AE 12: Rectangling data from the PokéAPI

Suggested answers

APPLICATION EXERCISE ANSWERS

MODIFIED
October 18, 2024

Packages

We will use the following packages in this application exercise.

- **tidyverse**: For data import, wrangling, and visualization.
- **jsonlite**: For importing JSON files

```r
library(tidyverse)
library(jsonlite)
```

Gotta catch em' all!

Pokémon (also known as **Pocket Monsters**) is a Japanese media franchise consisting of video games, animated series and films, a trading card game, and other related media.[1] The PokéAPI contains detailed information about each Pokémon, including their name, type, and abilities. In this application exercise, we will use a set of JSON files containing API results from the PokéAPI to explore the Pokémon universe.

[1] Source: Wikipedia

Importing the data

`data/pokedex.json` and `data/types.json` contain information about each Pokémon and the different types of Pokémon, respectively. We will use `read_json()` to import these files.

```r
pokemon <- read_json(path = "data/pokemon/pokedex.json")
types <- read_json(path = "data/pokemon/types.json")
```

Your turn: Use `View()` to interactively explore each list object to identify their structure and the elements contained within each object.

Unnesting for analysis

For each of the exercises below, use an appropriate rectangling procedure to `unnest_*()` one or more lists to extract the required elements for analysis.

How many Pokémon are there for each primary type?

Your turn: Use each Pokemon's **primary type** (the first one listed in the data) to determine how many Pokémon there are for each type, then create a bar chart to visualize the distribution.

Tip

Examine the contents of each list object to determine how the relevant variables are structured so you can plan your approach.

There are (at least) three ways you could approach this problem.

1. Use `unnest_wider()` twice to extract the primary type from the pokemon list and generate a frequency count.
2. Use `unnest_wider()` and `hoist()` to extract the primary type from the pokemon list and generate a frequency count.
3. Use `unnest_wider()` and `unnest_longer()` to extract the primary type from the pokemon list and generate a frequency count.

Pick one and have at it!

Note

Fancy a challenge? Label each Pokémon type in both **English** and **Japanese**.

