R for Librarians `dplyr` Exercises

Alistair Bailey

Last Updated on 2024-04-30

Table of contents

0.1 Importing and inspecting data

Exercise 1: Import the data contained in csv file to an object called `books` from: https://raw.githubusercontent.com/ab604/library-r/main/data/books-2024-04-30.csv

> **Tip**
>
> You can read data directly from paths to on-line datasets with `read_csv()` as you would for locally stored `csv` files.

```
# Exercise 1: Import the data contained in csv file to an object called `books`
books <- read_csv("https://raw.githubusercontent.com/ab604/library-r/main/data/books-2024-04-
```

Exercise 2: How many books (rows) and how many variables are there?

```
# Exercise 2: How many books (rows) and how many variables are there?
# 90 books with 6 columnns
glimpse(books)
```

Exercise 3: How many different genres and types are there?

```
# Exercise 3: How many different genres and types are there?
# 7 genres
books |>
  distinct(genre)
# 2 types
books |>
  distinct(type)
```

Exercise 4: What is the range of publication dates?

```
# Exercise 4: What is the range of publication dates?
books |>
  reframe(pub_year_range = range(year))
```

0.2 Filter

Exercise 1: Filter books published after the year 2000

```
# Exercise 1: Filter books published after the year 2000
books |>
  filter(year > 2000)
```

Exercise 2: Filter non-fiction books with more than 500 pages

```
# Exercise 2: Filter non-fiction books with more than 500 pages
books |>
  filter(type == "Non-fiction", pages > 500)
```

Exercise 3: Filter fiction books in the “Literature” genre published before 1900 or after 2000

```
# Exercise 3: Filter fiction books in the "Literature" genre published before 1900 or after 2
books |>
  filter(
    type == "Fiction",
    genre == "Literature",
    year < 1900 | year > 2000
  )
```

Filter exercises explanations

Explanations:

1. Exercise 1 is a simple filtering task that requires the user to filter books published after the year 2000. The user needs to use the `filter()` function with the condition `year > 2000`.
2. Exercise 2 is slightly more challenging as it involves filtering based on multiple conditions. The user needs to filter non-fiction books with more than 500 pages. They should use the `filter()` function with the conditions `type == "Non-fiction"` and `pages > 500`, separated by a comma.
3. Exercise 3 is the most complex among the three exercises. It requires the user to filter fiction books in the "Literature" genre that were published either before 1900 or after 2000. The user needs to use the `filter()` function with multiple conditions: `type == "Fiction"`, `genre == "Literature"`, and `year < 1900 | year > 2000`. The `|` operator represents the logical OR condition.

0.3 Select

Exercise 1: Select the title and year columns

```
# Exercise 1: Select the title and year columns
books |>
  select(title, year)
```

Exercise 2: Select the columns from title to pages

```
# Exercise 2: Select the columns from title to pages
books |>
  select(title:pages)
```

Exercise 3: Select the title column and all columns that end with "er"

```
# Exercise 3: Select the title column and all columns that end with "er"
books |>
  select(title, ends_with("er"))
```

Select exercises explanations

1. Exercise 1 is a straightforward task that requires the user to select specific columns by name. The user needs to use the `select()` function and provide the column names `title` and `year` as arguments.

2. Exercise 2 introduces the concept of selecting a range of columns. The user needs to select all columns from `title` to `pages` using the colon (`:`) operator within the `select()` function.

3. Exercise 3 combines selecting a specific column and using a helper function to select columns based on a pattern. The user needs to select the `title` column explicitly and then use the `ends_with()` helper function within `select()` to select all columns that end with the string "er".

0.4 Mutate

Exercise 1: Create new columns "title_letters" and "title_words" based on the "title" column, that count the numbers of letters and words in the title respectively.

```
# Exercise 2: Create new columns "title_letters" and "title_words" based on the
# "title" column, that count the numbers of letters and words in the title
# respectively.
books |>
  mutate(
    title_letters = str_count(title),
    title_words = str_count(title, "\\w+")) |>
  select(title,title_letters, title_words)
```

Exercise 2: Create a new column called "book_length" based on the number of pages. Categories should be, "Short" when pages $<$ 300, "Medium" when pages $<=$ 300 & $>=$ 500, and "Long" when pages $>=$ 500.

```
# Exercise 2: Create a new column called "pages_category" based on the number
# of pages
books |>
  mutate(book_length = case_when(
    pages < 300 ~ "Short",
    pages >= 300 & pages < 500 ~ "Medium",
    pages >= 500 ~ "Long"
  )) |>
  select(title, pages, book_length)
```

Mutate exercises explanations

1. Exercise 1 involves creating multiple new columns using `mutate()` and applying functions to existing columns. The user needs to create two new columns: "title_letters" (the number of characters in the title) using the `str_count()` function, and "title_words" (the number of words in the title) using the `str_count()` function with the additional argument `"\\w+"` from the `stringr` package to count the number of word characters.

