Proceedings of the REU2022

Cybertraining

Gregor von Laszewski, laszewski@gmail.com

04 June, 2022

Contents

1 INTRODUCTION

1.1 Contribute

Before you do anything, check first if it is already in one of the books. Evaluate if it needs to be expanded in the book, or if a new section is needed. If a new section is needed, please consult with Gregor and ask for approval. We will then also decide if the chapter will be located in reu2022 or in books (both are repos).

Contribution will be determined based on review of commits, lines in such a fashion that the final lines willbe considered. For example let us assume you spent significant time on a section that ia a duplication of what others do, we will then delete that sections and you have not achieved a contribution.

So please make sure that you contribute valuable things.

Ask if you have an idea and want confirmation.

1.1.1 GitHub Insights

- Project: https://github.com/cybertraining-dsc/reu2022/projects/2
- Issues: https://github.com/cybertraining-dsc/reu2022/issues
- Pulse: https://github.com/cybertraining-dsc/reu2022/pulse
- Contributors: https://github.com/cybertraining-dsc/reu2022/graphs/contributors
- Traffic: https://github.com/cybertraining-dsc/reu2022/graphs/traffic

1.2 Lines contributed

```
+---------------------------------------+--------+
| author                                | count |
+---------------------------------------+--------+
| Alex Beck                             | 7874  |
| Jackson Miskill                       | 7803  |
| Gregor von Laszewski                  | 1166  |
| Alison Lu                             | 1151  |
| Diyaolu04                             | 247   |
| J.P                                   | 97    |
| Robert Knuuti                       | 46     |
| Binary file (standard input) matches | 36     |
| AbdulBaqiy Diyaolu                  | 25     |
| Paul2002feb                         | 17     |
| JunyangHe                           | 15     |
| al07734                             | 13     |
| Not Committed Yet                   | 9      |
| Jackson                             | 4      |
+---------------------------------------+-------+
```

1.3 REU 2022

1.3.1 Books

- https://cloudmesh-community.github.io/pub/vonLaszewski-python.pdf
- https://cloudmesh-community.github.io/pub/vonLaszewski-python.epub
- https://cloudmesh-community.github.io/pub/vonLaszewski-linux.pdf
- https://cloudmesh-community.github.io/pub/vonLaszewski-linux.epub
- https://cloudmesh.github.io/cloudmesh-mpi/report-mpi.pdf
- https://cloudmesh-community.github.io/pub/vonLaszewski-writing.pdf
- https://cloudmesh-community.github.io/pub/vonLaszewski-communicate.pdf
- https://cloudmesh-community.github.io/pub/vonLaszewski-reu2022.pdf

1.3.2 The repository for our report(s)

- https://github.com/cybertraining-dsc/reu2022

1.3.3 The many UVA technical help ticket systems

- ITS: https://virginia.edusupportcenter.com/shp/uva/helpcenter
- Rivanna: https://varesearchhelp.atlassian.net/servicedesk/customer/user/requests?page=1&reporter=all
- BII: https://uva-biocomplexity.atlassian.net/servicedesk/customer/user/requests?page=1&reporter=all&statuses=open&statuses=closed

1.3.4 Our technical slack

- https://cloudmesh-reu2022.slack.com

1.3.5 The administrative slack

- https://biocomplexity-eoo5671.slack.com/archives/C031CR0B2QG

1.3.6 Rivanna Information

- https://www.rc.virginia.edu/
- https://www.rc.virginia.edu/userinfo/rivanna/overview/
- access through ssh: https://www.rc.virginia.edu/userinfo/rivanna/login/
- access through Web Browser: https://shibidp.its.virginia.edu/idp/profile/SAML2/Redirect/SSO?execution=e2s1

1.3.7 GitBash (Windows only)

- https://git-scm.com/downloads

1.3.8 Pycharm

- pycharm community edition: https://www.jetbrains.com/pycharm/download
- md: https://www.jetbrains.com/help/pycharm/markdown.html
- (optional) extension makefile: https://plugins.jetbrains.com/plugin/9333-makefile-language
- (optional) rst: https://www.jetbrains.com/help/pycharm/restructured-text.html
- 80 char: https://intellij-support.jetbrains.com/hc/en-us/community/posts/206070859-How-do-I-enable-the-80-column-guideline-change
- background to white as i can not read white on black when in meetings with me I will not support anyone that has black background: https://www.jetbrains.com/help/pycharm/user-interface-themes.html use intelliJ Light. After the meeting you can set it to whatever.

1.3.9 Bibliography Management

- jabref: https://www.jabref.org/
- bibtex: https://en.wikipedia.org/wiki/BibTeX

- draft bibtex from urls: https://addons.mozilla.org/en-US/firefox/addon/bibitnow/ (has to be modified once copied. Not everything will work.

1.3.10 GitHub repo

- https://github.com/cybertraining-dsc/reu2022
- Sandra, JP: https://github.com/cloudmesh/cloudmesh-mpi

1.3.11 Only important for Gregor (do not use)

- request storage: https://www.rc.virginia.edu/form/storage/
- rivanna partition/account: https://www.rc.virginia.edu/userinfo/rivanna/allocations/
- managing groups: https://mygroups.virginia.edu/

1.4 Communications with C4GC Students

- There are weekly meetings with Ben Hurt every Tuesday and Wednesday from 11-11:45 am on Zoom and in room 4402.
- Around 4 students present each at the meetings. Every student presents twice and the link to the schedule can be found at this link.
  - first presentation is about the student's goals and the introduction to the project:
    * How you define success for this summer
    * Who you're working with
    * What your project is
    * What you've done so far
    * [optional] What challenges you've faced so far
  - It should be around 5-7 minutes long.
- In order to access the Biocomplexity Institute building, you must alert your mentor and sign up on the BII app. If coming for the first time you need to request access with Ben S.
- C4GC Links and Recordings
- Rivanna Orientation (C4GC specific) on Thursday June 9, 10 am - 11 am.

2 INSTALL

2.1 Install

In this section, we present an easy to follw instalation for a recent version of python and cloudmesh.

Before you start doing the install, please read the entire section and develop a plan for installation.

2.1.1 Windows

The instalation of cloudmesh benefits from using gitbash as it allows us to have a terminal that is similar to that of macOS and Linux.

Hence before we install python and cloudmesh we install gitbash.

Furthermore, we provide the option to use chocolatey to install packages in similar fashion as on linus.

2.1.1.1 Git Bash install To install gitbash, pleas download it forst from

- https://git-scm.com/downloads

Click `Download` for Windows. The download will commence. Please open the file once it is finished downloading.

Next you start the downloaded program and follw the install screens carfully.

TODO: what is a UAC?

- The UAC Prompt will appear. Click `Yes`. It will show you Git's license: a GNU General Public License. Read it and Click `Next`. To ensure security of the operating system a UAC prompt allows operating systems, particularly windows, to prompt for consent or credentials from local administrators before starting a program.
- Click `Next` to confirm that `C:\Program Files\Git` is the directory where you want Git to be installed.
- Select the box to create a shortcut icon on the desktop. Click `Next` to continue with the install.
- Click `Next` to accept the default text editor which is vim,
- TODO: THIS IS WRONG, you do want the one with main Click `Next` again to Let Git decide the default branch name
- Click `Next` again to run Git from the command line and 3rd party software,

- Click `Next` again to use the OpenSSL library
- Click `Next` again to checkout Windows-style,
- Click `Next` again to use MinTTY,
- Click `Next` again to use the default git pull,
- Click `Next` again to use the Git Credential Manager Core,
- Click `Next` again to enable file system caching, and then
- Click `Install` because we do not need experimental features.

A video tutorial on how to install Git and Git Bash on Windows 10 is located at https://youtu.be/HCotEx_xCfA

A written tutorial on how to install Git and Git Bash on Windows 10 is located at https://cybertraining-dsc.github.io/docs/tutorial/reu/github/git/

2.1.1.2 Python 3.10 install

To install python 3.10 please go to

- https://python.org

and download the latest version.

- Click `Download`. The download will commence. Please open the file once it is finished downloading
- Click the checkbox `Add Python 3.10 to PATH`
- Click `Install Now`
- At the end of the installation click the option to `Disable path length limit`

A video tutorial on how to install Professional PyCharm is located at https://youtu.be/QPESX-VBnEU

A video on how to configure PyCharm with cloudmesh is located at https://youtu.be/eb1IQBx0D50

2.1.1.3 Installing cloudmesh

Cloudmesh can be installed in any shell that has python and git access. HOwever it is convenient t use gitbash as it simplifies the documentation and allows us to interact with linux commands with the windows file system. The installation is done with a Pyython `venv` so you do not effect your computres python install. Here are the steps:

```
$ python -m venv ~/ENV3
$ source ~/ENV3/Scripts/activate
$ cd
```

```
# mkdir cm
$ cd cm
$ pip install pip -U
$ pip install cloudmesh-installer
$ cloudmesh-installer get cmd5
$ cms help
$ touch .bashrc
$ echo "source ~/ENV3/Scripts/activate" >> .bashrc
$ echo "cd ~/cm" >> .bashrc
```

To activate it, start new gitbash and terminate the first gitbash window. If you see in the new window `(ENV3)`, continue. Git bash will initialize the environment. It will set now up some envireonment and thus you will start again gitbash and terminate the second gitbash you created. Now gitbash is properly created and configures.

If you do not want to always start in the directory `cm` do replace the line in your `.bashrc` `cd cm` with `cd`

2.1.1.4 Uninstall

```
$ rm -f ~/ENV3
```

Edit the .bashrc and .bash_profile file and delete the lines

```
$ source ~/ENV3/Scripts/activate
$ cd cm
```

2.1.2 Choco install

There are a number of usefull packages that you can install via choco this includes visual code, pychram, emacs, and make

Even python could be installed with it however we have not tested the install of python via choco. However we tested the install of emacs, pychram, and make.

2.1.3 Install Chocolatey

To install pycharm, pleas esatrt a gitbash terminal as administrator: To do so press the `Windows` key and type powershell. Click Run as Administrator. Click Yes.

2. In PowerShell execute the following command:

   ```
   PS C:\Windows\system32> Set-ExecutionPolicy AllSigned
   ```

   Then type `y`.

3. Next type in the command (copy and paste to not make a mistake)

   ```
   PS C:\Windows\system32> Set-ExecutionPolicy Bypass -Scope Process
       ↪  -Force; [System.Net.ServicePointManager]::SecurityProtocol
       ↪  = [System.Net.ServicePointManager]::SecurityProtocol -bor
       ↪ 3072; iex ((New-Object System.Net.WebClient).DownloadString
       ↪ ('https://community.chocolatey.org/install.ps1'))
   ```

4. Wait for the installation to complete; once you see

```
PS C:\Windows\system32>
```

with a blinking cursor again, and lines have stopped appearing, then the Chocolatey installation has finished. Type `choco` and you should see Chocolatey in green text.

Now you can install many programs by launching PowerShell as Administrator or gitbash.

