Lab 03 - Nobel laureates

Moriah Ruggerio

In January 2017, Buzzfeed published an article on why Nobel laureates show immigration is so important for American science. You can read the article here. In the article they show that while most living Nobel laureates in the sciences are based in the US, many of them were born in other countries. This is one reason why scientific leaders say that immigration is vital for progress. In this lab we will work with the data from this article to recreate some of their visualizations as well as explore new questions.

Learning goals

- Collaborating on GitHub and resolving merge conflicts
- Replicating published results
- Data wrangling and visualisation

Lab prep

You have two tasks you should complete before the lab:

- **Task 1:** Read the Buzzfeed article titled *These Nobel Prize Winners Show Why Immigration Is So Important For American Science.* We will replicate this analysis in the workshop so it's crucial that you're familiar with it ahead of time.
- **Task 2:** Read about merge conflicts below. The merge conflict exercise we'll start with during the lab will assume that you have this background information.

Merges and merge conflicts

This is the second week you're working in teams, so we're going to make things a little more interesting and let all of you make changes and push those changes to your team repository. Sometimes things will go swimmingly, and sometimes you'll run into merge conflicts. So our first task today is to walk you through a merge conflict!

- Pushing to a repo replaces the code on GitHub with the code you have on your computer.
- If a collaborator has made a change to your repo on GitHub that you haven't incorporated into your local work, GitHub will stop you from pushing to the repo because this could overwrite your collaborator's work!
- So you need to explicitly "merge" your collaborator's work before you can push.
- If your and your collaborator's changes are in different files or in different parts of the same file, git merges the work for you automatically when you *pull*.
- If you both changed the same part of a file, git will produce a **merge conflict** because it doesn't know how which change you want to keep and which change you want to overwrite.

Git will put conflict markers in your code that look like:

```
<<<<<<< HEAD

See also: [dplyr documentation](https://dplyr.tidyverse.org/)
```

```
=======

See also [ggplot2 documentation](https://ggplot2.tidyverse.org/)

>>>>>>> some1alpha2numeric3string4
```

The `===`s separate *your* changes (top) from *their* changes (bottom).

Note that on top you see the word `HEAD`, which indicates that these are your changes.

And at the bottom you see `some1alpha2numeric3string4` (well, it probably looks more like `28e7b2ceb39972085a0860892062`

This is the **hash** (a unique identifier) of the commit your collaborator made with the conflicting change.

Your job is to *reconcile* the changes: edit the file so that it incorporates the best of both versions and delete the `<<<`, `===`, and `>>>` lines. Then you can stage and commit the result.

Merge conflict activity

Setup

- Clone the repo and open the .Rmd file.
- Assign the numbers 1, 2, 3, and 4 to each of the team members. If your team has fewer than 4 people, some people will need to have multiple numbers. If your team has more than 4 people, some people will need to share some numbers.

Let's cause a merge conflict!

Our goal is to see two different types of merges: first we'll see a type of merge that git can't figure out on its own how to do on its own (a **merge conflict**) and requires human intervention, then another type of where that git can figure out how to do without human intervention.

Doing this will require some tight choreography, so pay attention!

Take turns in completing the exercise, only one member at a time. **Others should just watch, not doing anything on their own projects (this includes not even pulling changes!)** until they are instructed to. If you feel like you won't be able to resist the urge to touch your computer when it's not your turn, we recommend putting your hands in your pockets or sitting on them!

Before starting: everyone should have the repo cloned and know which role number(s) they are.

Role 1:

- Change the team name to your actual team name.
- Knit, commit, push.

Make sure the previous role has finished before moving on to the next step.

Role 2:

- Change the team name to some other word.
- Knit, commit, push. You should get an error.
- Pull. Take a look at the document with the merge conflict.
- Clear the merge conflict by editing the document to choose the correct/preferred change.
- Knit.
- **Click the Stage checkbox** for all files in your Git tab. Make sure they all have check marks, not filled-in boxes.
- Commit and push.

Make sure the previous role has finished before moving on to the next step.

