BIP Summer-Intro to R Workshop Activity

Joscelin Rocha-Hidalgo

2023-08-22

I am super excited to walk you through our activity report while I show you how you can use some functions and even create some basic visualizations!

How did I choose the questions?

You may be wondering why I asked you the questions you saw in the online survey. I am a developmental scientist and most of my current research is with children between the ages of 4 and 6 years. Some of them helped me come up with some of the "funner" questions such as the ones asking about superpowers. And of course, I also chose some questions that could give me some variability of responses and one that is very poorly stated so we can do some data cleaning on its content.

Let's load some packages

For the purposes of this workshop, most of our functions will come from packages part of the bigger package called *"Tidyverse"*.

Let's import our dataset

In theory, we could have imported the googlesheets directly to R (definitely possible!) but for simplicity and lack of time, I downloaded the responses as a csv and saved them to our repo under the "data" folder.

Let's clean & rename the variables

The variables exported from the googlesheet are too long and we need to `rename()` them to shorter ones.

```
##  [1] "timestamp"
##  [2] "how_many_books_have_you_read_so_far_this_2023"
##  [3] "on_a_scale_of_1_to_10_how_much_do_you_enjoy_cooking"
##  [4] "how_many_hours_of_sleep_do_you_get_on_average_per_night"
##  [5] "which_superpower_would_you_choose"
##  [6] "if_traveling_was_not_an_issue_you_can_teleport_anywhere_in_a_blink_what_would_be_your_preferre
##  [7] "whats_your_least_favorite_type_of_movie"
##  [8] "what_is_your_age_group"
##  [9] "what_is_your_gender"
## [10] "which_state_do_you_currently_reside_in"
## [11] "what_is_your_racial_background"
## [12] "are_you_latino_a_e_x"
```

Let's separate the demo questions and the "fun" questions into 2 new dataframes

To separate the demo questions and the fun questions into 2 separate dfs, we will use the function `select()`.

1. Assign the fun questions to a df called "fun_data"
2. Assign the demo question to a df called "demo_data"

Let's count the number of people grouped by age and gender

For this task we will use the following functions:

1. group_by
2. count

```
## # A tibble: 10 x 3
## # Groups:   gender, age_group [10]
##    gender            age_group     n
##    <chr>             <chr>     <int>
##  1 Female            18-24        31
##  2 Female            25-34        23
##  3 Female            35-44         6
##  4 Female            45-54         1
##  5 Female            55-64         1
##  6 Male              18-24         9
##  7 Male              25-34         5
##  8 Male              35-44         1
##  9 Non-binary        25-34         3
## 10 Prefer not to say 35-44         3
```

gender	age_group	n
Female	18-24	31
Female	25-34	23
Female	35-44	6
Female	45-54	1
Female	55-64	1
Male	18-24	9
Male	25-34	5
Male	35-44	1
Non-binary	25-34	3
Prefer not to say	35-44	3

Let's count the number of people grouped by age and gender BUT only for Latinx/e/a/o people

For this task we will use the following functions:

1. filter
2. group_by

3. count

4. We can use `<`, `>`, `<=`, `>=`, `==`, `!=`

gender	age_group	n
Female	18-24	5
Female	25-34	5
Female	35-44	2
Male	18-24	3
Male	25-34	2
Prefer not to say	35-44	1

Let’s clean the `state` variable & count the responses

We were very lenient with this question and we gave no parameters on how people should answer it so we received some responses that matched our original expectation but we also received very random responses. Let’s clean some of those

1. select

2. mutate

3. case_when

4. %in%

```
## # A tibble: 51 x 1
##    state
##    <chr>
##  1 PA
##  2 California
##  3 Arizona
##  4 TX
##  5 Alabama
##  6 NY
##  7 NJ
##  8 KY
##  9 DC
## 10 Florida
## # i 41 more rows
```

state_new	n
AL	3
AZ	1
CA	5
CT	1
Ca	1
Canada	1
DC	4
FL	2
Georgia	1

Germany	1
IL	1
Illinois	1
India	1
Indiana	1
Jharkhand	1
KY	4
MA	1
MD	4
MN	4
Maharashtra	1
Michigan	2
Missouri	1
N/A	1
NJ	5
NV	1
NY	9
New York City	1
New york	1
North Carolina	1
Oregon	1
Outside USA, Perú.	1
PA	5
Rhode Island	1
TX	2
Telangana, India	1
Texas	2
Tn	1
Vermont	1
Washington	2
NA	5

