Problem Set 4

Due Monday October 2, 2023, 10am

To successfully complete this problem set, please follow these steps:

1. Download the contents of the assignment Dropbox link in a folder (directory) on your computer designated for this problem set only. Follow the directory structure, e.g., putting the `data` folder containing the datasets as a subdirectory of the project directory.

2. Insert your answers in the yellow boxes using Microsoft Word, and prepare a single `.R` script for what you produce. Save the word document as a `.PDF`.

3. Please submit the PDF to the designated `PS-XX: pdf` link and your R Script to the `PS-XX: R` link.

(1) Your name:

Nikhilla Bhuvana Sundar

(2) Group members, if any:

(3) Compliance with the Academic Code on problem set[1] (sign with an X below)

X

[1] You may use the same code from classmates, Ed Discussion Board, instructors, and generative AI. However, you must hand in your own unique written work and code in all cases. Any copy/paste of another's work is plagiarism. In other words, you can work with your classmate(s), sitting side-by-side and going through the problem set question-by-question, or use generative AI to provide potential code for you, but you must each type your own answers and your own code.

This problem works with two datasets. Autograder will have the following packages loaded:

```
tidyverse
haven
broom
fixest
fs
```

Problem 1

This problem equips you with powerful tools for *iteration*: i.e., simplifying repetitive tasks. The R4DS chapter https://r4ds.hadley.nz/iteration serves as a reference.

1.1

In a twist, the dataset this time comes in multiple files, one for each country:

```
"data/banerjee/Ethiopia.dta"
"data/banerjee/Ghana.dta"
"data/banerjee/Honduras.dta"
"data/banerjee/India.dta"
"data/banerjee/Pakistan.dta"
"data/banerjee/Peru.dta"
```

Your task in this problem is to create a single tibble/dataframe object that has *stacked* all of them and adds a column called `country` that lists the country it comes from.

Manually, one *could* write six lines of `read_dta`, add a `country` variable to each, and stack them with `bind_rows()`. But, that is repetitive, and also will be unfeasible when there are 100s of files. So instead, please achieve the same objective through `map`, the main function in the `purrr` package.

- The `map` function asks for an argument `.f` that is a *function*, and an argument `.x` for a vector of values. In this case, the vector of values is the vector of filenames to read from. `map` will repeat running the function `.f` for each value of the vector, and output a **list** for each of the outputs. See appendix for what we mean by a *named* vector. Vectors need not be named but it is useful to have one, especially here.

- To distinguish which rows of the stack came from which value of `.x`, The `list_rbind` function also takes an optional argument `names_to`, which is a character for the *name* of the variable to store the values of list it takes. See appendix for tips and further explanations.

  (no answer required in this square but make sure it is provided in your R file)