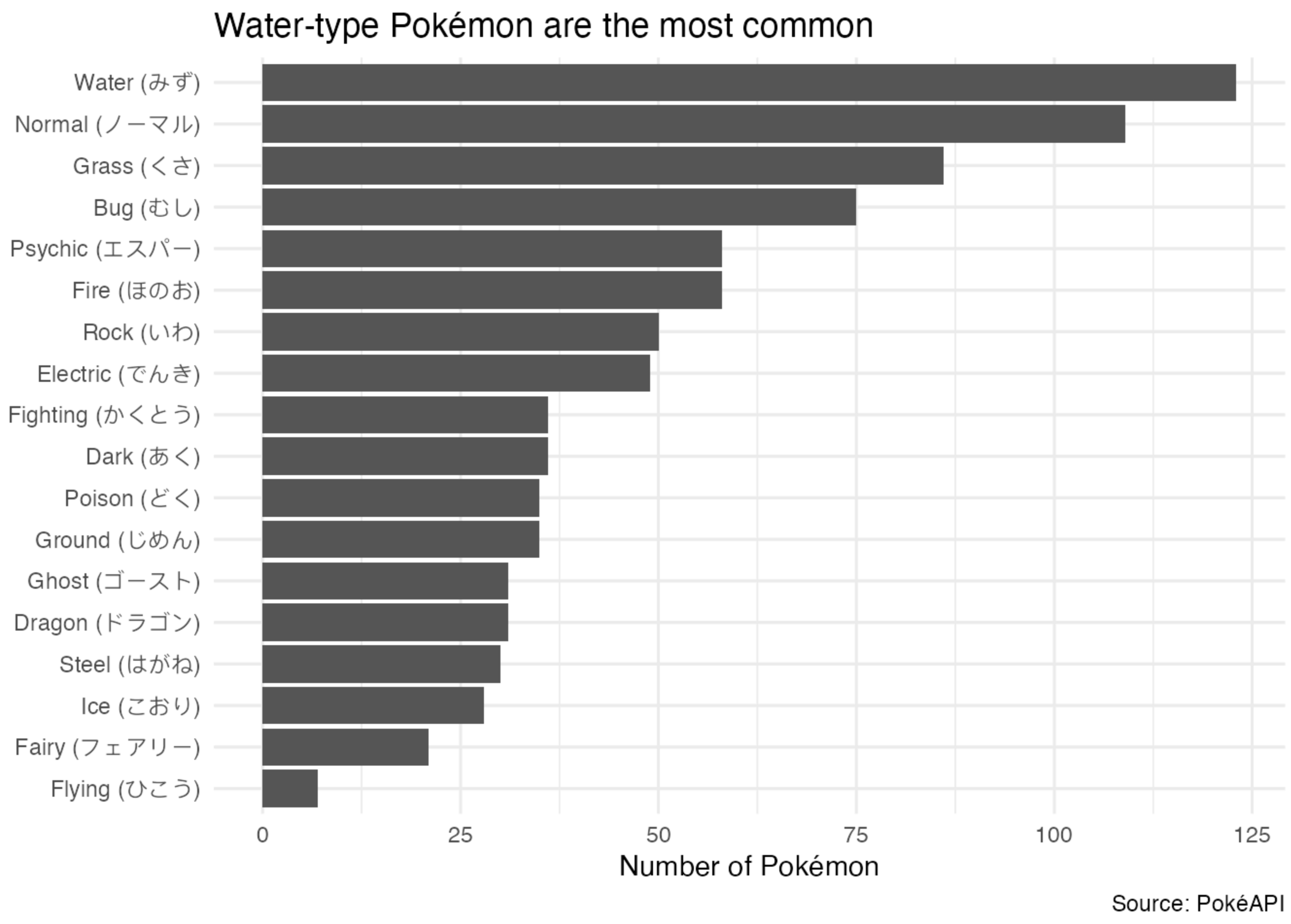
```
# extract the primary type from the pokemon list and generate a frequency count
## using hoist()
poke_types <- tibble(pokemon) |>
  # expand so each pokemon variable is in its own column
  unnest_wider(pokemon) |>
  # extract the pokemon's primary type from the type column
  hoist(.col = type, main_type = 1L) |>
  # generate frequency count
  count(main_type)

## using unnest_wider() twice
poke_types <- tibble(pokemon) |>
  # expand so each pokemon variable is in its own column
  unnest_wider(pokemon) |>
  # expand the type column so each type is in its own column
  unnest_wider(type, names_sep = "_") |>
  rename(main_type = type_1) |>
  # generate frequency count
  count(main_type)
```

```r
## using unnest_wider() and unnest_longer()
poke_types <- tibble(pokemon) |>
  # expand so each pokemon variable is in its own column
  unnest_wider(pokemon) |>
  # expand the type column so each type is in its own row
  unnest_longer(type) |>
  # keep just the first row for each pokemon
  slice_head(n = 1, by = id) |>
  # generate frequency count
  count(main_type = type)

# extract english and japanese names for types
types_df <- tibble(types) |>
  unnest_wider(types)

# combine poke_types with types_df and create a name column that includes both
# english and japanese
left_join(x = poke_types, y = types_df, by = join_by(main_type == english)) |>
  mutate(
    name = str_glue("{main_type} ({japanese})"),
    name = fct_reorder(.f = name, .x = n)
  ) |>
  ggplot(mapping = aes(x = n, y = name)) +
  geom_col() +
  labs(
    title = "Water-type Pokémon are the most common",
    x = "Number of Pokémon",
    y = NULL,
    caption = "Source: PokéAPI"
  ) +
  theme_minimal()
```



Which primary type of Pokémon are strongest based on total number of points?

Your turn: Use each Pokémon's **base stats** to determine which **primary type** of Pokémon are strongest based on the total number of points. Create a boxplot to visualize the distribution of total points for each primary type.