Role 3:

- Change the a label of the first code chunk
- Knit, commit, push. You should get an error.
- Pull. No merge conflicts should occur, but you should see a message about merging.
- Now push.

Make sure the previous role has finished before moving on to the next step.

Role 4:

- Change the label of the first code chunk to something other than previous role did.
- Knit, commit, push. You should get an error.
- Pull. Take a look at the document with the merge conflict. Clear the merge conflict by choosing the correct/preferred change. Commit, and push.

Make sure the previous role has finished before moving on to the next step.

Everyone: Pull, and observe the changes in your document.

Tips for collaborating via GitHub

- Always pull first before you start working.
- Resolve a merge conflict (commit and push) *before* continuing your work. Never do new work while resolving a merge conflict.
- Knit, commit, and push often to minimize merge conflicts and/or to make merge conflicts easier to resolve.
- If you find yourself in a situation that is difficult to resolve, ask questions ASAP. Don't let it linger and get bigger.

Getting started

Go to the course GitHub organization and locate your lab repo, which should be named `lab-03-nobel-laureates-YOUR_GITH`
Grab the URL of the repo, and clone it in RStudio. First, open the R Markdown document `lab-03.Rmd` and Knit it. Make sure it compiles without errors. The output will be in the file markdown `.md` file with the same name.

Warm up

Before we introduce the data, let's warm up with some simple exercises.

- Update the YAML, changing the author name to your name, and **knit** the document.
- Commit your changes with a meaningful commit message.
- Push your changes to GitHub.
- Go to your repo on GitHub and confirm that your changes are visible in your Rmd **and** md files. If anything is missing, commit and push again.

Packages

We'll use the **tidyverse** package for much of the data wrangling. This package is already installed for you. You can load them by running the following in your Console:

```
library(tidyverse)
```

Data

The dataset for this assignment can be found as a CSv (comma separated values) file in the `data` folder of your repository. You can read it in using the following.

```
nobel <- read_csv("data/nobel.csv")
```

The variable descriptions are as follows:

- `id`: ID number
- `firstname`: First name of laureate
- `surname`: Surname
- `year`: Year prize won
- `category`: Category of prize
- `affiliation`: Affiliation of laureate
- `city`: City of laureate in prize year
- `country`: Country of laureate in prize year
- `born_date`: Birth date of laureate
- `died_date`: Death date of laureate
- `gender`: Gender of laureate
- `born_city`: City where laureate was born
- `born_country`: Country where laureate was born
- `born_country_code`: Code of country where laureate was born
- `died_city`: City where laureate died
- `died_country`: Country where laureate died
- `died_country_code`: Code of country where laureate died
- `overall_motivation`: Overall motivation for recognition
- `share`: Number of other winners award is shared with
- `motivation`: Motivation for recognition

In a few cases the name of the city/country changed after laureate was given (e.g. in 1975 Bosnia and Herzegovina was called the Socialist Federative Republic of Yugoslavia). In these cases the variables below reflect a different name than their counterparts without the suffix '_original'.

- `born_country_original`: Original country where laureate was born
- `born_city_original`: Original city where laureate was born
- `died_country_original`: Original country where laureate died
- `died_city_original`: Original city where laureate died
- `city_original`: Original city where laureate lived at the time of winning the award
- `country_original`: Original country where laureate lived at the time of winning the award

Exercises

Take turns answering the exercises. Make sure each team member gets to commit to the repo by the time you submit your work. And make sure that the person taking the lead for an exercise is sharing their screen. You don't have to switch at each exercise, you can find your a cadence that works for your team and stick to it.

Get to know your data

1. How many observations and how many variables are in the dataset? Use inline code to answer this question. What does each row represent?

There are some observations in this dataset that we will exclude from our analysis to match the Buzzfeed results.

There are 935 rows and 26 columns in the the dataset. Each row represents a nobel winner.

2. Create a new data frame called `nobel_living` that filters for

- laureates for whom `country` is available

- laureates who are people as opposed to organizations (organizations are denoted with `"org"` as their `gender`)
- laureates who are still alive (their `died_date` is `NA`)

Confirm that once you have filtered for these characteristics you are left with a data frame with **228** observations, once again using inline code.

nobel_living has 228 observations.