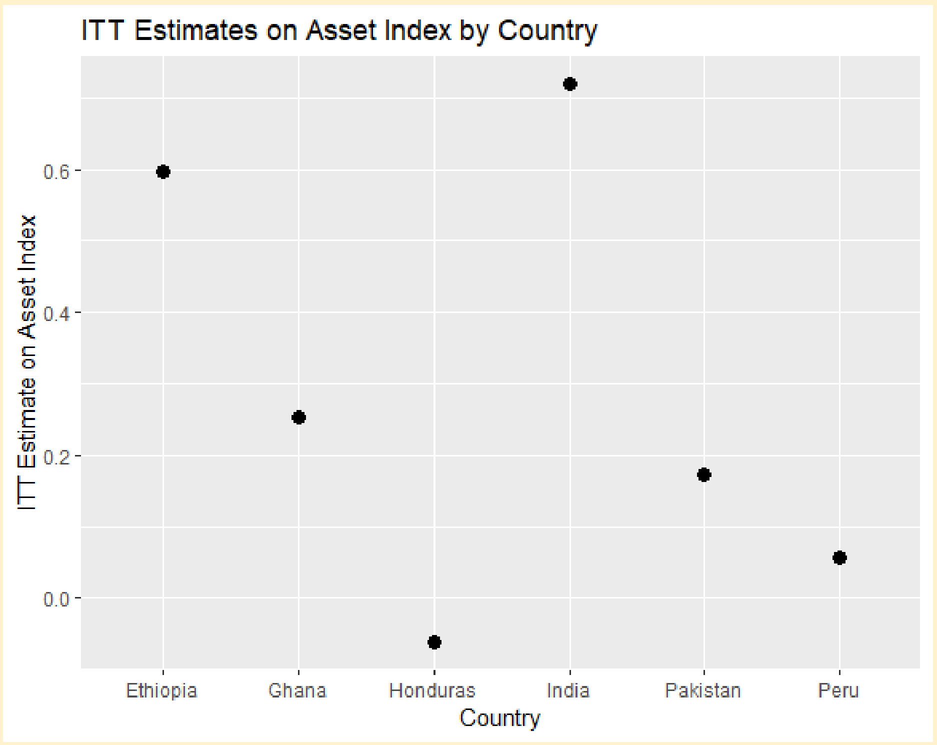
1.2

Next, we will ask you to define your own function instead of using one like `read_dta`. Being able to write your own functions vastly expands what you can do with datasets (and how quickly you can do it).

The objective: Estimate the treatment effect of assignment (the ITT) on a household's asset index at endline 2 (`asset_index_fup`), with the food security index at baseline (`asset_index_bsl`) as an additional control variable. However, do this for each country, so there will be six estimates from six regressions.

- You will still use `map()` and `list_rbind()` but now the input of `map()` is a vector of country values that you will use to subset (`filter`) the combined dataset.
- A useful new function to use *within* your custom function is `broom::tidy()`, which transforms regression objects into a tidy tibble where each row is a coefficient. This function is useful because `list_rbind` can only stack tibbles and dataframes, not the regression object itself.

Then, in the yellow box, present a small dot plot that summarizes the point estimate of the ITT on the asset index for each country. The dot plot should have six points with well-labeled axes.



Problem 2

This problem asks you to implement a instrumental variable (IV) regression to estimate treatment effects on the treated. PS-05 will ask you to run IV for our third paper (Miguel et al.). In the problem sets and section we will use to do IV with the function `fixest::feols` , although there are a few other packages with different syntax that can do IV. A screencast of using `feols` and another package: https://vimeo.com/755382687

2.1

Banerjee et al. note that of the six countries they study, *"[t]he India site was the only site in which some individuals refused participation,"* and that their estimates in their paper target the ITT (the intent to treat effect). In 2-4 brief sentences, explain when a policymaker would want to know the *ITT* of an intervention and when they would want to know the *TOT*. Your explanation should convey what ITT and TOT measure in this context in a way that is understandable to an educated audience not versed in statistics.

Given that ITT measures the outcomes of interest – asset index value of households based on initial assignment to the graduation program irrespective of whether they take it up or not, a policy maker would want to know ITT of this intervention to get a cost-benefit analysis of this treatment.

On the other hand, ToT focuses on outcomes of only those households that selected into the program. It will provide policymakers insights on the effectiveness of the graduation program itself along with quality of program delivery when rolled out although only among households that participated in the program.

2.2

For this problem I have created dataset, `pooled_hh_noncompliance.dta`. only for India that has a fake variable called `uptake` . This variable tries to recreate the noncompliance described in the paper. A value of 1 in this variable means that the household entered into the Graduation program. A value of 0 indicates that they did not. Use the dataset, \`pooled_hh_noncompliance.dta\`.

Report the rounded sample mean of the Household Asset Index *at baseline* among the following 4 groups:

1. the entire control group: 0.01
2. the entire treatment group: -0.05
3. the subset of the treatment group who selected into the treatment: -0.01
4. the subset of the treatment group that did not select into the treatment: -0.15

What do your numbers suggest about whether treatment assignment and uptake are randomly assigned?

By looking at the sample means of the entire control group and treatment we can say that the treatment assignment in randomly assigned. Due to randomization, the baseline characteristics of household asset index are almost same.

However, the mean figures of uptake subset groups suggest that there is an absolute difference of 0.14. This implies that the selection into the treatment is not random. Characteristics about the participants could have caused a difference in their uptake

2.3

In code, compute the *reduced form* effect and the *first stage* effect when this estimation is understood as an instrumental variables design where the instrument is treatment assignment.

Keeping in mind how the paper has standardized their outcome variables, interpret the two effects substantively in proper units. Your answer should make clarify *what* the effects are for, i.e., . "____ increases/decrease ____ by ____ [appropriate units here]."

1st stage: When households in India are randomly assigned to the graduation program, the chances of them selecting into the program increases by 68.38 percentage points.