Tip

To calculate the sum total of points for each Pokémon's base stats, there are two approaches you might consider. In either approach you first need to get each Pokémon's variables into separate columns and extract the primary type.

1. Use `unnest_wider()` to extract the base stats, then calculate the sum of the base stats.
2. Use `unnest_longer()` to extract the base stats, then calculate the sum of the base stats.

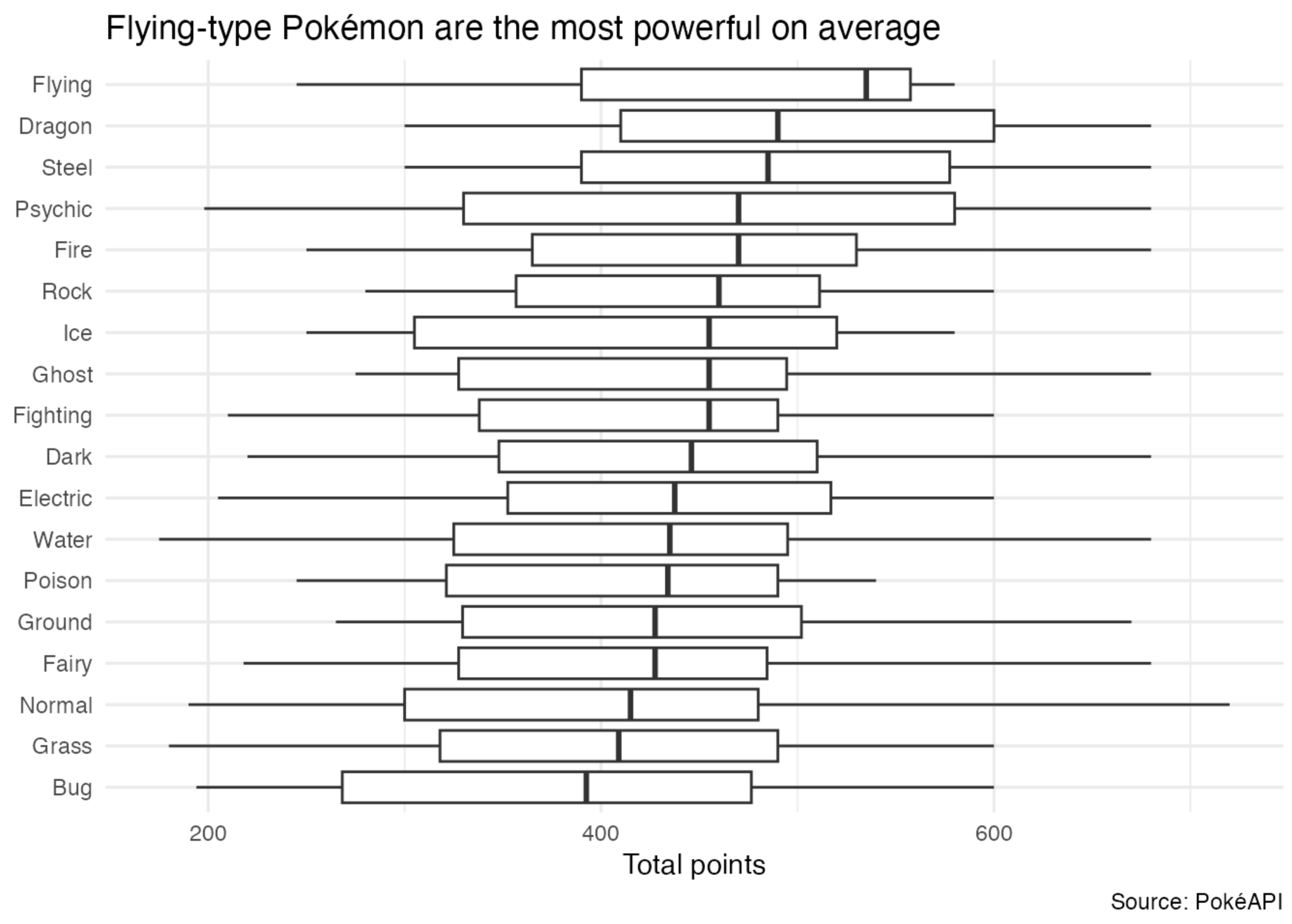
```r
# base stats in one column per stat
pokemon_points <- tibble(pokemon) |>
  # one column per variable
  unnest_wider(pokemon) |>
  # extract the pokemon's primary type from the type column
  hoist(.col = type, main_type = 1L) |>
  # expand to get base stats
  unnest_wider(base) |>
  # for each row, calculate the sum of HP:Speed
  rowwise() |>
  mutate(total = sum(c_across(cols = HP:Speed), na.rm = TRUE), .before = HP) |>
  ungroup() |>
  # exclude pokemon with total = 0 - means we don't have stats available
  filter(total != 0) |>
  select(id, main_type, total)
pokemon_points
```

```
# A tibble: 809 × 3
      id main_type total
   <int> <chr>     <int>
 1     1 Grass       318
 2     2 Grass       405
 3     3 Grass       525
 4     4 Fire        309
 5     5 Fire        405
 6     6 Fire        534
 7     7 Water       314
 8     8 Water       405
 9     9 Water       530
10    10 Bug         195
# ℹ 799 more rows
```

```
# base stats in one row per pokemon per stat
pokemon_points <- tibble(pokemon) |>
  # one column per variable
  unnest_wider(pokemon) |>
  # extract the pokemon's primary type from the type column
  hoist(.col = type, main_type = 1L) |>
  # expand to get base stats
  unnest_longer(base,
    values_to = "points",
    indices_to = "stat"
  ) |>
  # calculate sum of points for each pokemon
  summarize(total = sum(points, na.rm = TRUE), .by = c(id, main_type))
pokemon_points
```

