Module 9: Seaborn, cont., and Intro to Git

October 27, 2025

Seaborn lets us convenient make plots and we can pass matplotlib `Axis` to target the location of the plots.

```
import matplotlib.pyplot as plt
import seaborn as sns


fig, ax = plt.subplots(1, 2)
sns.lineplot(x='year', y='lifeExp', data=df, ax=ax[0], linestyle=':')
sns.lineplot(x='year', y='lifeExp', data=df, ax=ax[1], linestyle='-.')
```

We can also use seaborn to separate out our data by using a grouping.

Plots in seaborn accept the argument `hue` which lets you separate your data by the grouping variable.

```
import matplotlib.pyplot as plt
import seaborn as sns


fig, ax = plt.subplots()
sns.lineplot(x='year', y='lifeExp', data=df, ax=ax, hue='continent')
```

Notice how the plot displayed a solid line with an error band around it.

seaborn does this with its default values for `estimator` and `errorbar`.

`estimator` is the aggregation function that is done when seaborn receives multiple observations. This happens when we provide a grouping with hue.

`errorbar` is set as confidence interval to the 95%.

These values are modifiable.

Let's do a side-by-side comparison of a mean and median representation of the data.

```
fig, axes = plt.subplots(1,2)
sns.lineplot(x='year', y='lifeExp', hue='continent', data=df, estimator='mean', ax=axes[0])
sns.lineplot(x='year', y='lifeExp', hue='continent', data=df, estimator='median', ax=axes[1])
```

Now let's do a side-by-side comparison contrast of the error bound display options.

```
fig, axes = plt.subplots(1,2)

sns.lineplot(x='year', y='lifeExp', hue='continent',data=df, estimator='mean', ax=axes[0], err_style='bars')
sns.lineplot(x='year', y='lifeExp', hue='continent', data=df, estimator='median', ax=axes[1], err_style='bands')
```

Exercise

Explore the gapminder dataset and create some new plots of the data.

- Update the x- and y- axes to have a more descriptive label.
- Try changing the position of the legend
- Create a figure with a two subplots vertically stacked
  - Select two continents to compare
  - Choose different linestyles
  - Choose different colors

Intro to Git

Git is a distributed version control software that lets you download, share, and collaborate on a project.

We will demonstrate how we can collaborate together with the lecture notes.

Installation

https://git-scm.com/install/mac
https://git-scm.com/install/windows

Github Account

We will need to create a Github account to submit our PRs. If you don't have a GH account, let's spend the next few minutes creating one.

Cloning a git repository

A common task you will do with git is clone a repository. Cloning a repository creates a copy of the repository on your local device.

First, let's go to our workspace folder for this class:

```
cd ~/workspace/mskcc-python
```

Now, let's clone our first repo:

```
git clone https://github.com/teonbrooks/mskcc-python-lecture-notes.git
```

Remotes

Remotes refer to the remote repository that we are working with. For the example at hand, we will be working with the default remote, which is known as `origin`.

```
git remote -v
```

This command with the argument `-v`, gives us a verbose message of the remotes we have.

It will show the name of the remote and where it is stored.

The `main` branch

When you first clone a repo, you will notice that there's usually a `main` branch that comes with your project.

We can think of a repository as a tree and the main branch as being the trunk.

For a repo, there's only one default branch, and this is usually designated as `main`. Historically, this branch was called `master`, and you might still see some projects called that, but new projects default to their main branch being called `main`.

Branches

Branches are used when you want to try something out on your own and you want to have your own copy of files and edits.

This is useful with you are making major edits to a project or if you want to try a new set of analyses but you aren't ready to share or commit them to the main project.

To create a new branch, you can use the following command:

```
git branch new_branch
```

Branches, cont.

You can switch to the new branch using the following command:

```
git checkout new_branch
```

You can simplify these two separate steps into one by using the following command:

```
git checkout -b new_branch
```

This creates a new branches and switches to that branch all in one step.

Contributing to a git repository

`git` allows for you to share content to a repository. Often, what you want to do is contribute your changes or `patches` to a project and have them reviewed and eventually merged into the project.

Let's create our first patch together.

```
git checkout -b yearbook-teon-brooks
```

Contributing to the Yearbook

Let's all add an entry to this class yearbook. We will all submit a markdown file with our name `firstname_lastname.md` and it will contain a markdown message in its body.

As a reminder, this is the full path for our local copy of the git repo:

```
cd ~/workspace/mskcc-python/mskcc-python-lecture-notes
```

First, let's install a new shortcut for our Terminal

Using the Command Palette (Command+Shift+P on macOS/Ctrl+Shift+P on Windows)

```
>Install 'code' command in PATH
```

We can use the command `touch` to create an empty file.
We can use the command `code` to launch our VS Code session from the terminal.

```
cd yearbook/2025
touch teon_brooks.md
code teon_brooks.md
```

Creating a commit (patch)

In terminal, we can stage the file we just created to git. This lets git know that a file was created and added to the project.

```
git add teon_brooks.md
```

To create a commit, we use the `commit` command and we add a message to go along with it:

```
git commit -m "Adding Teon Brooks to the yearbook"
```

Pushing our Commit

To share our commit with the project, we need to push our branch with its new commit.

We will use the branch that we created earlier, for me, that was `yearbook-teon-brooks`

```
git push origin yearbook-teon-brooks
```

Github

Let's see where all these branches were just pushed to:

https://github.com/teonbrooks/mskcc-python-lecture-notes

Pulling down updates

For the lecture today, I will create a commit and we will all sync our repos to accept the updated commits.

```
git pull origin main
```