A list of programs that you can install with `choco` can be found at

- https://community.chocolatey.org/packages/

2.1.4 Installing Useful Developer Programs

The following useful developer programs can be installed. Select the once you like and install them with the appropriate command

```
$ choco install make -y
$ choco install emacs -y
$ choco install pycharm -y
$ choco install firefox -y
$ choco install vscode -y
$ choco install zoom -y
```

Once teh install completes PyCharm will be ready for you to use. You can install many programs this way, and the

2.1.5 Linux

2.1.5.1 Install Python 3.10.5

The instalatiton form source can be done easily as shown next

```
$ mkdir -p ~/tmp
$ cd ~/tmp
$ wget https://www.python.org/ftp/python/3.10.5/Python-3.10.5.tar.xz
$ tar xvf Python-3.10.5.tar.xz
$ cd Python-3.10.5/
$ ./configure --enable-optimizations
$ make -j $(nproc)
$ sudo make altinstall
$ pip install pip -U
$ python3.10 -V
```

2.1.5.2 Setting up the a venv

We assume you use bash

```
$ python3.10 -m venv ~/ENV3
$ source ~/ENV/bin/activate
$ cd
$ mkdir cm
$ cd cm
$ pip install cloudmesh-installer
$ cloudmesh-installer get cmd5
$ cms help
$ touch .bashrc
$ echo "source ~/ENV3/bin/activate" >> .bashrc
$ echo "cd cm" >> .bashrc
$ echo "source ~/ENV3/bin/activate" >> .bash_profile
$ echo "cd cm" >> .bash_profile
```

2.1.5.3 Uninstall

```
$ rm -f ~/ENV3
```

Edit the .zshrc and .zprofile file and delete the lines

```
$ source ~/ENV3/bin/activate
$ cd cm
```

2.1.5.4 Update

In case you need to update the Python version it is sufficient to follow the instructions provided in the section `Install Python 3.10.5`, while replacing the version number with the current python release number.

In case you need to create a new virtual ENV3. You can first uninstall it and then reinstall it.

An easy way to do all of this with a command is the following:

```
$ cd ~/cm
$ pip install cloudmesh-installer -U
$ cloudmesh-installer new ~/ENV3 cmd5 --python=/usr/local/bin/python3
   ↪ .10
$ source ~/ENV3/bin/activate
$ python -V
$ which python
```

2.1.6 macOS

We assume you use zsh which is the default on macOS

2.1.6.1 Cloudmesh

2.1.6.1.1 Install

Before any of the following, make sure to download the current version of python. At the time of this writing, it is python 3.10.5

Second, execute the following commands in your terminal. Make sure to do this in order.

```
$ cd
$ python3.10 -m venv ~/ENV3
$ source ~/ENV/bin/activate
$ mkdir cm
$ cd cm
$ pip install cloudmesh-installer
$ cloudmesh-installer get cmd5
$ cms help
$ echo "source ~/ENV3/bin/activate" >> .zshrc
$ echo "cd cm" >> .zshrc
$ echo "source ~/ENV3/bin/activate" >> .zprofile
$ echo "cd cm" >> .zprofile
```

It creates the virtual environment, a directory called `cm`, then installs `cloudmesh`. Following this, it sets the macOS startup commands `.zshrc` and `.zprofile` to start up in the virtual environment `ENV3`.

2.1.6.1.2 Uninstall

```
$ rm -rf ~/ENV3
```

You may need to enter your system password.

2.1.6.2 Updating Python

Before starting this process, ensure that python is in the correct path. Test in the terminal.

To do so remove the existing ENV3 first and start a new terminal in which you will be working.

```
$ rm -rf ~/ENV3
```

Start the new terminal and execute the commands to verify if you have the right updated version of python

```
$ which python3.10
$ where python3.10
$ python3.10 -V
```

Now execute

```
$ cd ~/cm
$ python3.10 -m venv ~/ENV3
$ source ~/ENV3/bin/activate
$ pip install cloudmesh-installer
$ cloudmesh-installer get cmd5
$ cms help
```

As `zsh` is already configured previously, we do not have to set it up again.

2.1.6.3 Homebrew install

Homebrew (`brew`) like `choco` is a package management software. Unlike choco `, it is used by macOS devices rather than Windows devices.` brew‘ is used to more easily download packages to an operating system; simply put, it eliminates the need for the user to search for and download the desired package.

Installing `brew` is simple.

- First, make sure the computer that is downloading Homebrew is up-to-date with the latest software for its OS.
- Second, ensure that `xcode` has been installed. `xcode` can be installed from the Apple App Store.
- Third, in the terminal, write out:

  ```
  $ xcode-select --install
  ```

  This installs `xcode` command line tools.
- Fourth, run the following command in the terminal:

  ```
  $ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/
     ↪ Homebrew/install/HEAD/install.sh)"
  ```
- Fifth, enter the administrator password into the desired location.
- It may take a moment for the software to install, but it will eventually say **"Installation successful!"** in the terminal. After that, Homebrew is installed onto the device.

After the user has correctly installed Homebrew, it is simple to install packages directly to the operating system:

```
$ brew install [package name]
```

2.2 Ramdisk

how to set up aand manage a ramdisk on lunux

develop cms program

cms ramdisk –… –size=SIZE

use humanize so we can us 1GB for size …

showcase

a) dynamic ramdisk no reboot needed, but if reboot, ramdisk needs to be set up new
b) ramdisk integrated in fstab with reboot
c) backu and load ramdisk

On macOS a RAM disk with 512MB space can be created with the following command:

```
n = 512 * 2048
os.system('diskutil eraseVolume HFS+ "RAMDisk" `hdiutil attach -
   ↪ nomount ram://{n}`')
```

2.2.1 Ubuntu

On Ubuntu, a RAM disk and its read-only shadow can be created by:

```
mount -t tmpfs -o size=512m tmpfs /mnt/ramdisk
mount -t aufs -o br:/mnt/ramdisk=ro none /mnt/readonly

# mkdir /tmp/ramdisk; chmod 777 /tmp/ramdisk
# mount -t tmpfs -o size=256M tmpfs /tmp/ramdisk/
```

using /dev/shm:

http://ubuntuguide.net/ubuntu-using-ramdisk-for-better-performance-and-fast-response

Various methods: (in german): https://wiki.ubuntuusers.de/RAM-Disk_erstellen/

ramfs is an older file system type and is replaced in mostly by tmpfs.

2.2.2 windows

https://forums.guru3d.com/threads/guide-using-imdisk-to-set-up-ram-disk-s-in-windows-with-no-limit-on-disk-size.356046/

3 GRAPH VIZUALIZATION

3.1 Python Data Management for Visualizations

In python, there are several ways that data can be interpreted in order to generate the graphics for data visualization.

Through several examples, we will show how to manage different types of data and how to best interact with them. They are lists, dictionaries and CSV files.

3.1.1 Lists

Lists are clumps of continuous values that are put directly next to each other in the computer memory system.

3.1.1.1 Construction In python, lists can be constructed like this:

```
example_list = ['example' , 2, 'b', 45, False]
```

Lists have order and can hold many types (bool, int, String, etc.) of values.

3.1.1.2 Accessing Values In python, it is easy to access values within a list. The following code provides an example of how to access certain values within a list:

```
index_4 = example_list[4]
```

It is important to note that lists have indices, which range from 0 to the length of the list - 1. The indices provide access to values within the list.

3.1.1.3 Updating Values To update a value within a list, simply utilize same brackets:

```
example_list[3] = 'bear'
```

This will change the value at the third index from 45 to bear.

3.1.1.4 Python Built-In Methods Python has several built-in methods for lists. These include `append()`, `clear()` `copy()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove(`, `reverse()`,and `sort()`.

A user can utilize these methods to make changes in the necessary ways to the list.

3.1.1.5 Examples

3.1.2 Dictionaries

Dictionaries are like specialized lists. They hold a key-value pair that allows for a user to look up a key and find the associated value. Dictionaries are useful for storing values in a way that is more organized than a linear list. Furthermore, dictionaries make it easy for users to look up the necessary information.

3.1.2.1 Construction In python, dictionaries are constructed as follows:

```
example_dictionary = {'motorcycles': 2, 'autocycles': 3, 'cars': 4, '
   ↪ small_trucks': 6
                      'large_trucks': '18'}
```

The string values are the keys, which provide access to the values within the dictionary. The colon provides the computer with the command for assigning key-value pair.

3.1.2.2 Accessing Values There are several built-in commands to access both the keys and values in a dictionary. They are as follows:

```
example_dictionary.get()
example_dicionary.keys()
example_dictionary.values()
```

The `.get()` method returns the value that is associated with the given key, the `.keys()` method returns a list of the keys alone, and the `.values()` method returns a list of the values alone. There are more methods (see the Python Built-In Methods section)

3.1.2.3 Updating Values To update the dictionary, there is one method that can be used:

```
example_dictionary.update({'motorcycles': 20})
```

This will update the value associated with this particular key-value pair.

3.1.2.4 Python Built-In Methods There are several built-in methods that allow for more dictionary manipulation: They are `clear()`, `copy()`, `fromkeys()`, `items()`, `pop()`, `popitem()`, and `setdefault()`.

3.1.2.5 Examples

3.1.3 CSV Files

CSV stands for “comma-separated-values” and is a data structure that is incredibly common for data management and analysis. There are many ways to access CSV files. The most common way is to use pandas, a python library. However, python also has a built-in CSV module that can be used. We will show examples of using both below.

3.1.3.1 Installing and Importing Before beginning with any of these CSV manipulation tasks, it is necessary to install and import the correct modules and libraries. The following showcases this:

```
$ pip install pandas
```

```
import pandas as pd
import csv
```

3.1.3.2 Construction CSV files are files that exist elsewhere and have already been created. Therefore, for creation, we are not creating a CSV files, but rather deconstructing it into something that is usable.

In pandas, this means creating a dataframe, which is essentially and indexed table. In the python `csv` module, this means using the `csvreader` object within the module. Following are two examples showcasing how exactly to do this.

For the `csv` module:

```
file = open('/Users/jacksonmiskill/Downloads/biostats.csv') # opens
   ↪ the csv and creates the reader object for it
reader = csv.reader(file)
```

The `reader` is an object that was created by python developers to help parse through the `csv` files.

For the `pandas` module:

```
file = pd.read_csv("/Users/jacksonmiskill/Downloads/biostats.csv")
df = pd.DataFrame(file)
```

3.1.3.3 Accessing Values For the `csv` module:

It is more challenging to access files using the `csv` module as opposed to the `pandas` library. To make it more simple, it is necessary to convert the values that lie within the `csv` into a list in order to access.

```
data = []
for each_row in reader:
    if each_row: # you have to check for blank lines within the
       ↪ document
```

```
        data.append(each_row[2])

print(data)
```

For the `pandas` module, it is less complicated to access that values. This is because the module includes a function that converts the data into a dataframe. After converting, you can use the various methods that are within the dataframe to essentially pass in the correct values.

3.1.3.4 Examples Once the `csv` values have been accessed, creation of the graphics can begin. Starting with the `csv` module and then moving into `pandas` the following will demonstrate this action.