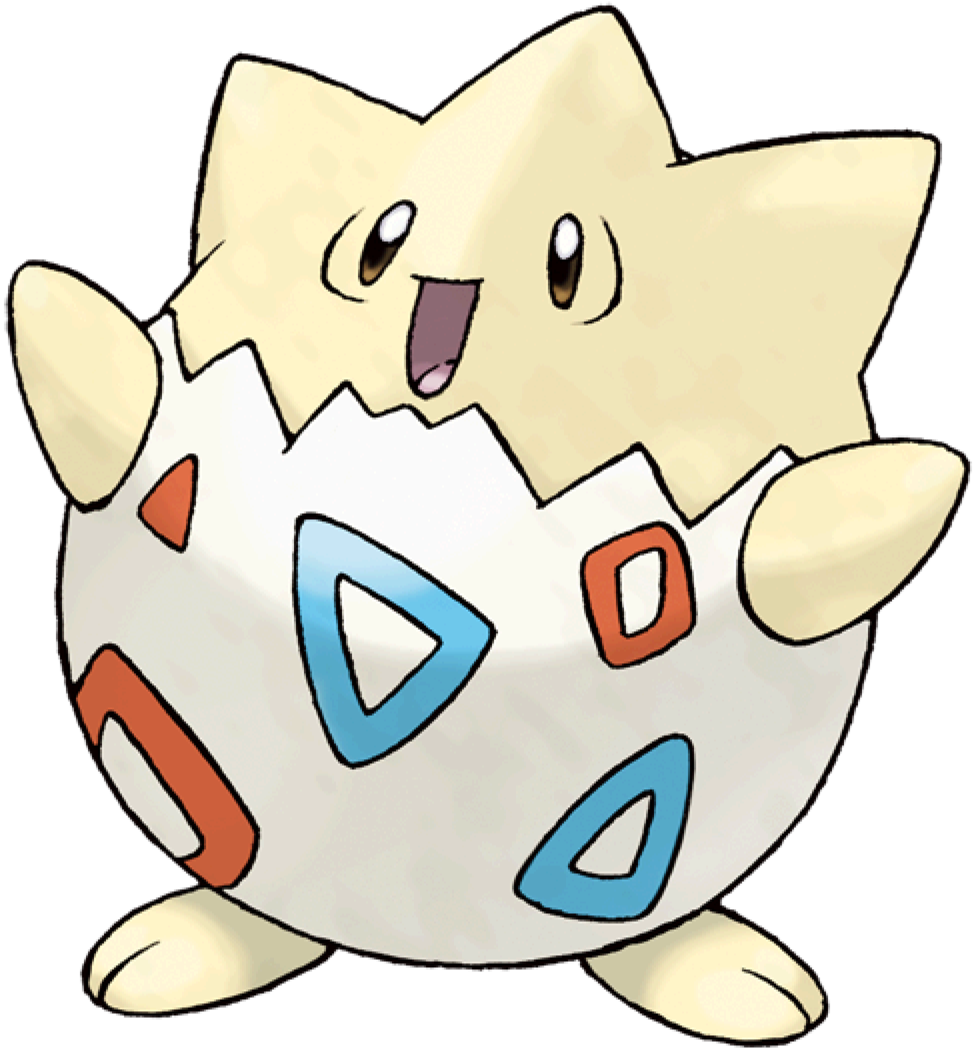
```
# A tibble: 809 × 3
      id main_type total
   <int> <chr>     <int>
 1     1 Grass       318
 2     2 Grass       405
 3     3 Grass       525
 4     4 Fire        309
 5     5 Fire        405
 6     6 Fire        534
 7     7 Water       314
 8     8 Water       405
 9     9 Water       530
10    10 Bug         195
# ℹ 799 more rows
```

```r
pokemon_points |>
  # order the boxplots meaningfully
  mutate(main_type = fct_reorder(.f = main_type, .x = total)) |>
  # generate the plot
  ggplot(mapping = aes(x = total, y = main_type)) +
  geom_boxplot() +
  labs(
    title = "Flying-type Pokémon are the most powerful on average",
    x = "Total points",
    y = NULL,
    caption = "Source: PokéAPI"
  ) +
  theme_minimal()
```