Reduced form: Household asset index at endline is 0.22 standard deviations away from the mean for those households in India that received the graduation program when compared to the control group

2.4

Now use instrumental variables to estimate the TOT, i.e. the causal effect of taking up the Graduation treatment. It should match with what the two earlier estimates imply about the TOT.

Yes, using the instrumental variables to estimate TOT matches with the two earlier estimates giving a co-efficient of 0.32

Problem 3 (please note there is a 4th “problem” this time)

For the class on the day this problem set is due, we will discuss the following paper:

> Edward Miguel, Shanker Satyanath, and Ernest Sergenti (2004). “Economic Shocks and Civil Conflict: An Instrumental Variables Approach”, *Journal of Political Economy*.

For class, please coming having made an attempt at reading the paper, although I do not expect you to read it as fully or as critically as the past two papers because we have not yet covered instrumental variables fully. I generally recommend skimming in the following order: abstract, first paragraph or two of the introduction, *equations*, *tables and figures*, then the rest of the paper.

3.1

The Stata dataset `miguel_africa.dta` is the author’s dataset described in the paper. Each observation represents a country-year combination, and includes the following variables.

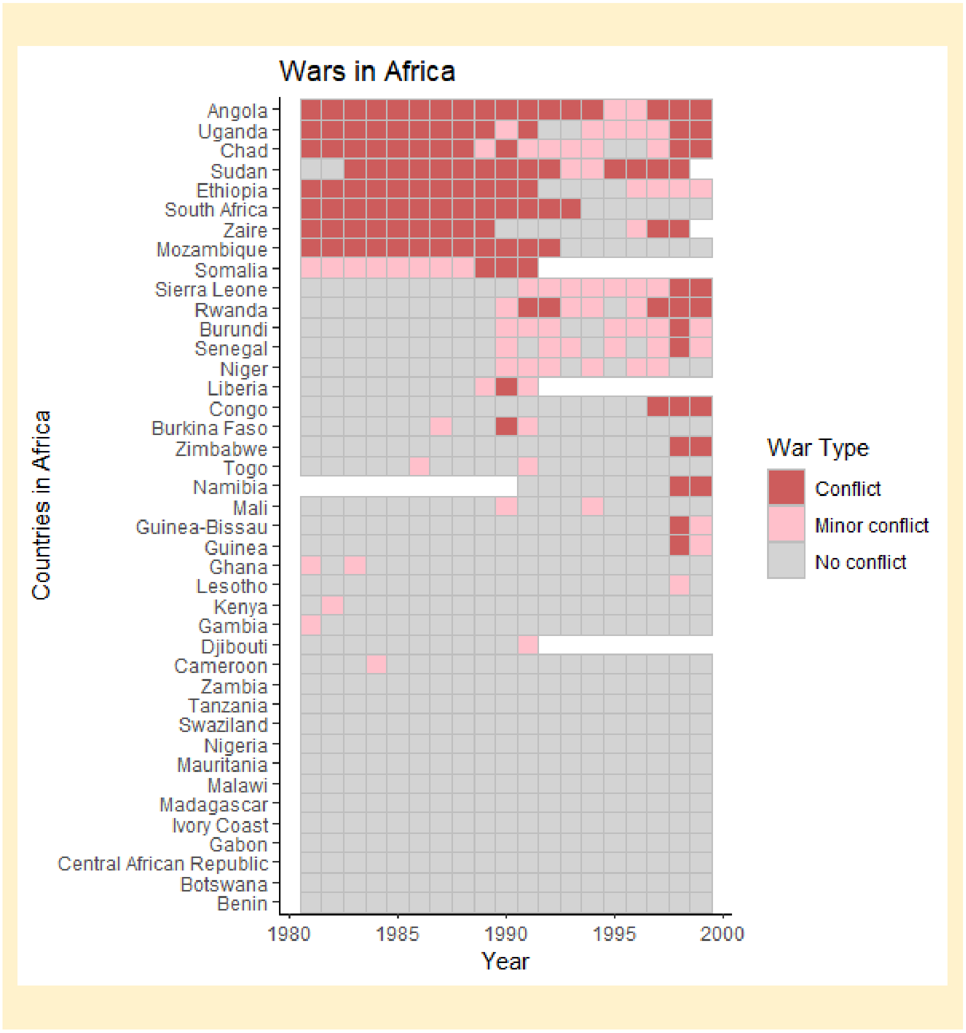
Variable name	**Description**
`year`	Year of measurement
`country_name`, `country_code`, `ccode`	Country (name, abbreviation, numerical code)
`minor_prio`	A binary variable for the occurrence of any conflict fitting PRIO’s definition of minor civil conflict (25 battle deaths or more)
`war_prio`	A binary variable for the occurrence of any civil conflict fitting PRIO’s definition (1000 battle deaths or more)
`any_prio`	The union of `minor_prio` and `any_prio`
`war`	A categorical variable indicating the combination of `minor_prio` (`Minor conflict`) or `war_prio` (`Conflict`), or none (`No conflict`)
`n_wars`	The number of conflicts the country has experienced in the dataset. A country will have the same value of this variable across all years.