The `csv` file that will be utilized for the following examples can be found here. It represents made up data on a group of made up people such as age, height, and weight.

It is relatively easy, but slow, to create graphs with the `csv` module:

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

names = []
sex_data = []
age_data = []
height_data = []
weight_data = []

with open('/Users/jacksonmiskill/Downloads/biostats.csv',
          'r') as file:  # opens the csv and creates the reader
             ↪ object for it
    reader = csv.reader(file, delimiter=',')

    for each_row in reader:
        if each_row:  # you have to check for blank lines within the
           ↪ document

            names.append(each_row[0])
            sex_data.append(each_row[1])
            age_data.append(each_row[2])
            height_data.append(each_row[3])
            weight_data.append(each_row[4])

# Have to clean the data in order to actually create the graphs.
names.pop(0)
height_data.pop(0)
for i in range(0, len(height_data)):
    height_data[i] = int(height_data[i])

# let's plot the heights and weight of everyone. We could maybe use a
   ↪ hue here to denote who it is
plt.plot(names, height_data)
plt.xlabel("Name")
plt.ylabel("Height")
plt.xticks(rotation=90)
plt.savefig('images/csv-lineplot.png')
plt.savefig('images/csv-lineplot.svg')
plt.savefig('images/pandas-lineplot.pdf')
plt.legend()
plt.title("Names and Corresponding Height")
plt.show()
```

This code can be access from GitHub

This code produces Figure *csv-lineplot*:

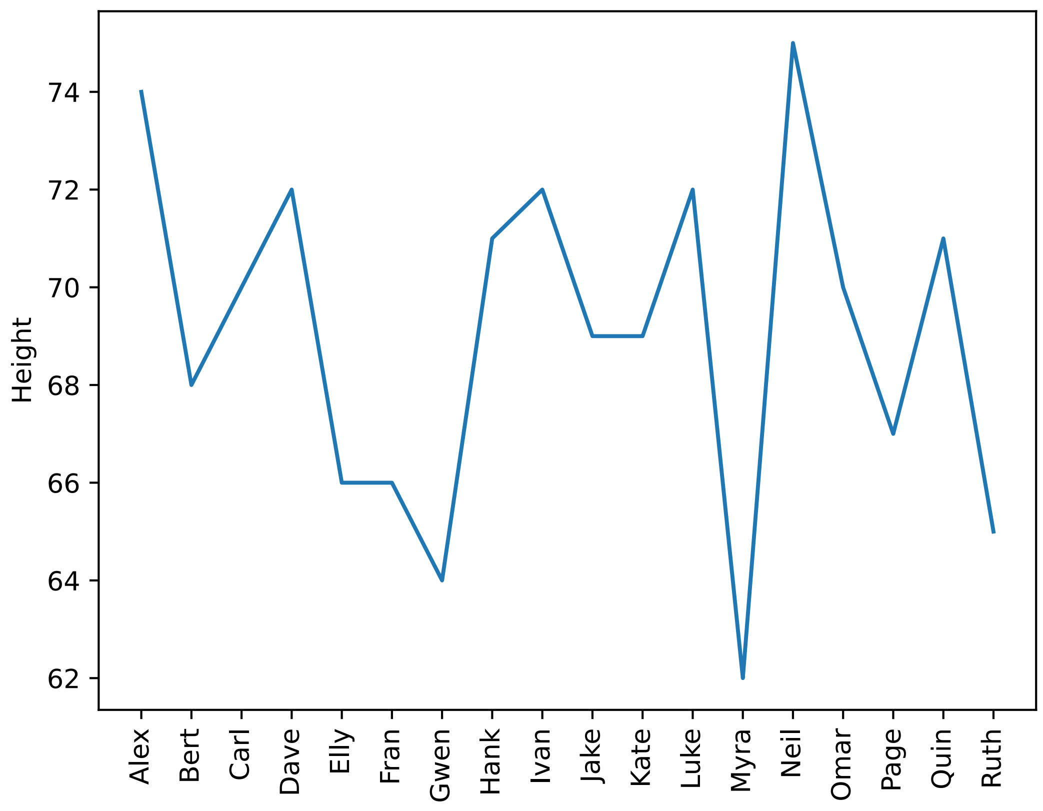


Figure 1: lineplot

Figure *csv-lineplot*: created using the data available here.

How do we use csv module without having to clean the data?

It is so much more simple to accomplish this with the `pandas` library:

```
file = pd.read_csv("/Users/jacksonmiskill/Downloads/biostats.csv")
biostats = pd.DataFrame(file)

plt.plot(biostats['Name'], biostats[' "Height (in)"'])
plt.xlabel("Name")
plt.ylabel("Height")
plt.xticks(rotation=90)
plt.savefig('images/pandas-lineplot.png')
plt.savefig('images/pandas-lineplot.svg')
plt.savefig('images/pandas-lineplot.pdf')
plt.title("Names and Corresponding Height")
plt.show()
```

This code can be accessed from GitHub

This code produces the Figure *pandas-lineplot*:

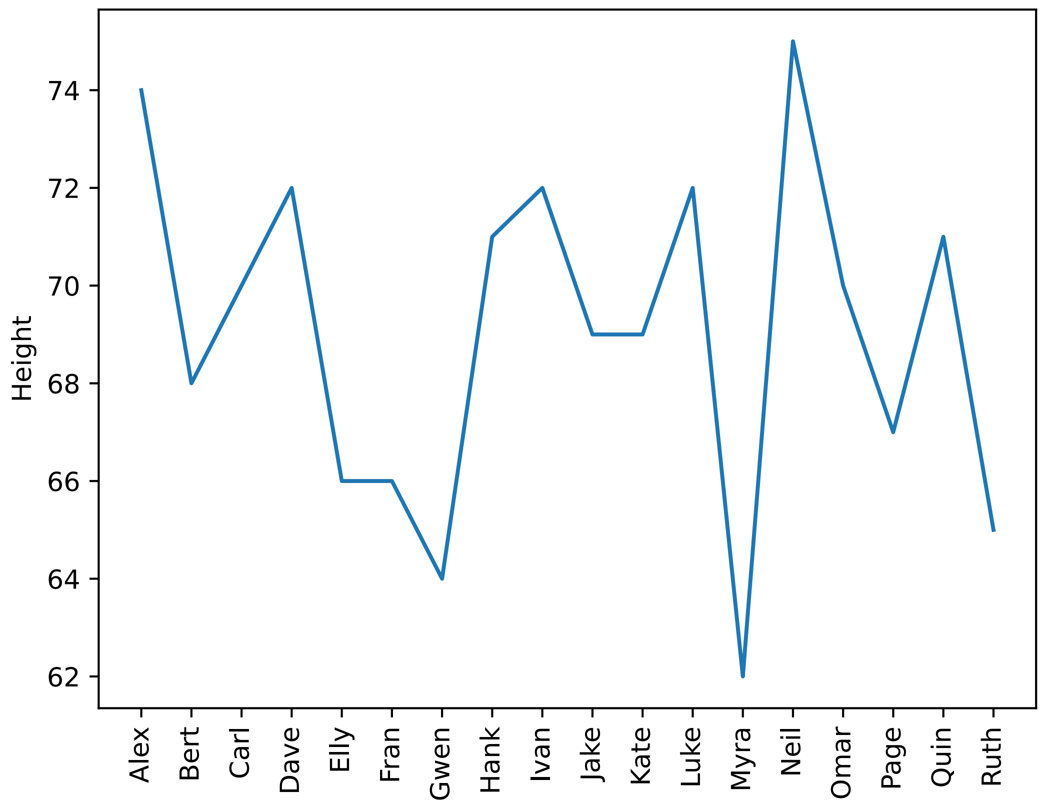


Figure 2: lineplot

Figure *pandas-lineplot*: created using the data available here. Figure *pandas-lineplot*: created using the data available here.

3.1.4 Sources

3.1.4.1 Lists

- https://towardsdatascience.com/python-basics-6-lists-and-list-manipulation-a56be62b1f95
- https://www.w3schools.com/python/python_ref_list.asp

3.1.4.2 Dictionaries

- https://www.pythonforbeginners.com/dictionary/dictionary-manipulation-in-python
- https://www.w3schools.com/python/python_ref_dictionary.asp
- https://www.w3schools.com/python/ref_dictionary_update.asp

3.1.4.3 CSV Files

- https://www.geeksforgeeks.org/creating-a-dataframe-using-csv-files/
- https://docs.python.org/3/library/csv.html#examples
- https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html
- https://docs.python.org/3/library/functions.html#open
- https://www.protechtraining.com/blog/post/python-for-beginners-reading-manipulating-csv-files-737#extracting-information-from-a-csv-file
- https://stackoverflow.com/questions/13039392/csv-list-index-out-of-range
- https://www.geeksforgeeks.org/visualize-data-from-csv-file-in-python/

3.2 Python Graphics

In Python, data and equations can be visually represented using graphs and plots. Whereby showcasing how to use different plotting libraries, this includes Matplotlib, Bokeh, and Seaborn.

3.2.1 Matplotlib

Matplotlib is a library that allows the user to visualize data. The library can create pie charts, bar charts, line plots, and other graphs specifically for data visualization. Matplotlib creates figures that can be manipulated and transformed. This includes manipulations of axes, labels, fonts, and the size of the images.

3.2.1.1 Installation To install matplotlib, please use the command:

```
$ pip install matplotlib
```

3.2.1.2 Import Statements The user will need to supply these import statements at the top of their code in order for Matplotlib to be imported.

```
import matplotlib.pyplot as plt
import numpy as np
```

3.2.1.3 Bar Chart In matplotlib, it is easy to create bar charts. For example, this is a demonstration of a simple bar chart using data from a user using Spotify.

```
import matplotlib.pyplot as plt

# you can also do this: from matplotlib import pyplot as plt

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz':
   ↪ 25}
categories = data.keys()
count = data.values()

# Creating the bar chart
plt.bar(categories,
        count,
        align='center',
        color='darkorange',
        width=0.4,
        edgecolor="royalblue",
        linewidth=4)

# Editing the bar chart's title, x, and y axes
plt.xlabel("Genre of Music")
plt.ylabel("Number of songs in the genre")
plt.title("Distribution of Genres in My Liked Songs")
plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *barchart*.

