Lab 04 - Nobel laureates

2022-03-23

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 first time 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 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 `28e7b2ceb39972085a0860892062810fb812a08f`).

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

- One member of your group should serve as group leader and clone the repo https://github.com/musabu/lab4.git to their github account by importing the repo and making it private. [As a reminder, you go to your github account and click the plus (+) sign on top-right corner and 'Import repository.']
- Add your team members to the project as collaborators (Click settings from your repo page, then Manage Access) using their github usernames. They will receive emails with the request to be added to the project. Each team member should click on the link in the email they received and 'Accept' to become a collaborator in the project.
- Create and add a personal access token (PAT) to the repository on RStudio. As a reminder, to create a PAT, go to Github Settings, then Developer settings, then Personal Access Token, then Generate new token, and give the token a name, for instance, lab4.
- Share the token with your team members.
- To add the token to RStudio project, issue the following commands in console:
- install.packages("gitcreds")
- library(gitcreds)
- gitcreds_set(): It should prompt you for your access token. Paste the PAT created on Github earlier and hit ENTER. Note: if the system asked you what to do or to make a select from some options, type 2, and hit ENTER, and then paste the PAT, and ENTER.
- Note that you may sometimes be prompted for your username and password when you try to push changes even though you have added the PAT, just write your Github username and paste the PAT as the password.
- All members of your group should import the repo as a project in their RStudio cloud using the link https://github.com/GithubUsernameOfTeamLeader/lab4.git

- Add the PAT your received from the team leader using the above procedure.
- Assign the numbers 1, 2, 3, 4, and 5 to each of the team members.

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.

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

Role 1: Group Leader

- Open the R Markdown document `lab-04.Rmd` and Knit it.
- Change the team name to your actual team name (Give your team whatever name interests you). Write the names and IDs of all team members in the file.
- Knit, commit, push.

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

Role 2: second team member

- Change the team name to some other word.
- Knit, commit, push. You should get an error.
- Pull (not push this time; pull enable to download the changes make by other collaborators). 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: third team member

- 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: fourth team member

- 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.

Role 5: fifth team member

- 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.

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.

Packages

Open the file titled 'lab-04.Rmd' from the 'Files' window in the bottom-right of RStudio to do the exercises.

We'll use the **tidyverse** package for much of the data wrangling. It may not be installed in your project at the moment, and you might see a notification on top of the Rmd file that will ask you whether you want to install it. Click on 'install.' Alternatively, run the install code in your console.

Data

The dataset for this lab 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 code: 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. A team member that does not make any commit **will not get any marks from this lab exercise**

And make sure that the person taking the lead for an exercise is sharing their screen. You can create a group using MS Team (Download MS Team and login using your UHB username and password - https://www.microsoft.com/en/microsoft-teams/download-app) to enable you to work together.

Get to know your data

Role 2: second team member

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

Role 3: third team member

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`)

Role 4: fourth team member

3. Confirm that once you have filtered for these characteristics you are left with a data frame with 288 observations, once again using inline code.

Role 5: fifth team member

4. "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"`.

Role 1: Group Leader

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

Role 1: Group Leader

6. 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.

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!