Knit, *commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.*

Most living Nobel laureates were based in the US when they won their prizes

... says the Buzzfeed article. Let's see if that's true.

First, we'll create a new variable to identify whether the laureate was in the US when they won their prize. We'll use the `mutate()` function for this. The following pipeline mutates the `nobel_living` data frame by adding a new variable called `country_us`. We use an if statement to create this variable. The first argument in the `if_else()` function we're using to write this if statement is the condition we're testing for. If `country` is equal to `"USA"`, we set `country_us` to `"USA"`. If not, we set the `country_us` to `"Other"`.

```
Note that we can achieve the same result using the `fct_other()` function we've seen before (i.e. with
```

```
nobel_living <- nobel_living %>%
  mutate(
    country_us = if_else(country == "USA", "USA", "Other")
  )
```

Next, we will limit our analysis to only the following categories: Physics, Medicine, Chemistry, and Economics.

```
nobel_living_science <- nobel_living %>%
  filter(category %in% c("Physics", "Medicine", "Chemistry", "Economics"))
```

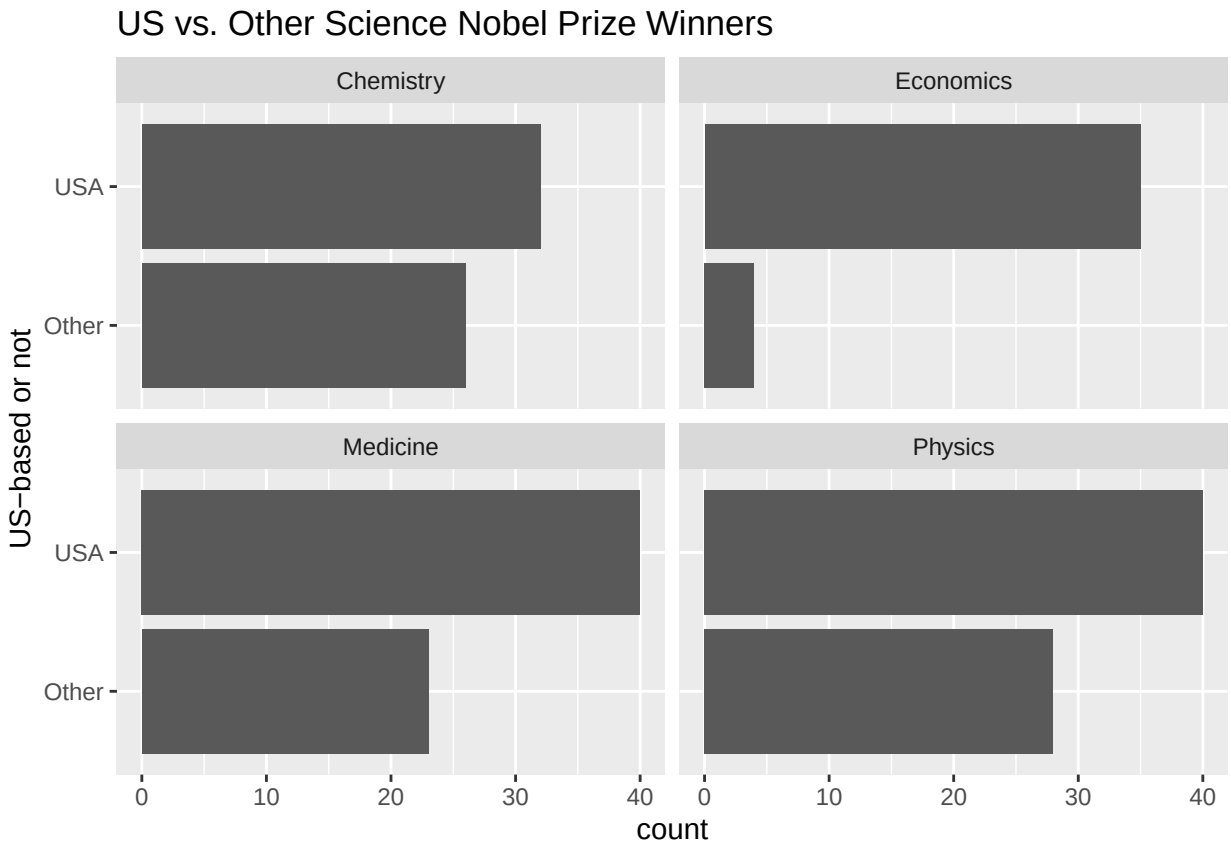
For the next exercise work with the `nobel_living_science` data frame you created above. This means you'll need to define this data frame in your R Markdown document, even though the next exercise doesn't explicitly ask you to do so.