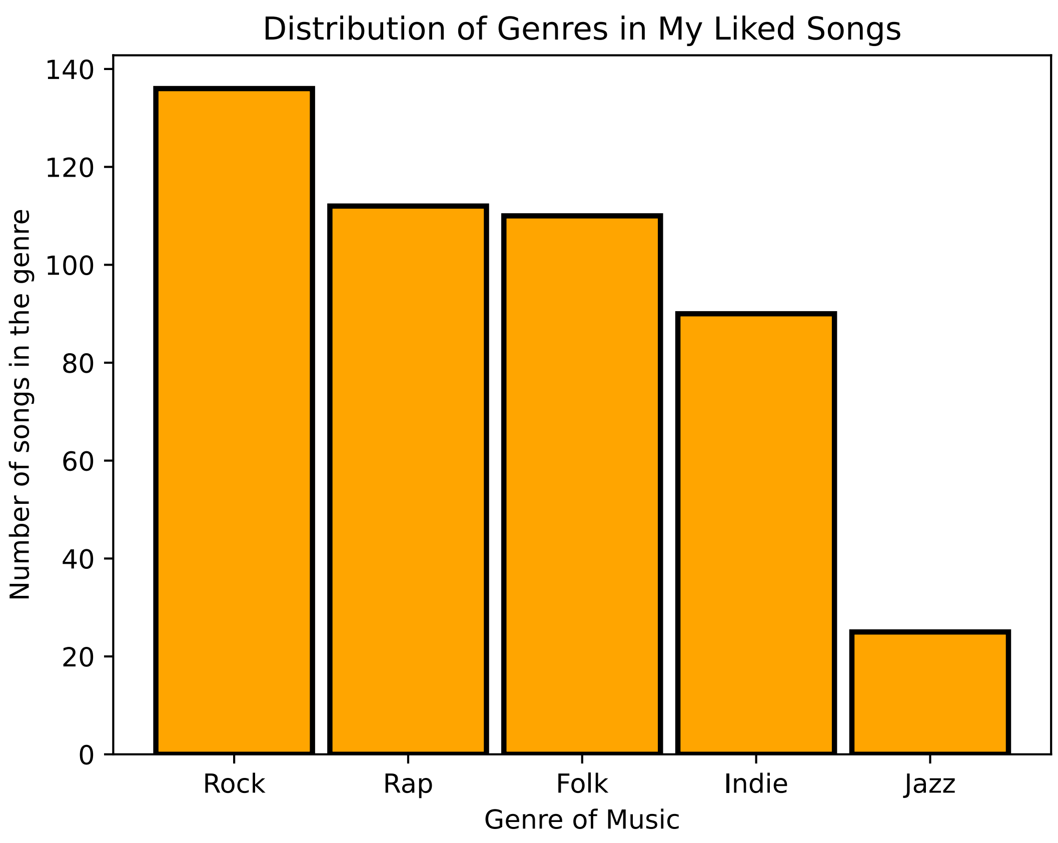


Figure 3: barchart

Figure *barchart*: Barchart created from data from Spotify

3.2.1.4 Line Chart The matplotlib library in python allows for comprehensive line plots to be created. Here a line chart was created using a for loop to generate random numbers in a range and plot it against the `x` and `y` axis to display the changes between two variables/data sets.

```
import matplotlib.pyplot as plt
import random

x = []
y = []
for i in range(0, 100):
    x.append(i)
    value = random.random() * 100
    y.append(value)

# creating the plot and labeling axes and title
plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Plot Test")
plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *linechart*.

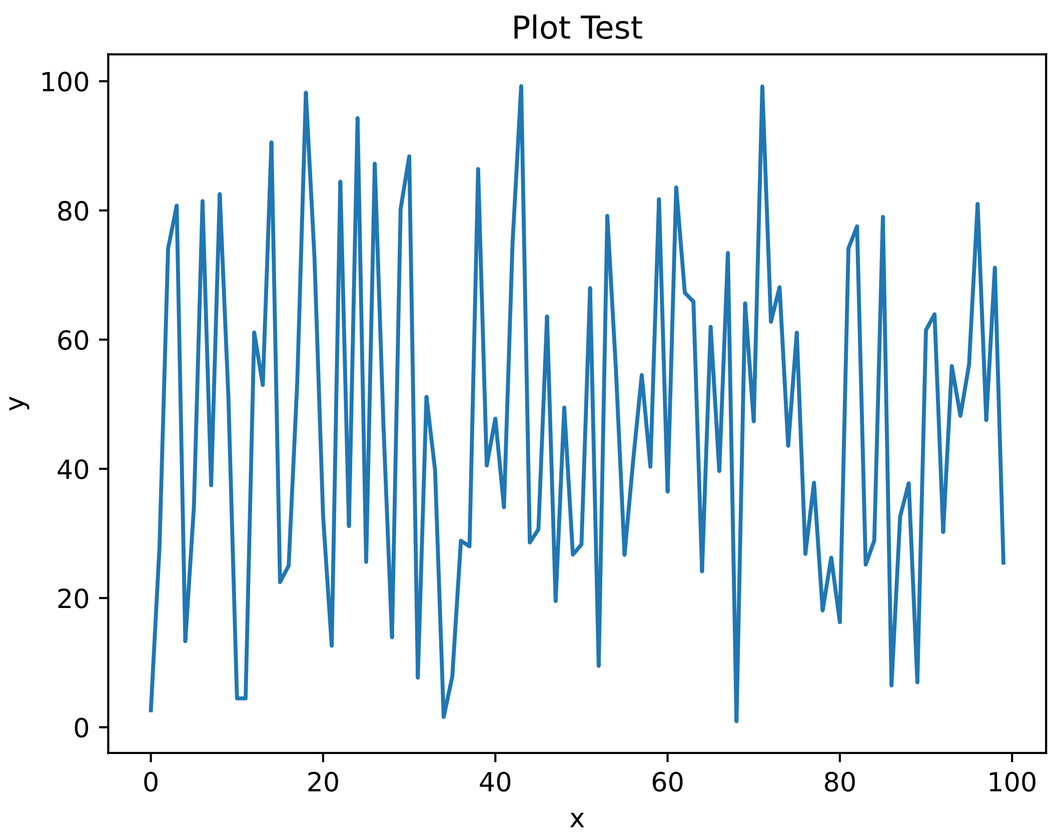


Figure 4: linechart

Figure *linechart*: Linechart created from random variables

3.2.1.5 Pie Chart A pie chart is most commonly used when representing the division of components that form a whole thing e.g. showing how a budget is broken down into separate spending categories. In matplotlib, the function `pie()` creates a pie chart. In the following code example, a user's Spotify data will be displayed as a pie chart.

```
import matplotlib.pyplot as plt

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz':
    ↪ 25}
categories = data.keys()
count = data.values()

# Creating the pie chart
plt.pie(count, labels=categories)

plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *piechart*.

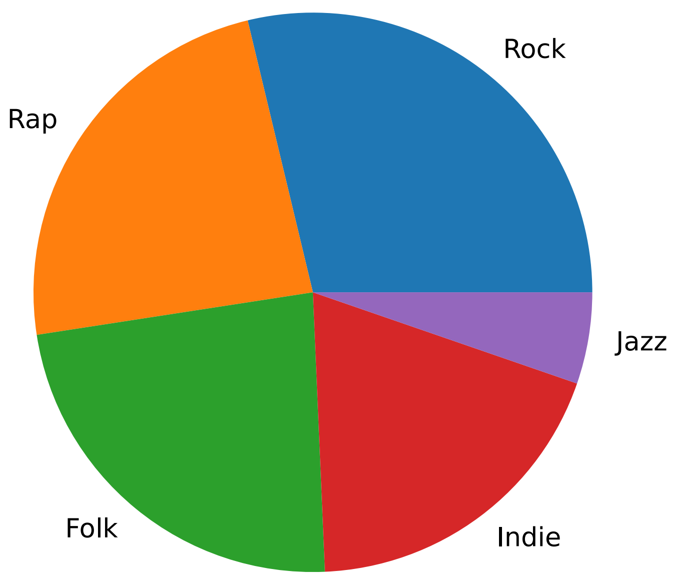


Figure 5: piechart

Figure *piechart*: Barchart created from data from Spotify

3.2.1.6 Contour Plot Unlike the previous types of plots shown, contour plots allow data involving three variables to be plotted on a 2D surface. In this example, an equation of a hyperbolic paraboloid is graphed on a contour plot.

```
import matplotlib.pyplot as plt
import numpy as np

#creating an equation for z based off of variables x,y
x, y = np.meshgrid(np.linspace(-10, 10), np.linspace(-10, 10))
z = 9*(x**2+1)+8*x-(y**2)
levels = np.linspace(np.min(z), np.max(z), 15)
```

```
#creating a contour graph based off the equation of z
plt.contour(x,y,z, levels=levels)



plt.xlabel("x")
plt.ylabel("y")
plt.title("Function of z(x,y)")
plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *contourplot*.

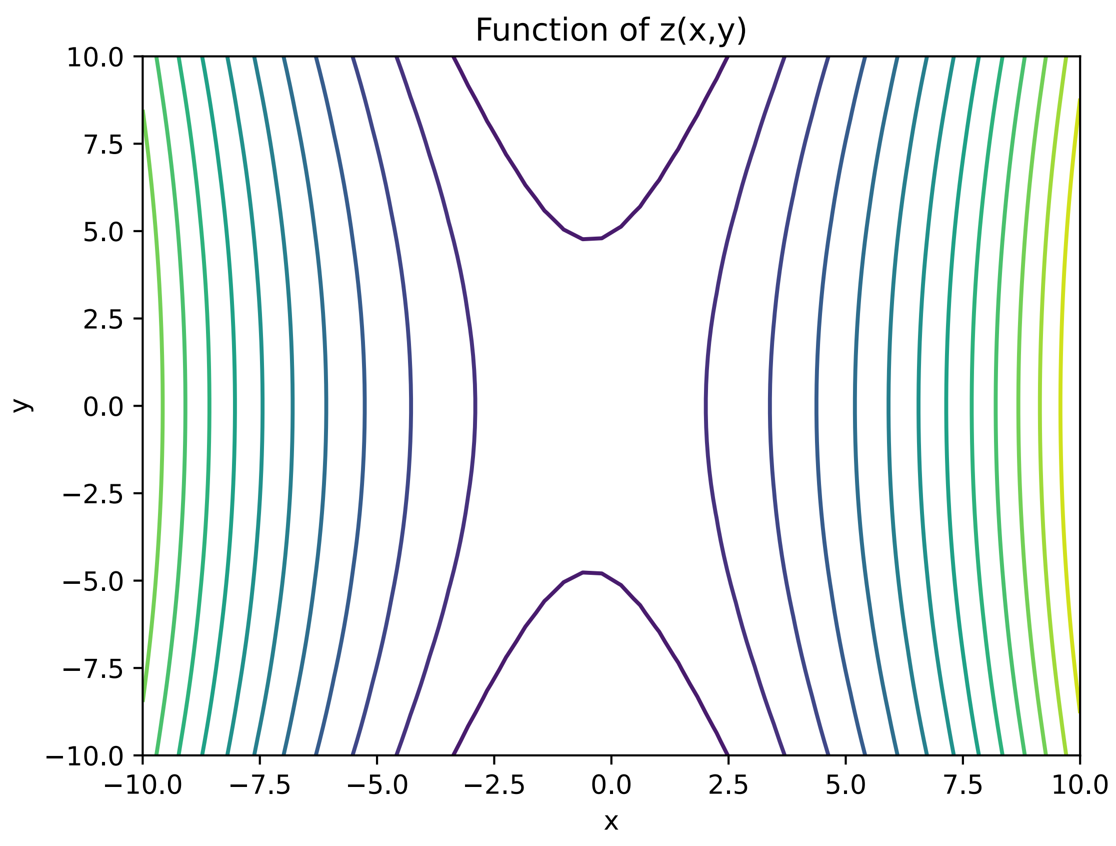


Figure 6: contour

Figure *contourplot*: Multivariable (x, y, z) Equation Plotted

A contour plot allows data and equations consisting of three variables to be plotted through plotting

3D surfaces as 2D slices on a `xy` plane. Matplotlib can display data and equations through contour graphs after they are inputted. Shown below are the parameters for `plt.contour`.