2. Exercise 2 demonstrates the use of `case_when()` within `mutate()` to create a new column based on conditional logic. The user needs to create a new column called "book_length" that categorizes books as "Short" (less than 300 pages), "Medium" (between 300 and 499 pages), or "Long" (500 pages or more) based on the number of pages.

0.5 Combining filter, select and mutate

Exercise 1:

```
# Exercise 1: Filter fiction books, select the title and pages columns, and create a new colu
books |>
  filter(type == "Fiction") |>
  select(title, pages, year) |>
  mutate(pages_per_year = pages / (2023 - year))
```

Exercise 2:

```
# Exercise 2: Filter books with more than 500 pages, select the title, genre, and publisher c
books |>
  filter(pages > 500) |>
  select(title, genre, publisher) |>
  mutate(title_length = nchar(title))
```

Exercise 3:

```
# Exercise 3: Filter non-fiction books published after 2000, select the title, pages, and yea
books |>
  filter(type == "Non-fiction", year > 2000) |>
  select(title, pages, year) |>
  mutate(
    decade = paste0(floor(year / 10) * 10, "s"),
```

```
    pages_category = case_when(
      pages < 300 ~ "Short",
      pages >= 300 & pages < 500 ~ "Medium",
      pages >= 500 ~ "Long"
    )
  )
```

Explanations:

1. Exercise 1 combines `filter()`, `select()`, and `mutate()` functions. The user needs to filter fiction books, select the "title", "pages", and "year" columns, and create a new column called "pages_per_year" by dividing the number of pages by the difference between the current year (2023) and the publication year.
2. Exercise 2 involves filtering books with more than 500 pages, selecting the "title", "genre", and "publisher" columns, and creating a new column called "title_length" using the `nchar()` function to calculate the number of characters in the title.
3. Exercise 3 requires the user to filter non-fiction books published after 2000, select the "title", "pages", and "year" columns, and create two new columns: "decade" (the decade of publication using the `floor()` function and concatenating "s" to the result) and "pages_category" (categorizing books as "Short", "Medium", or "Long" based on the number of pages using `case_when()`).

These exercises combine the concepts learned in the previous exercises and test the user's ability to chain multiple functions together using the pipe operator (`|>`). They involve filtering data based on conditions, selecting specific columns, and creating new columns using various functions and conditional logic.

0.6 Grouped summaries

Exercise 1:

```
# Exercise 1: Calculate the average number of pages for each book type
books |>
  group_by(type) |>
  summarise(avg_pages = mean(pages))
```

Exercise 2:

```r
# Exercise 2: Find the longest book title for each genre and publisher combination
books |>
  select(title, genre, publisher) |>
  mutate(title_length = nchar(title)) |>
  group_by(genre, publisher) |>
  summarise(longest_title = max(title_length))
```

```
`summarise()` has grouped output by 'genre'. You can override using the
`.groups` argument.
```

Exercise 3:

```r
# Exercise 3: Calculate the total number of pages and the count of books for each genre, cons
books |>
  filter(type == "Fiction", year > 1990) |>
  group_by(genre) |>
  summarise(
    total_pages = sum(pages),
    book_count = n()
  )
```

Explanations:

1. Exercise 1 demonstrates the basic usage of `group_by()` and `summarise()`. The user needs to group the books by the "type" column and calculate the average number of pages for each book type using the `mean()` function within `summarise()`.
2. Exercise 2 combines `select()`, `mutate()`, `group_by()`, and `summarise()`. The user needs to select the "title", "genre", and "publisher" columns, create a new column "title_length" using `mutate()` and the `nchar()` function, group the data by "genre" and "publisher", and find the longest book title for each group using `summarise()` and the `max()` function.
3. Exercise 3 involves filtering, grouping, and summarizing data. The user needs to filter fiction books published after 1990, group the filtered data by the "genre" column, and calculate the total number of pages using `sum(pages)` and the count of books using `n()` for each genre within the `summarise()` function.

These exercises test the user's understanding of grouping data using `group_by()`, summarizing data using `summarise()`, and combining these functions with other data manipulation functions like `filter()`, `select()`, and `mutate()`. They involve grouping data based on specific columns, calculating aggregate measures, and applying filters and transformations to the data before grouping and summarizing.