Statistics with Recitation: TA Session

Danny Po-Hsien Kang (康柏賢)

November 18, 2025

Today's agenda

1. Data Wrangling
    - `summarize()`
    - `group_by()`
2. Hypothesis Testing: Chi-Squared Test
    - `prop.table()`
    - `chisq.test()`

Today's Dataset

- Please download the two datasets from the OpenIntro website.
  - `earthquake.csv`: Record some notable earthquakes that happened in the 20th century.
  - `offshore_drilling.csv`: A survey data randomly asking registered voters in California for their position on drilling for oil and natural gas off the Coast of California.
- After that, import the data

```r
earthquake <- read.csv("data/earthquakes.csv")
oil <- read.csv("data/offshore_drilling.csv")
```

Summarize Data: summarize()

- **Data:**

```
> earthquake
    year    month day richter       area    region deaths
1   1902    April  19     7.5        ... Guatemala   2000
2   1902 December  16     6.4 Uzbekistan    Russia   4700
3   1903    April  28     7.0  Malazgirt    Turkey   3500
4   1903      May  28     5.8       Gole    Turkey   1000
5   1905    April   4     7.5     Kangra     India  19000
```

Summarize Data: summarize()

- **Syntax:**

```
dataset %>% summarize(
    mean_var1 = mean(var1),
    sd_var1 = sd(var1),
    max_var2 = max(var2),
    min_var3 = min(var3),
    ...
)
```

Summarize Data: summarize()

- **Example:**

```
earthquake %>%
    na.omit() %>%
    summarize(
        avg_scale = mean(richter),
        avg_death = mean(deaths)
    )
```

- **Output:**

```
  avg_scale avg_death
1  7.166379   18029.1
```

Group Data by Variables: group_by()

- **Syntax:**

```
data %>%
  group_by(var) %>%
  summarize(stats = func())
```

- **Example:**

```
earthquake %>%
  group_by(region) %>%
  summarize(
    max_scale = max(richter),
    min_scale = min(richter),
    avg_death = mean(deaths)
    ) %>%
  arrange(desc(avg_death))
```

Group Data by Variables: group_by()

- **Output:**

```
  region        max_scale min_scale avg_death
  <chr>             <dbl>     <dbl>     <dbl>
1 Peru                7.9       7.3    350700
2 Turkmenistan        7.3       7.3    110000
3 China               8         5.9     40236.
4 Armenia             6.8       6.8     25000
5 Italy               7.2       6.5     21950.
6 Japan               8.4       6.9     20324.
7 Pakistan            8         6.2     13100
```

Group Data by Variables: group_by()

- group_by() adds a grouping attribute to the data.
- Many dplyr verbs then behave per group, not on the full data:
  - summarize(), mutate(), filter(), etc.
- Suppose you want to add summary statistics to the dataframe by region for each row:

```
earthquake %>%
  group_by(region) %>%
  mutate(
    max_scale = max(richter),
    min_scale = min(richter),
    avg_death = mean(deaths)
    ) %>%
  arrange(desc(avg_death)) %>%
  ungroup()
```

Group Data by Variables: group_by()

- **Output:**

	year		deaths	region	max_scale	min_scale	avg_death
	<int>	...	<int>	<chr>	<dbl>	<dbl>	<dbl>
1	1946	...	1400	Peru	7.9	7.3	350700
2	1970	...	700000	Peru	7.9	7.3	350700
3	1948	...	110000	Turk...	7.3	7.3	110000
4	1917	...	1800	China	8	5.9	40236.
5	1918	...	1000	China	8	5.9	40236.
6	1920	...	200000	China	8	5.9	40236.
7	1923	...	3500	China	8	5.9	40236.

Group Data by Variables: group_by()

- If you do not want the group structure in data, use ungroup().
  - Forget using ungroup() sometimes can cause tricky problems.
- Suppose you want to find the group with the largest revenue:

```
sales <- data.frame(region = c("N","N","S","S","S","S"),
                    store  = c("A","B","A","A","B","B"),
                    sales  = c(100,120,90,110,150,140))
# forget to ungroup()
by_region_store <- sales %>%
  group_by(region, store) %>%
  summarise(total_sales = sum(sales))

#   region store total_sales
# 1 N      A             100
# 2 N      B             120
# 3 S      A             200
# 4 S      B             290
```

Group Data by Variables: group_by()

- Then you use `filter()` to select the rows with largest mean:

```
by_region_store %>% filter(total_sales == max(total_sales))
```

- Output:

```
  region store total_sales
1 N      B             120
2 S      B             290
```

- We get two columns here!
- `summarise()` defaults to `.groups = "drop_last"`, which drops the last group.
  - `by_region_store` is now group by `region`.
- To avoid this, add `ungroup()` after `summarise()`
  - or use `summarise(..., .groups = "drop")`

Create Proportion Table: prop.table()

- Let's first take a look at the oil data:

```
> oil
        v1           v2
1 position college_grad
2  support          yes
3  support          yes
4  support          yes
5  support          yes
6  support          yes
7  support          yes
```

- Our colnames are placed in the first row...
  - This often happens in raw data.

Create Proportion Table: prop.table()

- Quick fix:

```
colnames(oil) <- oil[1, ]
oil <- oil[-1, ]
```

- After fixing:

```
> oil
    position college_grad
2    support          yes
3    support          yes
4    support          yes
5    support          yes
6    support          yes
7    support          yes
```

Create Proportion Table: prop.table()

- Then, let's construct a contingency table:

```
count_table <- table(oil$position,
                     oil$college_grad
                     )
```

- **Output:**

```
            no yes
do_not_know 131 104
oppose      126 180
support     132 154
```

Create Proportion Table: prop.table()

- **Syntax:**

```
pct_table <- prop.table(contingency_table,
                        margin = ...)
```

- **Example:**

```
# by row
pct_table_1 <- prop.table(count_table, margin = 1)

# by column
pct_table_2 <- prop.table(count_table, margin = 2)
```

Create Proportion Table: prop.table()

- **Output:** pct_table_1 (by row)

```
                  no       yes
do_not_know 0.5574468 0.4425532
oppose      0.4117647 0.5882353
support     0.4615385 0.5384615
```

- **Output:** pct_table_2 (by column)

```
                  no       yes
do_not_know 0.3367609 0.2374429
oppose      0.3239075 0.4109589
support     0.3393316 0.3515982
```

Chi-Squared Test: chisq.test()

- **Syntax:**

```
chisq.test(contingency_table)
```

- **Example:**

```
chisq.test(count_table)
```

Chi-Squared Test: chisq.test()

- **Output:**

```
        Pearson's Chi-squared test

data:  count_table
X-squared = 11.461, df = 2, p-value = 0.003246
```

- **Remark:**
  - The results of `chisq.test()` will be saved in a list.
  - Don't use the proportion table to perform the Chi-squared test.
  - Because the Chi-squared test needs the original count in the data to compute the Chi-squared test statistic.