3. Create a faceted bar plot visualizing the relationship between the category of prize and whether the laureate was in the US when they won the nobel prize. Interpret your visualization, and say a few words about whether the Buzzfeed headline is supported by the data.

    - Your visualization should be faceted by category.
    - For each facet you should have two bars, one for winners in the US and one for Other.
    - Flip the coordinates so the bars are horizontal, not vertical.

```
library(ggplot2)

ggplot(data = nobel_living_science,aes(y = country_us)) +
  geom_bar() +
  facet_wrap(~category) +
  labs(title = "US vs. Other Science Nobel Prize Winners", y = "US-based or not")
```



For each science category, there are more living nobel winners based in the US than not. This data supports the Buzzfeed headline.

Knit, *commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.d*

But of those US-based Nobel laureates, many were born in other countries

```
**Hint:** You should be able to ~~cheat~~ borrow from code you used earlier to create the `country_us` v
```

4. Create a new variable called `born_country_us` that has the value `"USA"` if the laureate is born in the US, and `"Other"` otherwise. How many of the winners are born in the US?

```
nobel_living_science <- nobel_living_science %>%
  mutate(
    born_country_us = if_else(born_country == "USA", "USA", "Other")
  )

nobel_living_science %>%
  group_by(born_country_us) %>%
  count()
```

```
## # A tibble: 2 x 2
## # Groups:   born_country_us [2]
##   born_country_us     n
##   <chr>           <int>
## 1 Other             123
## 2 USA               105
```

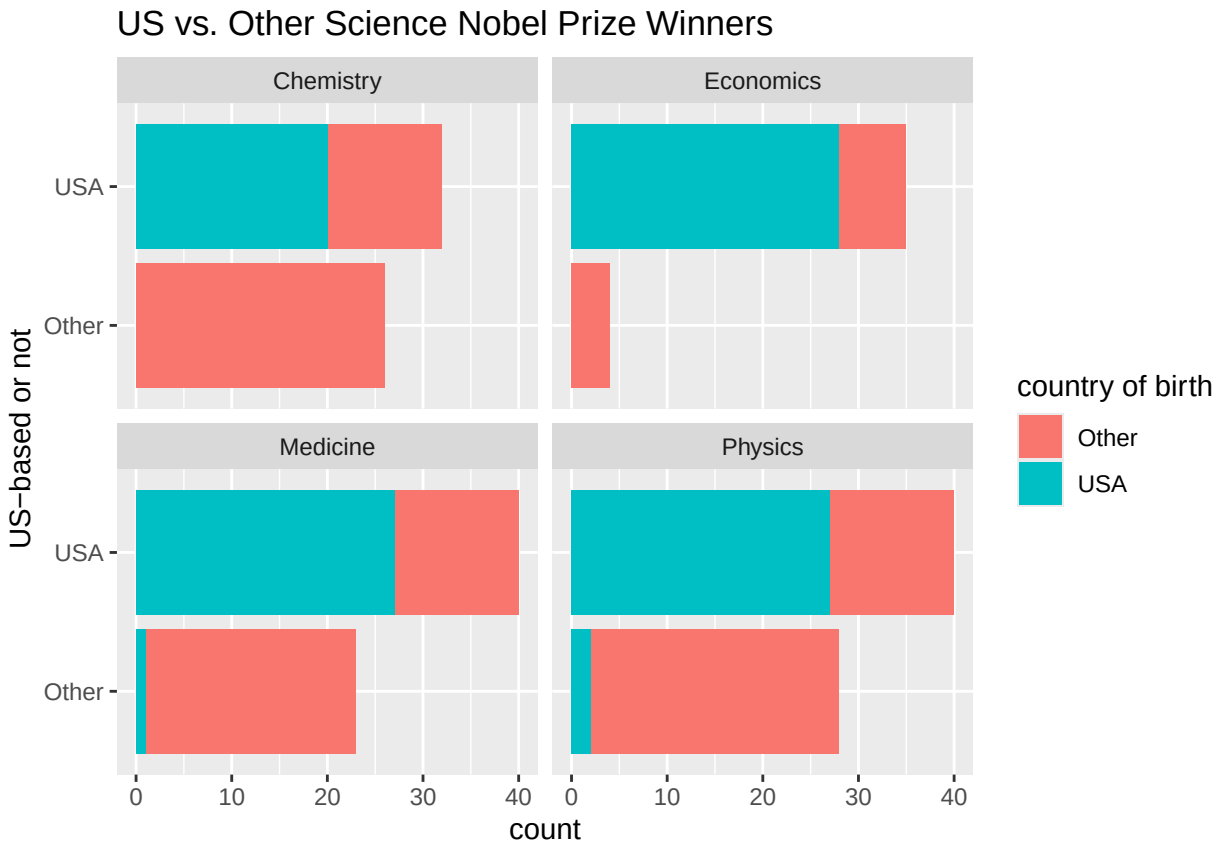
105 of the 228 total winners were born in the USA.