```
plt.contour([x, y], z, levels)
```

The independent variables `x` and `y` must be defined so the dependent variable `z` can be defined. The variables can come in the form of a list or dictionary or as an equation. The `levels` parameter determines the number of contour lines that can be drawn.

3.2.1.7 Titles, Labels, and Legends

3.2.1.7.1 Titles Titles are necessary to let the reader know about your graph or plot is exactly about. To give a title to your whole graph in matplotlib, simply type:

```
plt.title("Title you want to set").
```

3.2.1.7.2 x-axis labels and y-axis labels Within the matplotlib library are the functions `plt.xlabel()` and `plt.ylabel()`. All these functions do is set a string to the two axes. To use these functions, simply type:

```
plt.xlabel("Label you want to set")
plt.ylabel("Label you want to set")
```

3.2.1.7.3 Legend Sometimes, a legend may be necessary to let the reader know which part of the graph/plot corresponds to each part of the data shown. To show a legend, use the command:

```
plt.legend()
```

3.2.1.8 Rotating Ticks When a chart is created, ticks are automatically created on the axes. By default, they are set horizontally; however, they can be rotated using `plt.xticks(degrees)` for the `x-axis` or `plt.yticks(degrees)` for the `y-axis`. This can be shown by this simple example.

```
import matplotlib.pyplot as plt

x = range(0,4)
```

```
y = x
plt.plot(x,y)

# Rotating Ticks
plt.xticks(rotation=90)
plt.yticks(rotation=45)

plt.xlabel('x values')
plt.ylabel('y values')
plt.title(r'$y=x$')
plt.show()
```

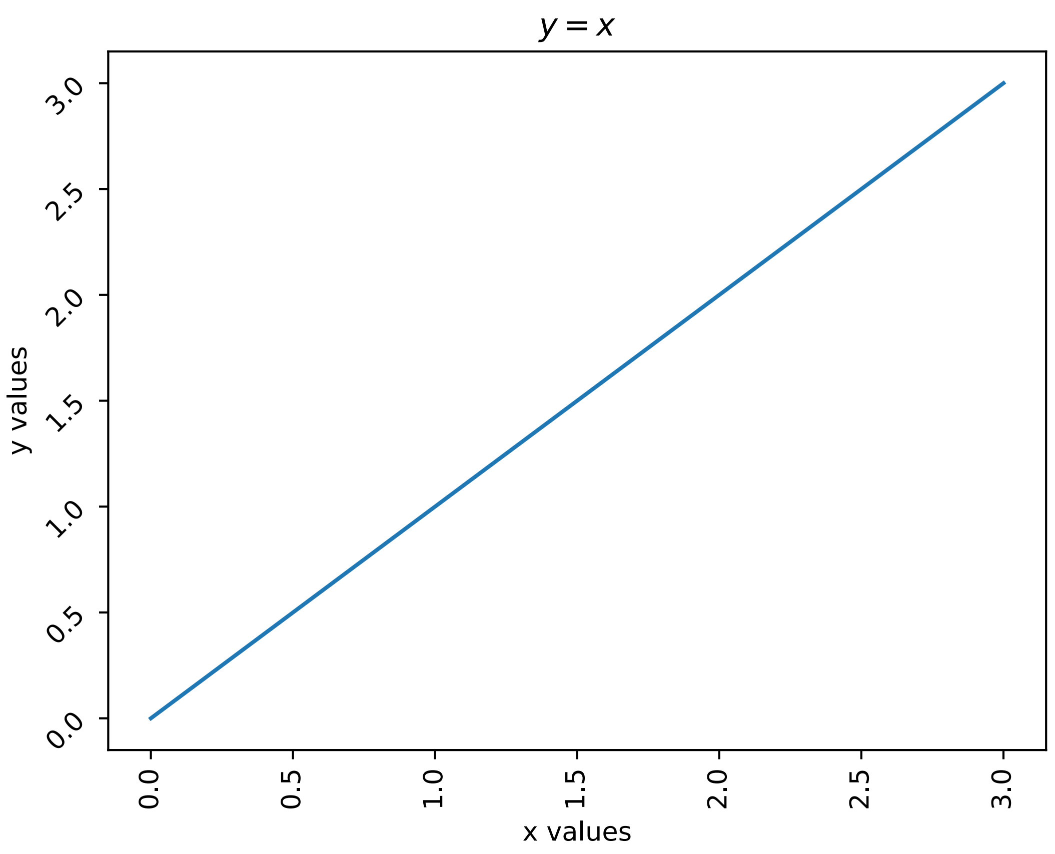


Figure 7: ticks

Figure *ticks* `x-axis` ticks rotated by 90° and `y-axis` ticks rotated by 45°

3.2.1.9 Exporting

3.2.1.9.1 Saving Chart as Files After a chart is created and displayed, it can be exported as a file outside the code using this command:

```
plt.savefig("fname", dpi='figure')
```

The name and format of the file are set as a string using `fname`. Make sure to specify the format of the file by using a `.` after the file name and specify the type after such as `.pdf`, `.png`, `svg`, etc.

The parameter `dpi` sets the DPI (Dots per Inch) of the image being saved. Specify this number in the form of a float. For example, set `dpi=300`.

Additionally, there is another way to save files that may be faster than calling a specific method for each file. The following code showcases this:

```
import matplotlib.pyplot as plt
import os
from matplotlib import pyplot


def save():
    name = os.path.basename(__file__).replace(".py", "")
    plt.savefig(f'/filepath/{name}.png')
    plt.savefig(f'filepath/{name}.pdf')
    plt.savefig(f'filepath/{name}.svg')
    plt.show()
```

This code can be accessed on GitHub

3.2.1.9.2 Display The very last command that should be written is `plt.show()`, as this command displays the graph that you made. To show, simply type:

```
plt.show()
```

3.2.2 Bokeh

Bokeh is a Python library useful for generating visualizations for web browsers. It generates graphics for all types of plots and dashboards powered by JavaScript without the user's need to write any

JavaScript code. The guide below will walk you through useful Bokeh commands and features.

3.2.2.1 Installation

To install Bokeh, please use the command:

```
$ pip install bokeh
```

3.2.2.2 Import Statements

To plot figures, we import the `show` and `figure` functions from the Bokeh libraries.

```
from bokeh.io import show
from bokeh.plotting import figure
```

3.2.2.3 Bokeh Plotting Interface

`bokeh.plotting` is the library's main interface. It gives the ability to generate plots easily by providing parameters such as axes, grids, and labels. The following code shows some of the simplest examples of plotting a line and a point on a chart.

```
from bokeh.io import show
from bokeh.plotting import figure

# labeling the title, specifying the range of the x-axis, labeling
   ↪ the y-axis, specifying the height to be 500 pxls
p = figure(title = "My Graph", x_range = [0,20], y_axis_label = "the
   ↪ y axis", height = 500)

# plotting a line from (0,0) to (20,20); any of the CSS colors can be
   ↪  used
p.line([0,20],[0, 20], color='indigo')

# plotting a point (circle) at (5,10)
p.circle(5,10, color = 'green')

show(p)
```

This program can be downloaded from GitHub

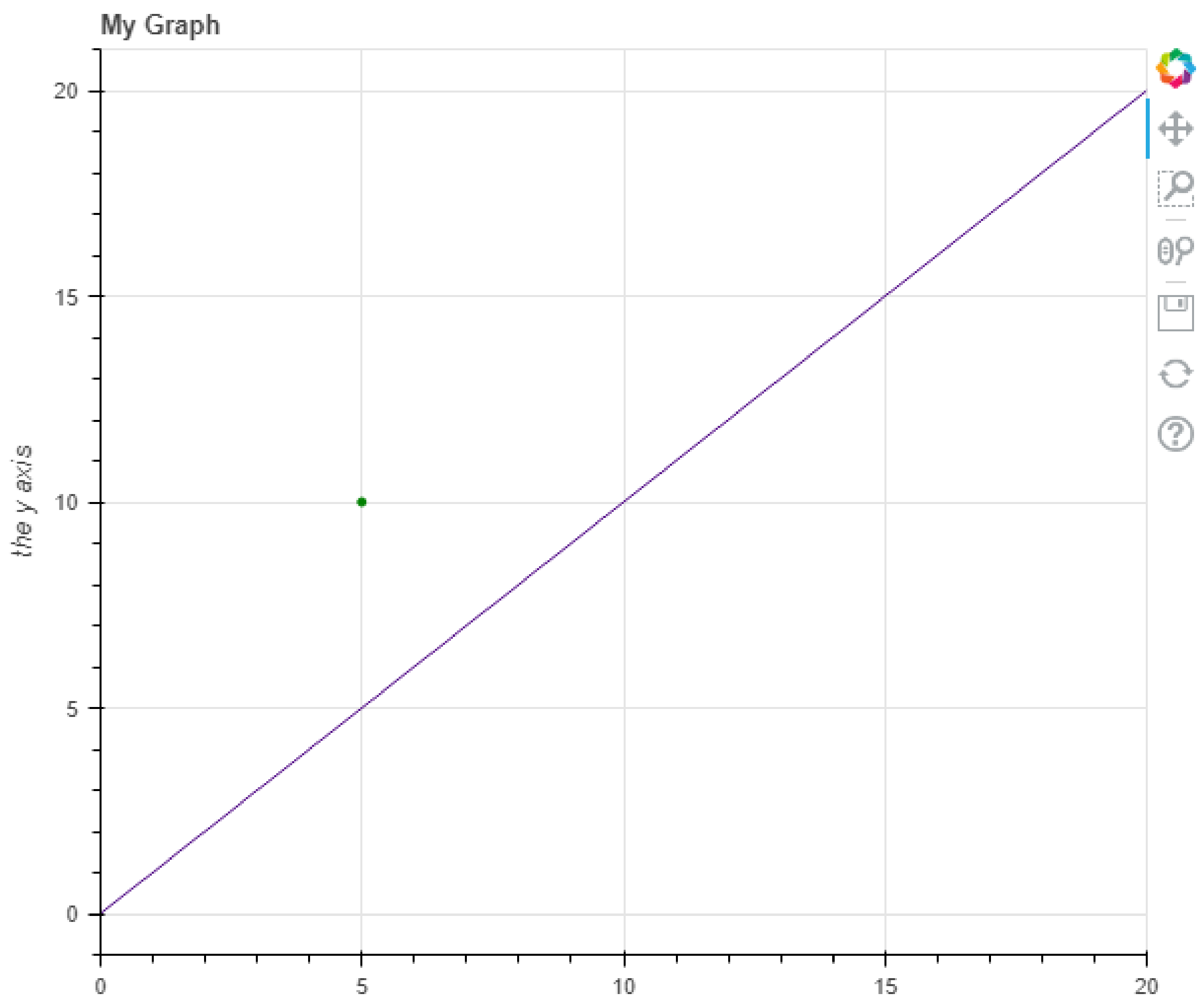


Figure 8: figure

Figure *lineplot*: Figure created with Bokeh.

3.2.2.4 Figure Parameters Example

- x_axis_label and y_axis_label: labels for the x and y axis
- x_range and y_range: specifications for the range of the x and y axis
- title: text title for your graph
- width and height: width and height of your graph in pixels
- background_fill_color: the background of the figure (takes any CSS colors)

3.2.2.5 Saving Figures Bokeh also supports outputs to a static HTML file with a specific name.