Report how many (1) unique years, (2) unique countries , and (3) total rows there are in the data.

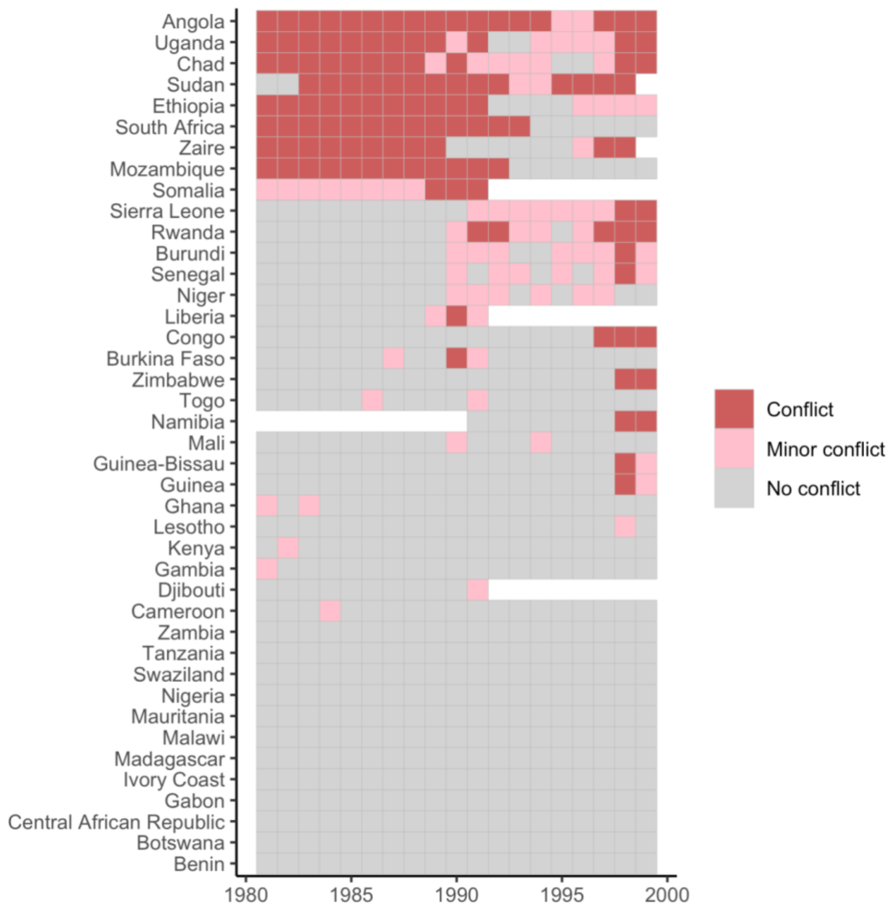
> Unique years: 19
>
> Unique countries: 41
>
> Total rows in the data: 743

3.2

Exploring a paper's dataset is often a good way to understand the paper better. Using `geom_tile()`, create a well-labeled[2] graph like the following that shows the pattern of the outcome variable `war` by year and country. Try reordering the countries by the number of conflicts it has experienced, not alphabetically.



[2] Throughout this class, any time we ask for a graph in a problem set or final paper we always want it to be well-labeled so a reader can understand roughly what is being shown. For example, if there is a color legend, it should indicate what the colors indicate. Axes should have titles, with the exception that you do not need one when it is obvious from the context. In the problem here, it is rather obvious that the axes represent countries and years, so I have omitted the axis titles, following advice in https://clauswilke.com/dataviz/figure-titles-captions.html#axis-and-legend-titles.



This paper will try to explain whether economic downturn causes the sort of civil conflict visualized here.