5. Add a second variable to your visualization from Exercise 3 based on whether the laureate was born in the US or not. Based on your visualization, do the data appear to support Buzzfeed's claim? Explain your reasoning in 1-2 sentences.

    - Your final visualization should contain a facet for each category.
    - Within each facet, there should be a bar for whether the laureate won the award in the US or not.
    - Each bar should have segments for whether the laureate was born in the US or not.

```
library(ggplot2)

ggplot(data = nobel_living_science,aes(y = country_us, fill = born_country_us)) +
  geom_bar() +
  facet_wrap(~category) +
  labs(title = "US vs. Other Science Nobel Prize Winners", y = "US-based or not", fill = "country of bi
```



Yes, the data appears to support Buzzfeed's claim that of the US-based Nobel winners many were born outside of the US. For all the categories except economics, at least a 1/3 of the winners were born outside of the US. Since, an exact amount was not specified, just "many," the claim holds.

Knit, *commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.*

Here's where those immigrant Nobelists were born

```
Note that your bar plot won't exactly match the one from the Buzzfeed article. This is likely because th
```

6. In a single pipeline, filter for laureates who won their prize in the US, but were born outside of the US, and then create a frequency table (with the `count()` function) for their birth country (`born_country`) and arrange the resulting data frame in descending order of number of observations for each country. Which country is the most common?

```
nobel_living_science %>%
  filter(born_country_us == "Other") %>%
  count(born_country) %>%
  arrange(desc(n))
```

```
## # A tibble: 33 x 2
##    born_country       n
##    <chr>          <int>
##  1 Germany           20
##  2 Japan             17
##  3 United Kingdom    16
##  4 France             8
##  5 Canada             6
##  6 China              6
##  7 Switzerland        6
##  8 Israel             5
##  9 Norway             4
## 10 Australia          3
## # i 23 more rows
```

Germany is the most common country of birth with 20 followed by Japan and the UK.

Knit, *commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards and review the md document on GitHub to make sure you're happy with the final state of your work.*

Now go back through your write up to make sure you've answered all questions and all of your R chunks are properly labelled. Once you decide as a team that you're done with this lab, all members of the team should pull the changes and knit the R Markdown document to confirm that they can reproduce the report.

Interested in how Buzzfeed made their visualizations?

The plots in the Buzzfeed article are called waffle plots. You can find the code used for making these plots in Buzzfeed's GitHub repo (yes, they have one!) here. You're not expected to recreate them as part of your assignment, but you're welcomed to do so for fun!