```
from bokeh.plotting import output_file
output_file("name.html")
```

After importing the Bokeh plotting interface, it is possible to be able to create different types of plots utilizing the figure created with the figure function.

3.2.2.5.1 Saving Figures as PNG In order to save figures as a PNG, both Selenium and a web driver will need to be installed. We will use Chromium here for our web driver. To install both at once, use the commands:

(Windows)

```
$ pip install selenium chromedriver-binary
$ pip install chromedriver-binary-auto
```

There seems to be issues installing `chromedriver-binary` on Mac computers due to the built-in security, so it is recommended to simply save Bokeh figures as a `.html` file.

When writing a program, Chromium must be added to the PATH through these import statements:

```
from selenium import webdriver
import chromedriver_binary
```

Bokeh appears to support saving files as a `.svg` but it seems to have bugs and is not recommended. To use the functions, `export_png()` and `export_svg()` must be imported, and can be used as follows:

```
from bokeh.io import export_png, export_svg

export_png(fig, filename="file-name.png")
export_svg(fig, filename="file-name.svg")
```

Note that Chromium is slow and this process may take delay the execution and performance of the program.

Similarly to matplotlib, Bokeh can utilize a function to save all created images.

```
from matplotlib import pyplot as plt
from bokeh.io import export_png, export_svg
import os
```

```

def save(p):
    name = os.path.basename(__file__).replace(".py", "")
    export_png(p, filename=f"images/{name}.png")
    export_svg(p, filename=f"images/{name}.svg")
    plt.show(p)
```

This code can be accessed on GitHub.

3.2.2.6 Scatter Plot The Bokeh library provides various marker shapes for marking points on the scatter plot. The example below demonstrates how to create a scatter plot with two points at locations (1,3) and (2,4) respectively with circular and square marker shapes. The size parameter controls the size of the marker.

```
from bokeh.io import show
from bokeh.plotting import figure

p = figure(title="Scatter Plot")

# Circle
p.circle([0,3], [4,5], size = 10)

# Square
p.square([1,2], [3,4], size = 10)

show(p)
```

This program can be downloaded from GitHub

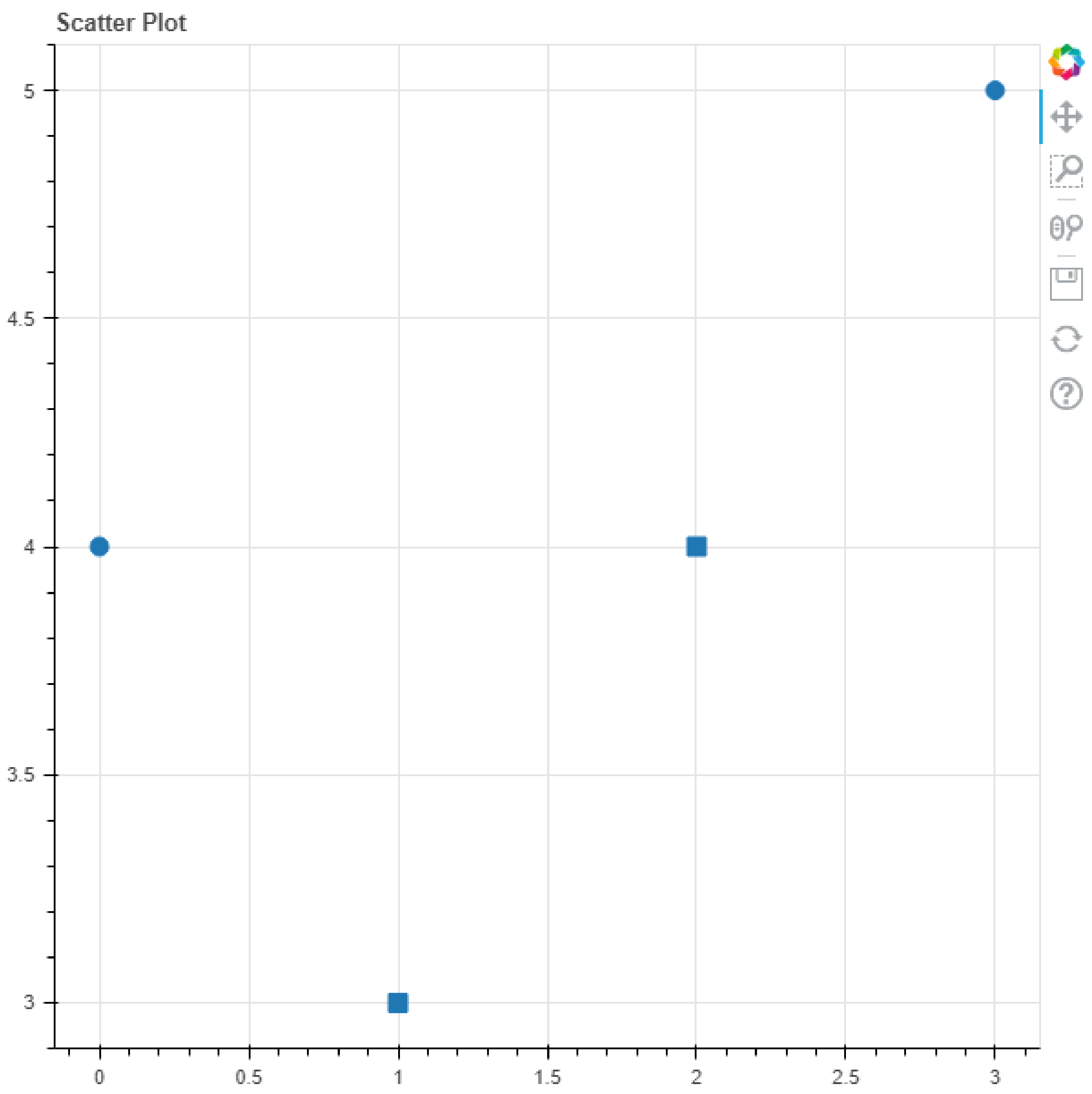


Figure 9: Scatter Plot

Figure *Scatter Plot*: Scatter Plot created with user Spotify data.

The list of all possible marker types and the functions used to create them can be found here

3.2.2.7 Line Plots The library provides a series of functions for creating various types of line graphs ranging from a single line graph, step line graph, stacked line graph, multiple line graph, and so on.

```
from bokeh.io import show, export_png, export_svg
from bokeh.plotting import figure
import random

x = []
y = []
for i in range(0, 100):
    x.append(i)
    value = random.random() * 100
    y.append(value)

p = figure(title="Plot Test", x_axis_label = "x", y_axis_label = "y")
p.line(x,y)

show(p)
```

This program can be downloaded from GitHub

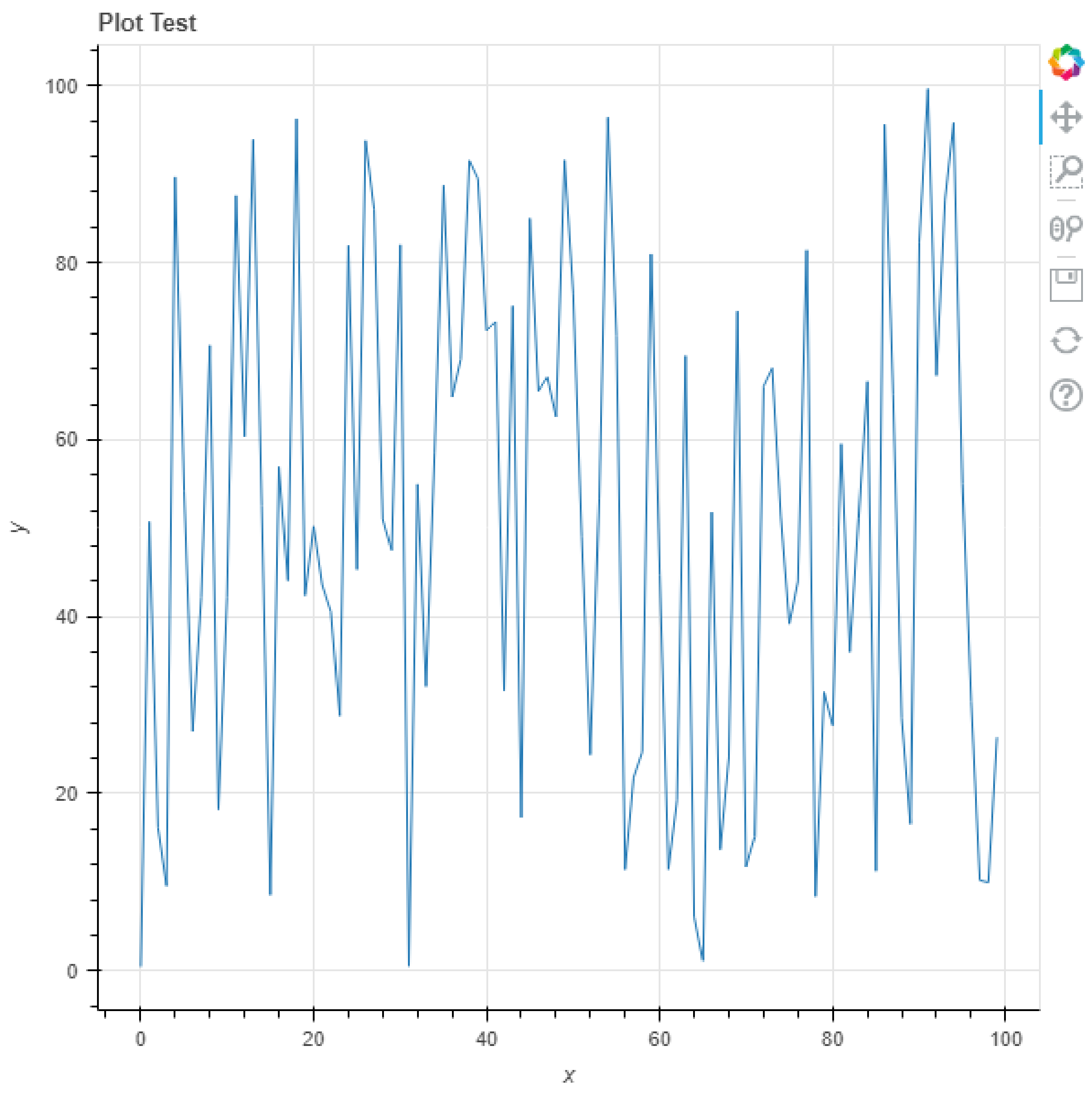


Figure 10: Line Plot

Figure *Line Plot*: Line Plot created with user Spotify data.

You can find the source code for other types of line plots here: http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html

3.2.2.8 Bar Chart Similarly, the `hbar()` and `vbar()` functions can be used to display horizontal and vertical bar graphs, respectively.