Problem 4

Please provide feedback about the course through this form:

https://yalesurvey.ca1.qualtrics.com/jfe/form/SV_9oC5ZIzaIb3K2J8

The survey is **anonymous**, and it is important that *everyone participate so the feedback we get is representative*. Please respond with a X or whatever signature here once you have completed the survey.

X

Appendix

Problem 1

1.1

Vectors

- Vectors were defined in the first primer: https://posit.cloud/learn/primers/1.2.
- A unnamed vector, for example, is: `c(1, 2, 3)`. A named vector is `c(Ethiopia = 1, Ghana = 2, Honduras = 3)`. In both, the values of the vector are the same (1, 2, 3) but the named vector associates each value with a label that is often useful information to keep track of.

purrr:map()

- The output of `map()` is a list. Remember from PS-02 and the primer that the *list* is a object in R that is quite flexible: we can work with a list of regression objects, or a list of other objects and values. For more restrictive output, `purrr` provides `map_vec` when the output of all values will be a length-one object like a number.

Extracting file names

- The input vector in this dataset should be the file names. You can type these in manually, e.g. `filenames2 <- c(Ethiopia = "data/banerjee/Ethiopia.dta", Ghana = "data/banerjee/Ghana.dta")` for two files, and we will accept that answer here. A faster and more robust way to extract this is through a function for getting the filenames of files in a particular file. `fs::dir_ls()` will list (ls) the items in a directory or folder (dir). So `filenames <- dir_ls("data/banerjee")` will extract the filenames of interest.

- The output of `dir_ls()` is unnamed. The function `names()` extracts the names of a vector, and new names can be created, e.g. `names(filenames2) <- c("Ethiopia", "Ghana")`. If you had more countries than you can manually type, you can also take the values of the vector (the filenames) and extract the names from the file character. `stringr` is the R package for such character manipulations (characters are called "strings" in computer science). For example, you can try running

```
c("data/banerjee/Ethiopia.dta",  "data/banerjee/Ghana.dta") |>
  str_remove_all("data/banerjee")

[1] "/Ethiopia.dta" "/Ghana.dta"
```

wil remove the string "data/banerjee" from the input string. You can also give it more complicated strings, such as "extract the sequence of non-period characters that come after"data/banerjee". That expression is:

```
c("data/banerjee/Ethiopia.dta",  "data/banerjee/Ghana.dta") |>
  str_extract("(?<=data/banerjee/)[^\\.]+")

[1] "Ethiopia" "Ghana"
```

the particular syntax here is cumbersome (`()`, `[]`, `^`, `?<=` are special characters with different meanings), but the good news is that this syntax is universal beyond R, and it is called regex. A list of all regex patterns in stringr is https://github.com/rstudio/cheatsheets/blob/main/strings.pdf. I also find ChatGPT explains and proposes regex generally well, though it sometimes makes mistakes.

1.2

The base-R way to define a function is: `function(x) { }`, where `function()` is a function to create a function, `x` indicates the argument(s) of the function, and the multi-line content that goes into `{ }` indicates the steps to implement within the function.[3] The inside of `{ }` is called the body. The proper code style for defining a function is,

```
add_two <- function(x) {
  x + 2
  }
```

where `x` is an argument, and the body of the function starts with the arguments `x`. In this example, the function adds a number to the input `x` and return a number with two added. Just like other R functions, the functions can take many different arguments. For example, if the Banerjee et al. household data is called `banerjee`, then the function

```
fun <- function(x, data) {
 data |>
  mutate(log_c_bsl = log(ctotal_pcmonth_bsl + x))
}
```

can be used as `fun(x = 1, data = banerjee)` to specify that the function should take the Banerjee et al. household data, add `1` to the per capita consumption, take it's log, and return the output. However, in our use of `map`, we can only *vary* one argument at a time. To specify a default for the second argument, we denote it in the definition of the argument using an equals:

```
fun <- function(x, data = banerjee) {
 data |>
  mutate(log_c_bsl = log(ctotal_pcmonth_bsl + x))
}
```

This takes function will default to using `banerjee` as its `data` argument unless otherwise specified, so `fun(x = 1)` will work fine even though the second argument is not specified. Of course, the object `banerjee` has to exist somewhere prior in your environment, by your creation.

More detail about functions can be found in R4DS Chapter 26 (https://r4ds.hadley.nz/functions.html).