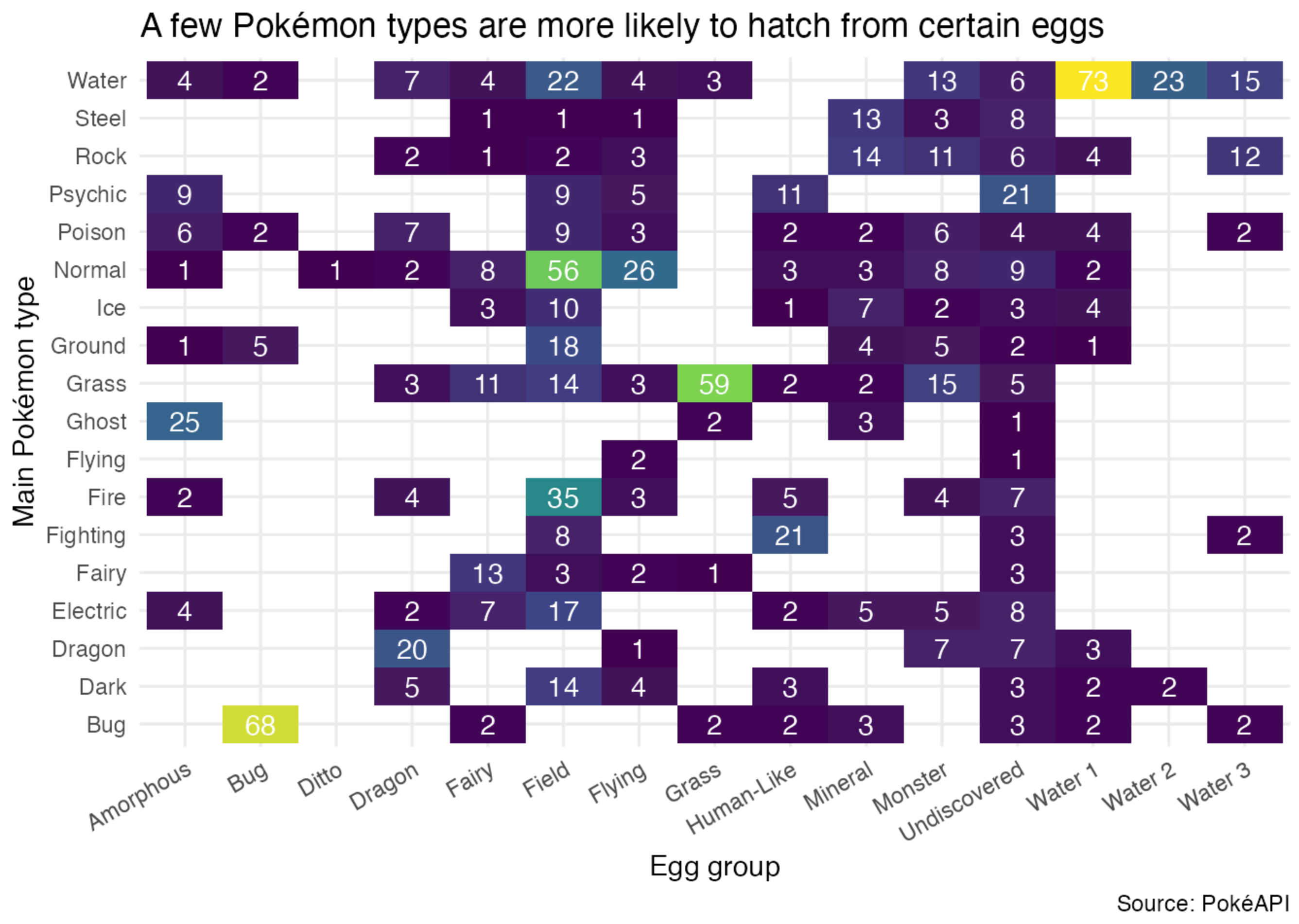
From what types of eggs do Pokémon hatch? (Bonus!)