```
from bokeh.io import show, export_png, export_svg
from bokeh.plotting import figure

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz':
    ↪ 25}
x = list(data.keys())
y = list(data.values())

p = figure(x_range = x, title="Bar Chart")

p.vbar(x=x, top = y, line_color = 'black',color='orange', width =
    ↪ 0.9, line_width = 2)

show(p)
```

This program can be downloaded from GitHub

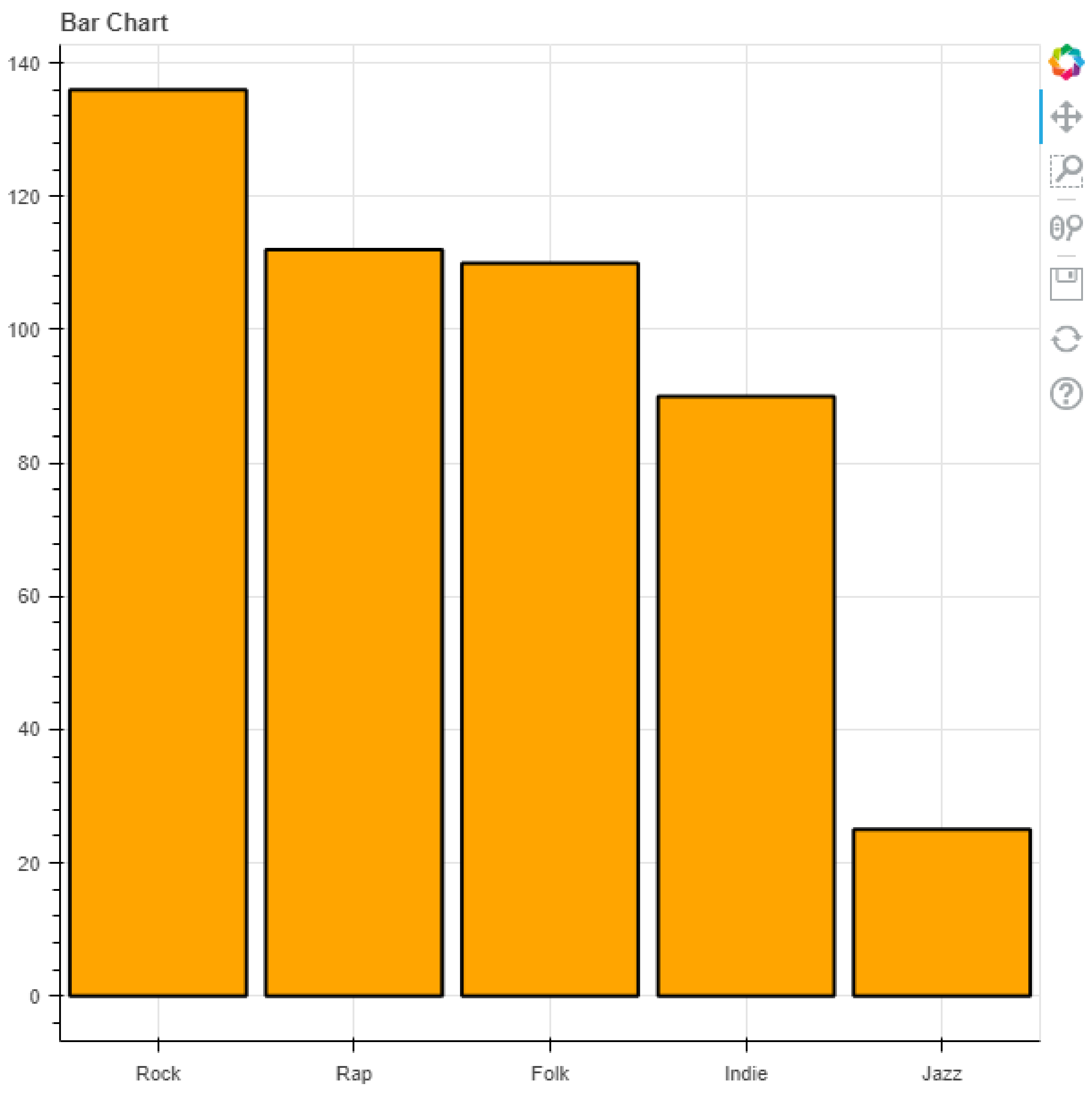


Figure 11: Bar Chart

Figure *Bar Chart*: Bar Chart created with user Spotify data.

3.2.3 Seaborn

Seaborn, like Matplotlib, is a data visualization tool. However, the graphs and charts that Seaborn can create are more complex than Matplotlib. The graphs that are created in Seaborn are more statistically detailed. Unlike matplotlib, Seaborn draws upon other imported libraries such as Matplotlib, Numpy, and Pandas. This is because Seaborn relies on more complex math (Numpy) and data frames (generated from Pandas) that are passed into its functions as the data.

Several types of plots can be made from Seaborn; they are relational, distributional, categorical, regression, and matrix plots.

We have created examples to demonstrate the abilities of Seaborn.

3.2.3.1 Installation

Seaborn can be installed in the same way as the other libraries installed earlier. The user who is installing the library should make sure that it is being installed in the correct environment.

```
$ pip install seaborn
```

3.2.3.2 Import Statements

The user will need to supply these import statements at the top of their code in order for Seaborn to be imported. Additionally, the data created for the examples represents a user's Liked songs from Spotify.

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

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz':
   ↪ 25}
categories = list(data.keys())
count = list(data.values())
personal_rank = [3, 4, 2, 1, 5]
```

3.2.3.3 Relational Plots

Relational plots showcase the relationship between variables in a visual format. It is a broad term for data representation. Examples of relational plots in Seaborn are `relplot` `lineplot` and `scatterplot`.

It is simple to create a relational plot with Seaborn:

```
sns.relplot( x=months , y=photos)
plt.xlabel("Month of the year")
plt.ylabel("Amount of photos taken")
plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *lineplot*

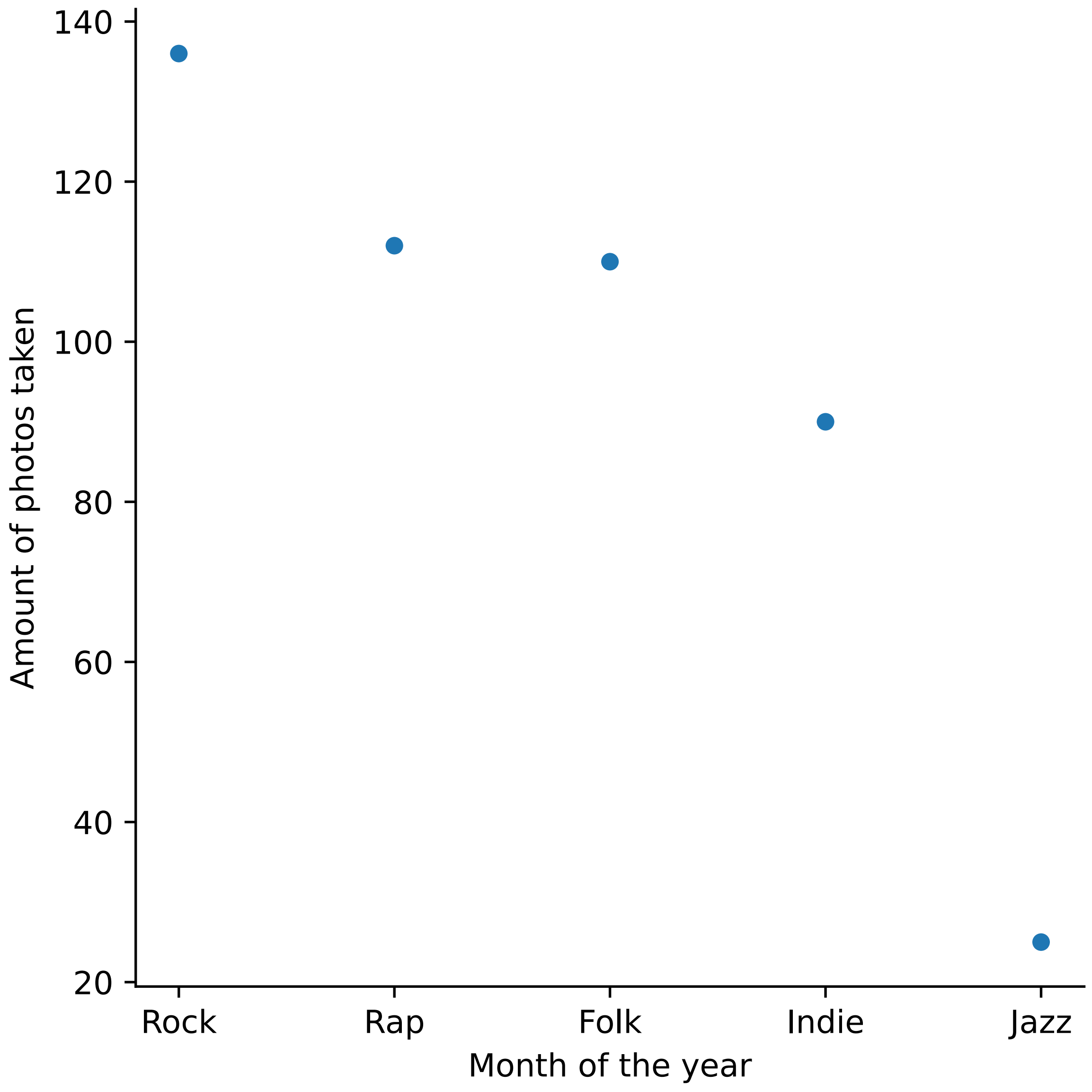


Figure 12: lineplot

Figure *lineplot*: Lineplot created from user Spotify data.

3.2.3.4 Distribution Plots A distribution plot shows how the data is concentrated in a range of values. The graph that appears looks similar to a bar graph in that there are bars. However, these bars show the concentration of a variable across a range of values rather than the quantity possessed by a singular variable. The distributional plots in Seaborn are `displot` `histplot` `kdeplot` `ecdfplot` and `rugplot`.