Let’s calculate ther sample’s average and SD for the following variables:

1. Books read so far this 2023 (`books_2023`)
2. How much people enjoy cooking (`enjoy_cooking`)
3. Number of hours people sleep (`hrs_sleep`)

Use the functions:

1. `select()`
2. `summarize()` or `summarise()`
3. `mean()`
4. `sd()`
5. `round()`
6. `across()`
7. `everything()`

Then…

Write a text paragraph that updates its numbers if we get new data. (HINT: using embedded code)

The summary paragraph:

Out of the 83 responses we received, the average number of books people have read so far this 2023 was 8.68 books (*SD* = 11.28). Participants also reported sleeping 6.71 hours on average (*SD* = 1.18). In fact, the shortest reported number was 3 and the longest was 9 hours. Finally, when they were asked to report how much they enjoy cooking (1 = "hate it" to 10 = "love it"), the average score was 6 (*SD* = 2.45).

Let's visualize the frequency of chosen superpowers

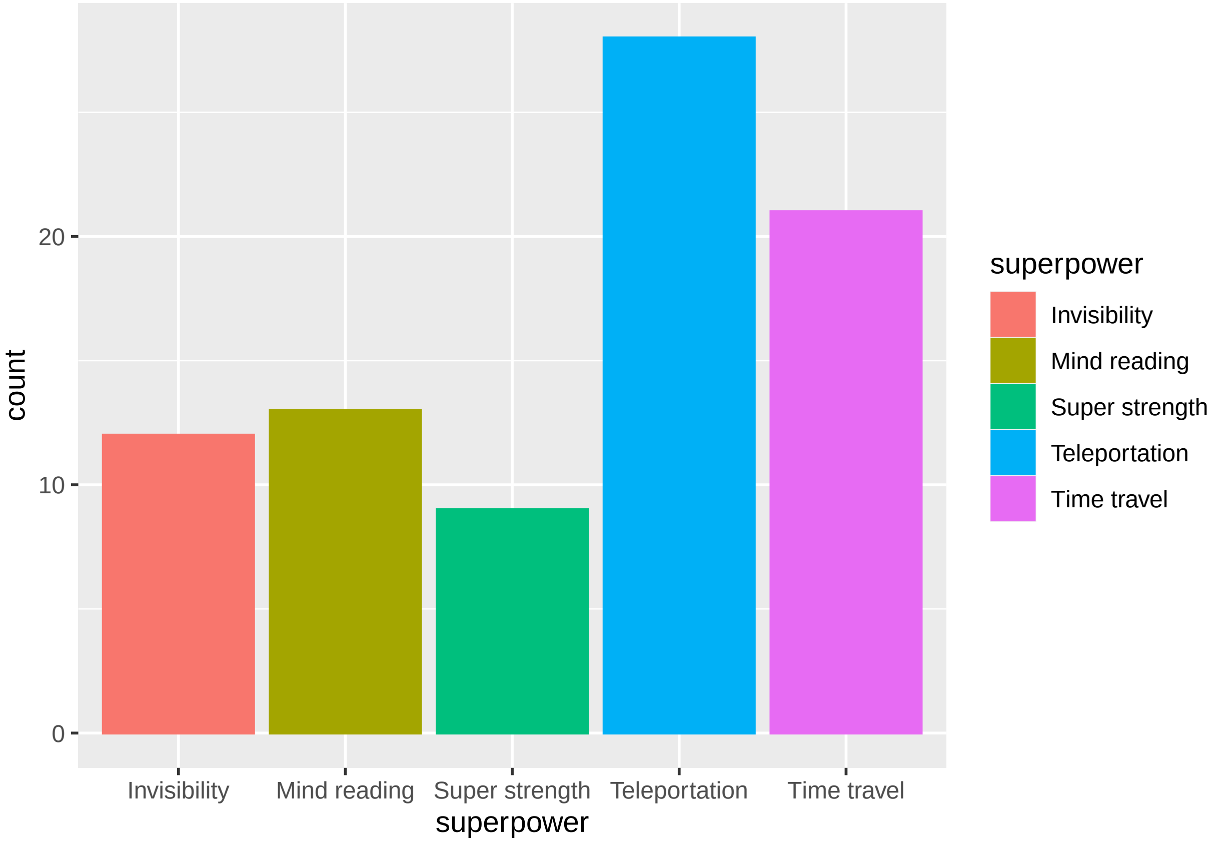
ggplot is based on the "grammar of graphics." A regular graph would normally need 3 main components:

1. A dataset
2. A coordinate system: describes how data coordinates are mapped to the plane of the graphic. Provides axis and grid lines to help read the graph. [the cartesian coordinate system is the default one]
3. Geoms: this refers to what you actually see in the plot such as points, lines, polygons, bars, etc.

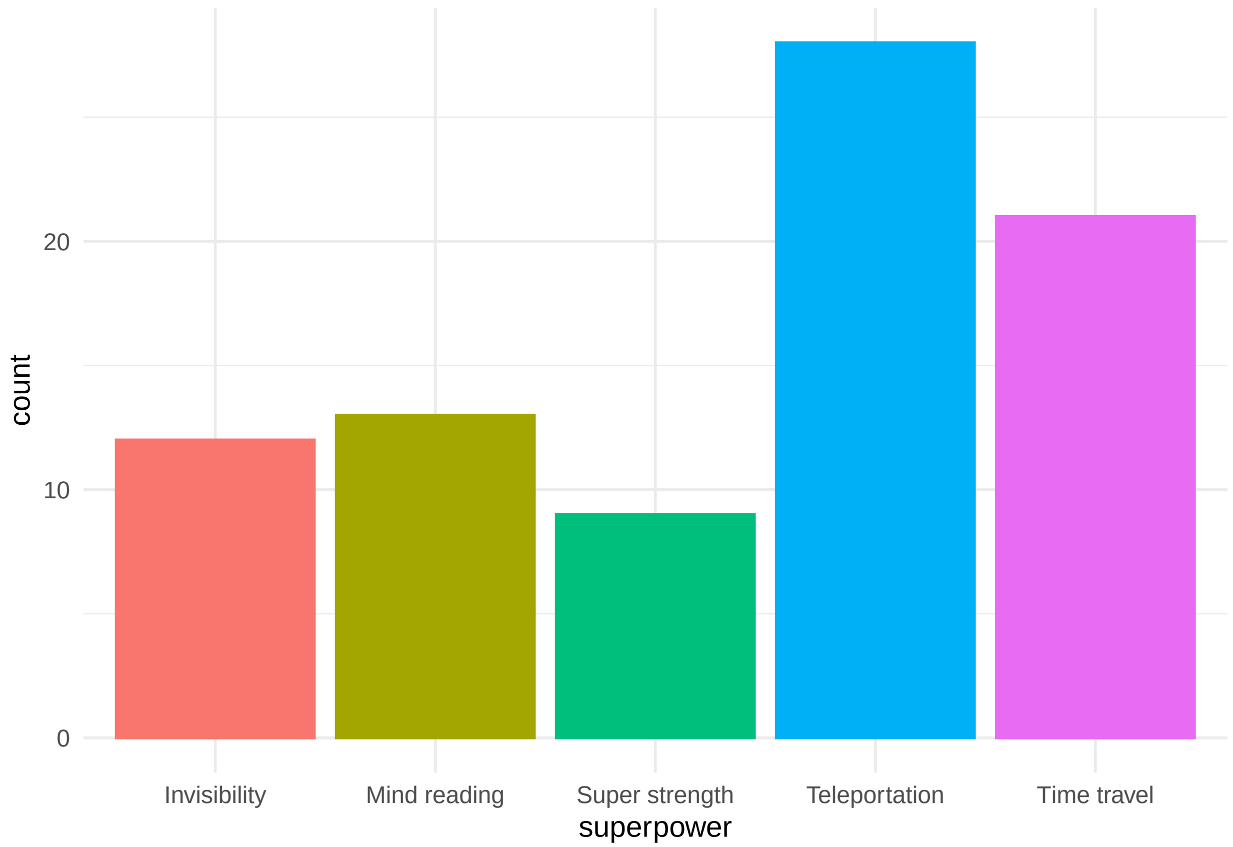
this is usually the code for it:

```r
ggplot(data, aes(x = height, y = weight, colour = age)) +
  geom_point()
```

