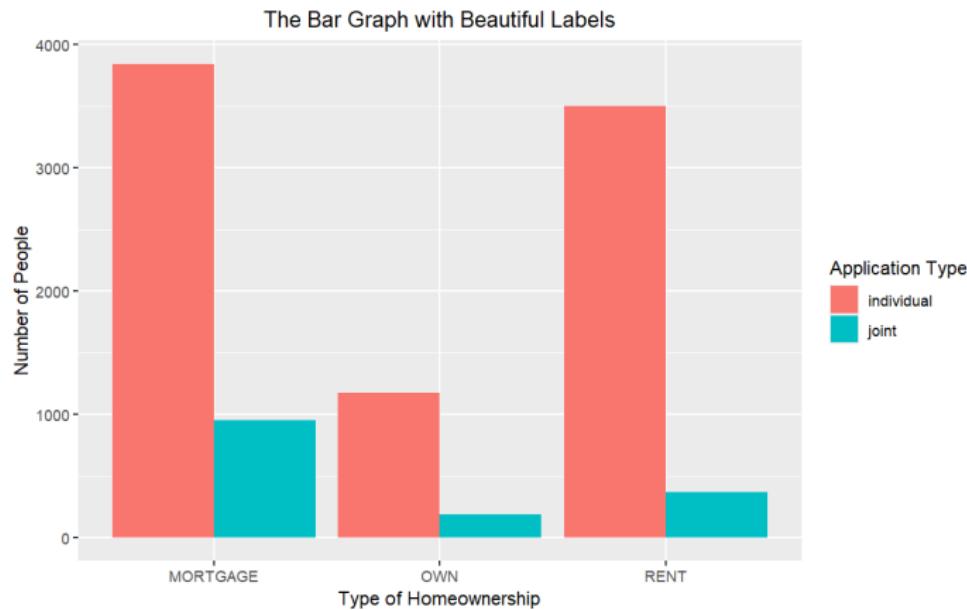


Figure: Same bar plot, **but with title and labels.**

# Add Title and Axis Labels: labs()

- **Syntax:**

```
Your_Plot +  
  labs(x = ..., y = ..., title = ...) +  
  theme(plot.title = element_text(hjust = 0.5))
```

- **Example:**

```
ggplot(loan, aes(x = homeownership, fill = application_type)) +  
  geom_bar(position = "dodge") +  
  labs(  
    x = "Type of Homeownership",  
    y = "Number of People",  
    title = "The Bar Graph with Beautiful Labels",  
    fill = "Application Type"  
  ) +  
  theme(plot.title = element_text(hjust = 0.5))
```

# Adjust Scale of x, y Axes: scale\_x\_continuous()

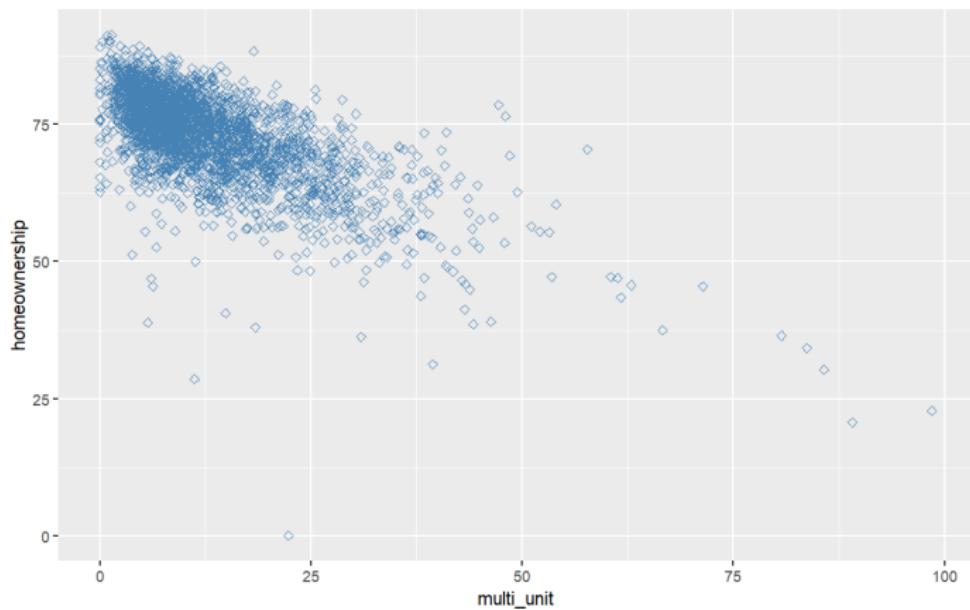


Figure: Scatter plot last week

# Adjust Scale of x, y Axes: scale\_x\_continuous()

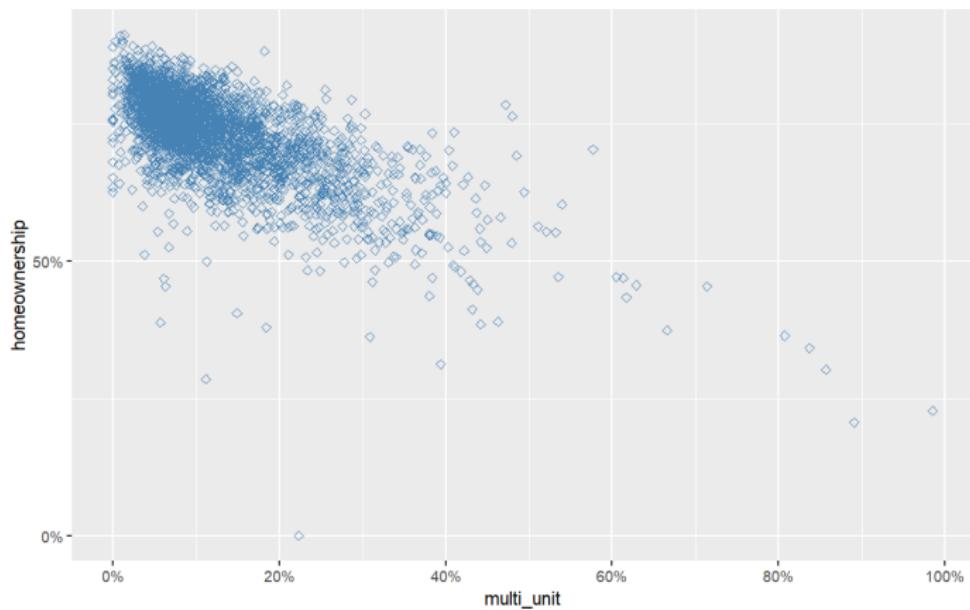


Figure: Scatter plot last week, with adjusted scale.

# Add Title and Axis Labels: labs()

- **Syntax:**

```
Your_Plot +  
  scale_x_continuous(breaks = ..., labels = ...) +  
  scale_y_continuous(breaks = ..., labels = ...)
```

- **Example:**

```
... +  
  scale_x_continuous(  
    breaks = c(0, 20, 40, 60, 80, 100),  
    labels = c("0%", "20%", "40%", "60%", "80%", "100%")  
  ) +  
  scale_y_continuous(  
    breaks = c(0, 50, 100),  
    labels = c("0%", "50%", "100%")  
  )
```