In Generation II, Pokémon introduced the concept of breeding, whereby Pokémon can produce offspring. In Generation III, Pokémon eggs were introduced, which can be hatched to produce a Pokémon.



Togepi was the first Pokémon to be introduced as an egg.

Use each Pokémon's egg group to determine from what types of eggs Pokémon hatch. Create a heatmap like the one below to visualize the distribution of egg groups for each primary type.



Tip

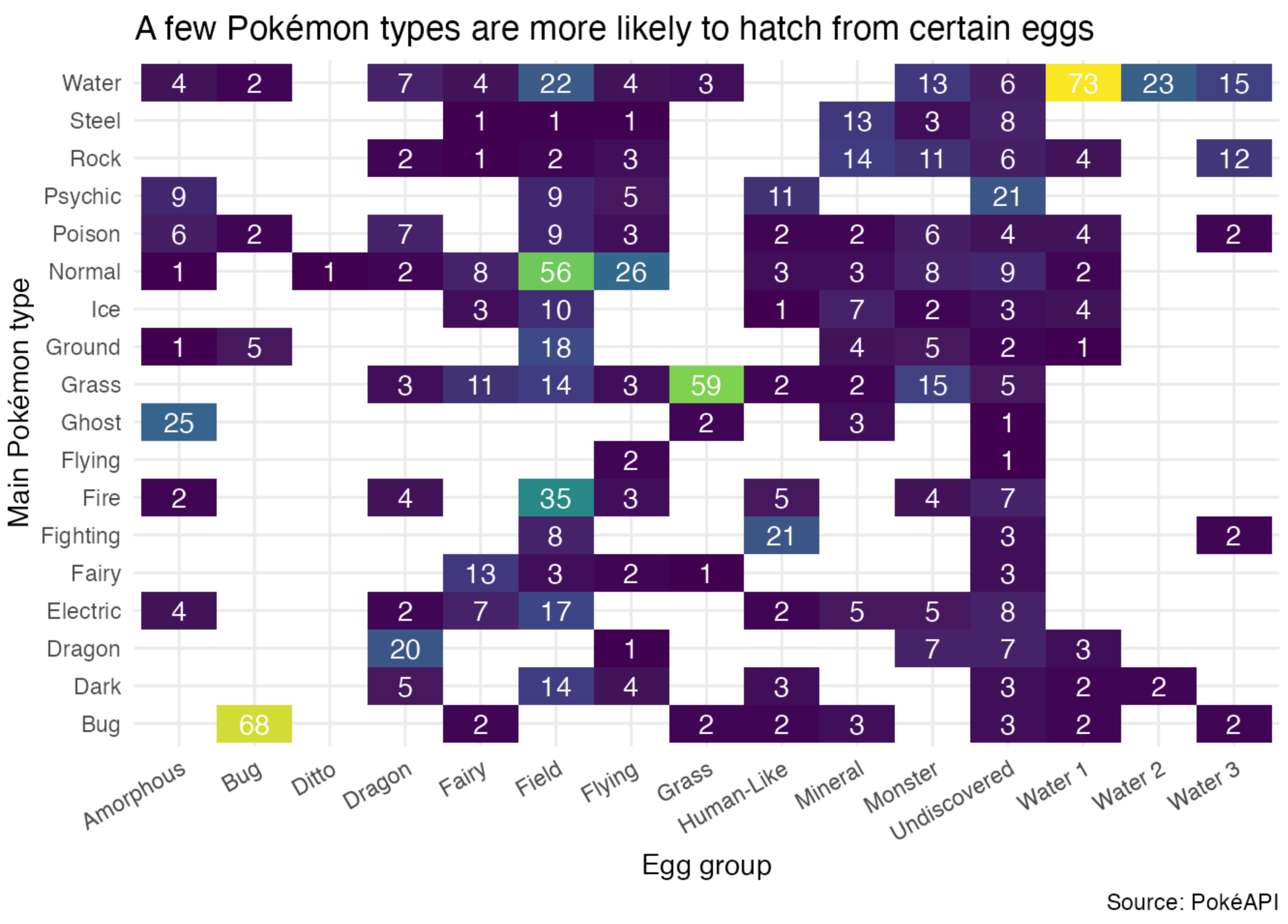
Consider using `hoist()` to extract the main type and egg group from each Pokémon.

```
tibble(pokemon) |>
  unnest_wider(pokemon) |>
  # extract the main type
  hoist(
    .col = type, main_type = 1L
  ) |>
  # extract the type of eggs from which the pokemon can hatch
  hoist(
    .col = profile, egg = "egg"
  ) |>
  select(main_type, egg) |>
  # some pokemon have more than one egg group, need to unnest longer
  unnest_longer(egg) |>
  # count the number of type-egg pairings
  count(main_type, egg) |>
  # draw the plot
  ggplot(mapping = aes(x = egg, y = main_type, fill = n)) +
  geom_tile() +
  geom_text(mapping = aes(label = n), color = "white") +
  scale_fill_viridis_c() +
  theme_minimal() +
  labs(
    title = "A few Pokémon types are more likely to hatch from certain eggs",
    x = "Egg group",
    y = "Main Pokémon type",
    caption = "Source: PokéAPI"
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 30, hjust = 1)
  )
```

Acknowledgments

- JSON data files obtained from `Purukitto/pokemon-data.json`

Session information


A few Pokémon types are more likely to hatch from certain eggs
Main Pokémon type
Water
Steel
Rock
Psychic
Poison
Normal
Ice
Ground
Grass
Ghost
Flying
Fire
Fighting
Fairy
Electric
Dragon
Dark
Bug
Amorphous
Bug
Ditto
Dragon
Fairy
Field
Flying
Grass
Human-Like
Mineral
Monster
Undiscovered
Water 1
Water 2
Water 3
Egg group
Source: PokéAPI