Let's try our own graph for superpowers: Building a histogram

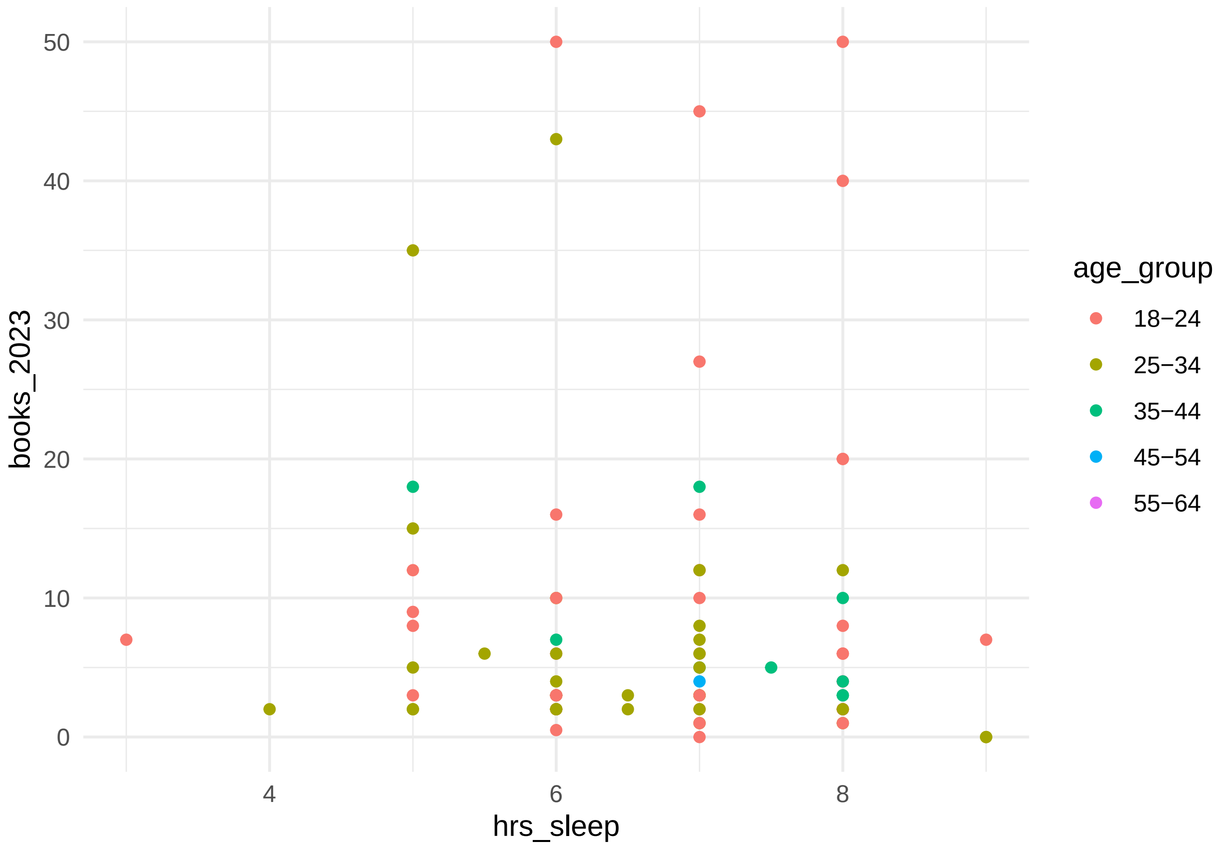


We could clean it a little bit

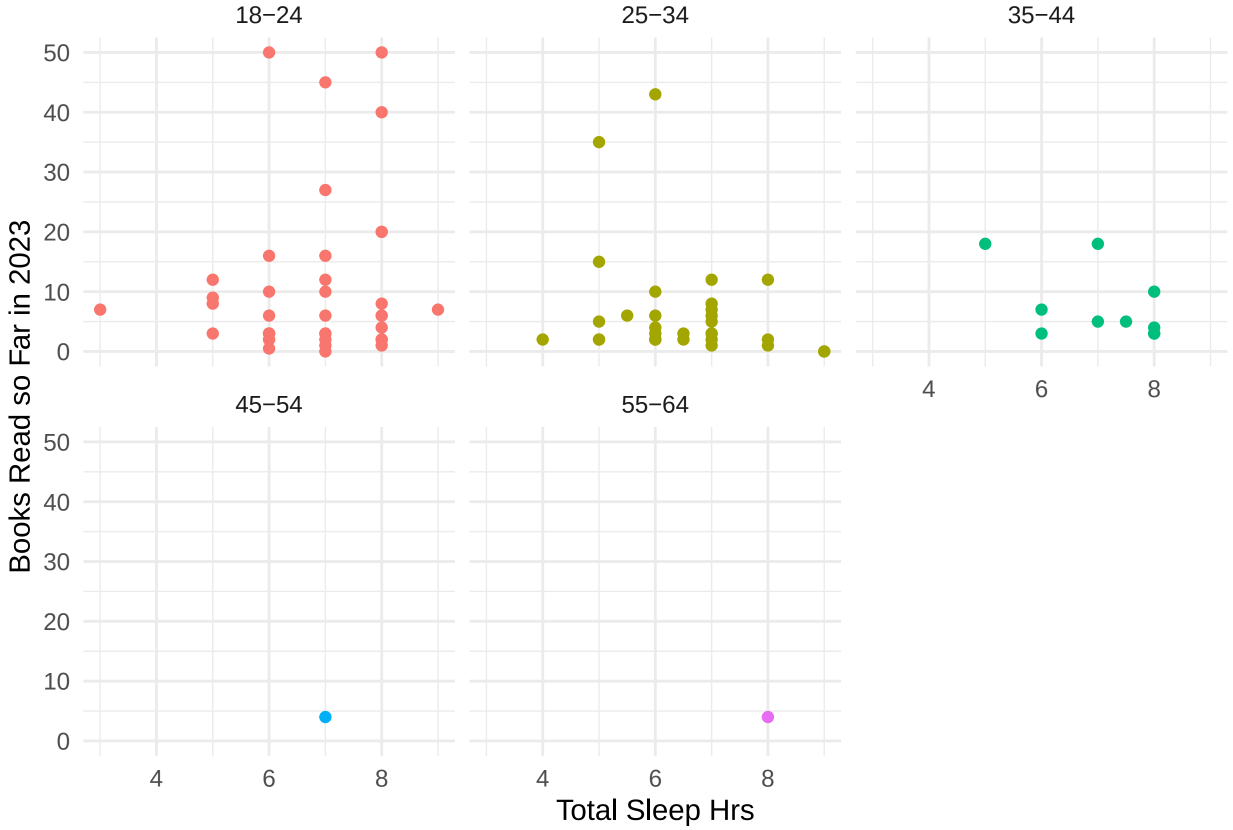


Let's visualize 2 continuous variables and then separate them by the age_group

Scatterplot for 2 continuous variables (sleep + books)



Now by age groups



After you clean your variables, you can export the new df as a csv or other type

```
write_csv(demo_data, "data/example_to_export.csv")
```