Alternative strategies for Problem 1:

- A more general way to repeat a common procedure is to use a for loop. We can create an empty list with `list()` to store results, and then write a for loop that goes through each country. This operation takes the form

[3] Other alternative ways of defining a function in R are through the shorthand `\(x) {}` in recent versions of R

```
for (x in countries) {

}
```

where `countries` is a vector to iterate over. The subsetting and regression estimate would occur in the body. `for`-loops are not functions but you can see they use similar style. This procedure is also acceptable, but we chose not to use it as the primary example because involves the extra step of saving an object into a container. The `map*` functions from `purrr` can be thought of as an abbreviated way to do a for-loop with only one index.

- There is an even shorter way to solve this problem, which is to use `group_by` and `summarize` in the following way: `data |> group_by(country) |> summarize(tidy(lm(Y ~ treat, data = cur_data())))`. But this is an advanced use of function arguments. The use of `map_dfr` or for loops is widespread enough that we ask you to practice it here.

Problem 2

- We will use the `fixest` package in class and section, because we will reuse it in other contexts. Other packages like `AER::ivreg` or `estimatr::iv_robust` will also give you the same answer and are generally good packages, and although we will not be explicitly teaching them, it is fine to use these if you wish.

- `summary()` will print a regression table, whereas `coef()` will provide a *vector* of the coefficient estimates. The latter is more useful in 2.3. Vectors can be subset by square brackets. For example, if you run the regression `fit <- lm(Y ~ X1 + X2 + X3, dat)` on a dataset that has the columns `"Y"`, `"X1"`, `"X2"`, and `"X3"`, then `coef(fit)` will be a vector of length 4, for the intercept and then the covariates in the order within the (not the outcome). Each element will be named (see appendix to problem 1), and vectors can be subset by their location or their names (if they have names). So for example and `coef(fit)["X1"]` will give the coefficient estimate on `"X1"`, and `coef(fit)[2]` will give the same thing. Subsetting by names is often better practice because it is clear to the reader of the code what you are extracting. To extract multiple values, you can use the concatenation function `c()` to make your own vector: e.g., `coef(fit)[c("X1", "X2", "X3")]` or `coef(fit)[c(2, 3, 4)]`. A shorthand for sequential integers like `c(2, 3, 4)` is `2:4`, so `coef(fit)[2:4]` will do the same thing and is shorter.

- A demonstration of using instrumental variables is at https://vimeo.com/manage/videos/755382687

Problem 3

- There are many ways to do 3.1:
  - Using `dplyr` tools that we've already used, we can apply `count()` by country or by year and see how many rows it produces. The function `nrow()` will return the number of rows of a tibble.
  - `dplyr::distinct` works similarly as `count` but only generates of the distinct levels.

  - `unique()` is a base R function that will take a vector and condense it to its unique elements; `length()` is similar to `nrow()` but takes the number of elements in a vector. So, combining these will also give you the correct answer. In order to turn a column into a vector, use `dplyr::pull()`, as in `dat |> pull(country)`. Then this can be piped into something like `unique`, e.g., `dat |> pull(country) |> unique() |> length()`.
  - A base-R shorthand for `dat |> pull(country)` is `dat$country`. `$` is a special character in R that you can think of as "slot" since it looks like an "S" (take the slot called `country` in dataset `dat`). Although we do not recommend mixing base R and tidyverse in a single script, for very snippets like this it is acceptable.
- `geom_tile()` takes familiar aesthetics `x` and `y` for the location of its tiles. Use `fill` for the colors inside the tile and `color` for the color of its outlines.
- Since the data is `dta`, the variable `war` is a labelled variable (1, 2, 3, with labels). So that ggplot does not treat the variable as a continuous number, you should change it to a factor like you did in PS-03.
- To reorder countries by a variable, remember `fct_reorder` we briefly saw in the visualization lecture.
- Intuitive color legends greatly enhance the clarity of a plot. In the example shown, I've used `c(Conflict = "indianred", "Minor conflict" =  "pink", "No conflict" = "lightgray")` with `theme_classic()` as the theme, although you do not need to use these colors.
- The *title* of a legend is controlled similarly to the title of a axis, so `labs(x = "x variable", fill = "War Type")` will title the fill legend `"War Type"`. To show no legend at all (e.g. when it is self-explanatory), set the value to `NULL`.