```
sns.displot(x=source, y=value)
plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *displot*

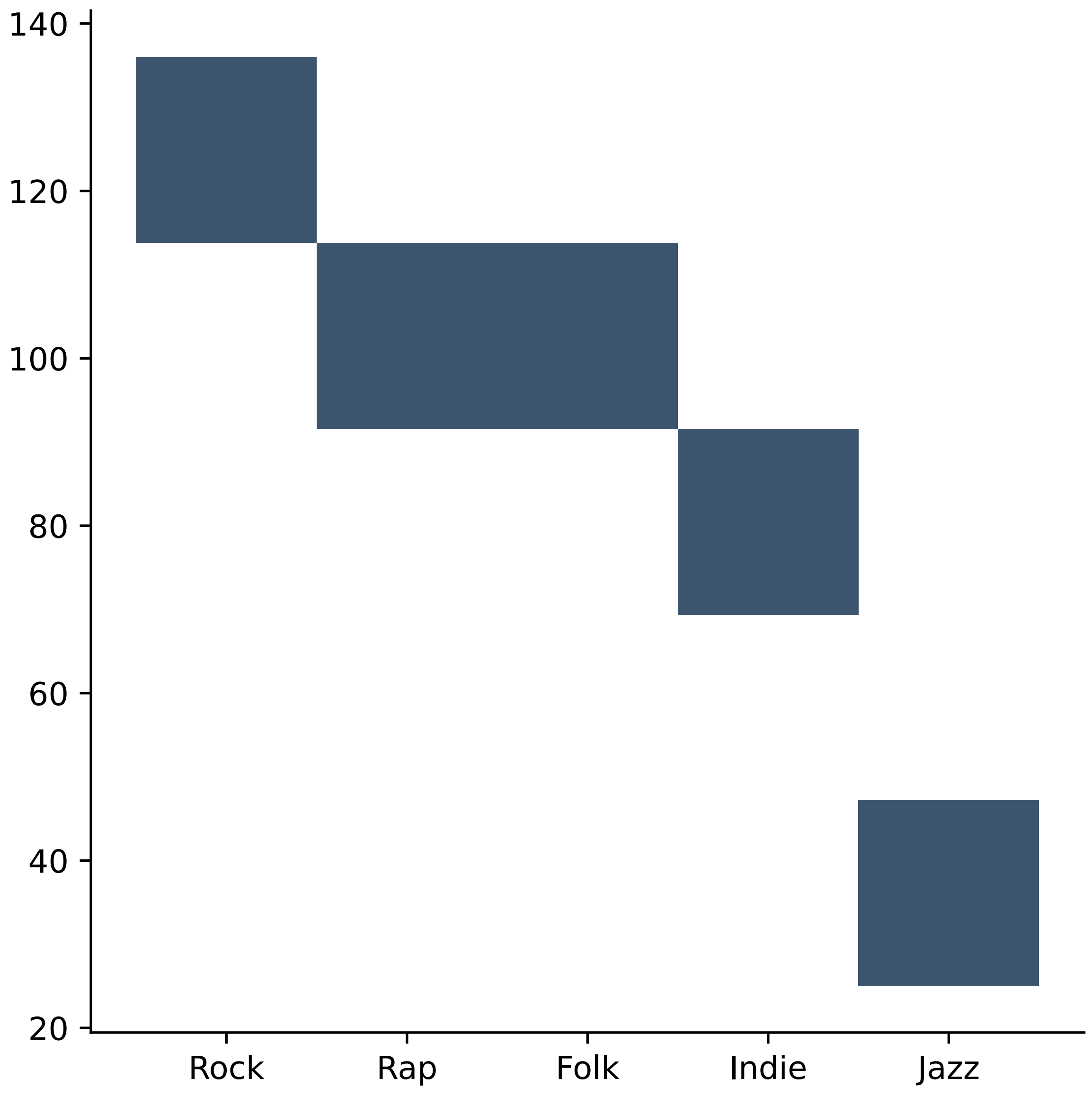


Figure 13: displot

Figure *displot*: Displot created from user Spotify data.

3.2.3.5 Categorical Plots Categorical plots are statistical graphs that help visualize the magnitudes of different variables in a dataset. A type of categorical plot is a bar chart, exactly like the example produced in the Matplotlib section. The categorical plots are `catplot` `stripplot` `swarmplot` `boxplot` `violinplot` `boxenplot` `pointplot` `barplot` and `countplot`.

Categorical plots are relatively simple to implement. If using the `catplot` method, it is necessary to include the `kind` parameter.

```
sns.barplot(x=source, y=value)
plt.show()
```

This program can be downloaded from GitHub

The output from the program is showcased in Figure *catplot*

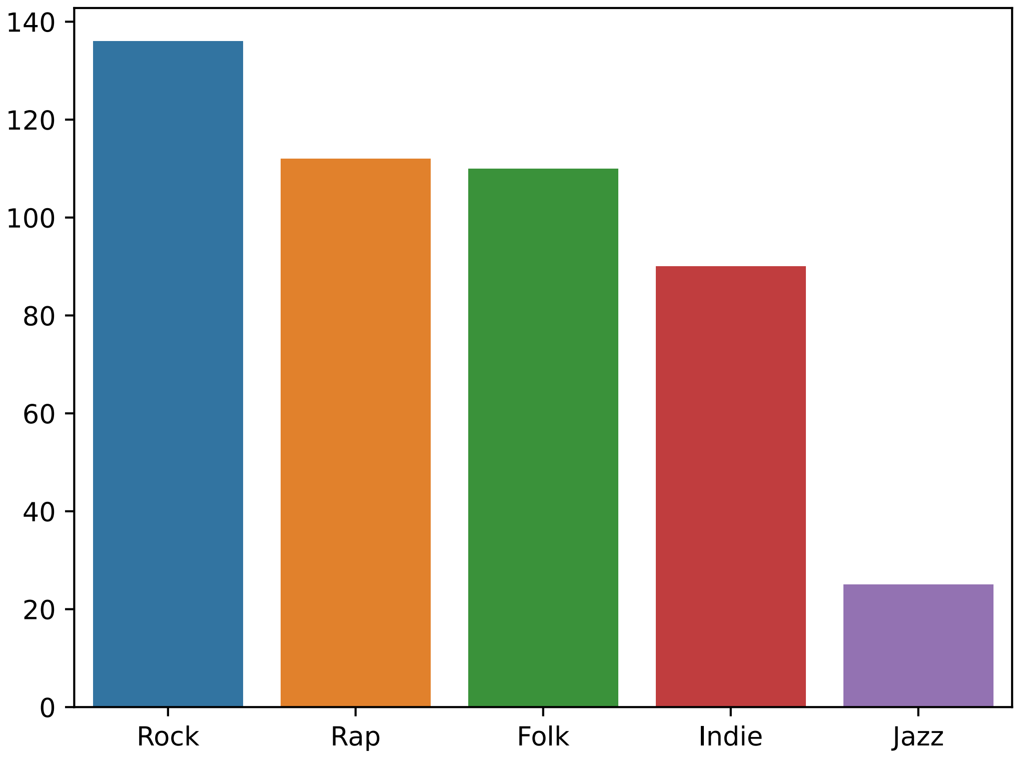


Figure 14: catplot

Figure *catplot*: Created from user Spotify data.

3.2.3.6 Regression Plots Regression plots are like relational plots in the way that they help visualize the relationship between two variables. Regression plots, however, show the linear correlation that may or may not exist in a scatter plot of data. Their regression plots are `lmplot` `regplot` and `residplot`.

Regression plots are simple to implement:

```
sns.regplot(x=months, y=photos)
plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *regplot*

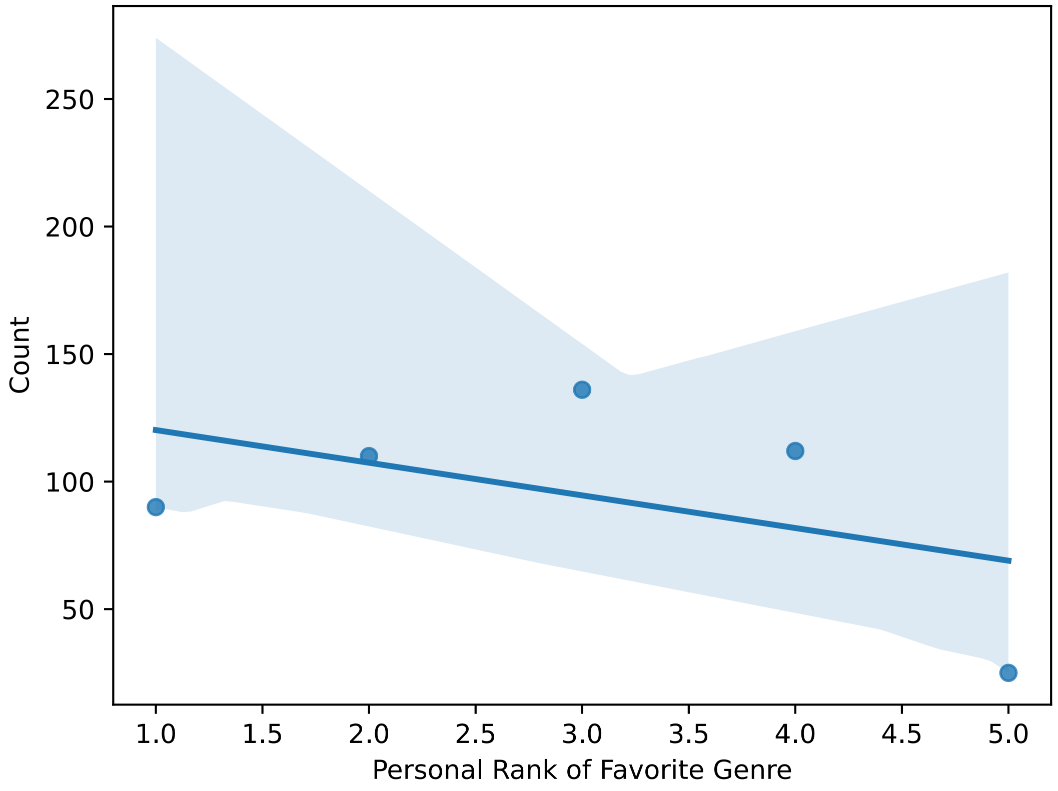


Figure 15: regplot

Figure *regplot*: Created from user Spotify data.

Each of these plots can be manipulated to the users needs via the API that is listed in the sources section.

3.2.3.7 Saving Figures

Saving figures created by Seaborn is quite simple. This is because it is the exact same as in Matplotlib.

To save a figure:

```
plt.savefig('figure_path/figure_name')
```

This program can be downloaded from GitHub

3.2.4 Sources

3.2.4.1 Matplotlib

- https://matplotlib.org/
- https://matplotlib.org/stable/api/pyplot_summary.html
- https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/
- https://www.geeksforgeeks.org/bar-plot-in-matplotlib/

3.2.4.2 Bokeh

- http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html
- http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html
- http://docs.bokeh.org/en/latest/
- https://docs.bokeh.org/en/latest/docs/reference/plotting/figure.html
- https://docs.bokeh.org/en/latest/docs/user_guide/export.html

3.2.4.3 Seaborn

- https://seaborn.pydata.org/api.html
- https://www.geeksforgeeks.org/python-seaborn-tutorial/
- https://www.geeksforgeeks.org/introduction-to-seaborn-python/
- https://www.geeksforgeeks.org/plotting-graph-using-seaborn-python/
- https://stackoverflow.com/questions/30336324/seaborn-load-dataset
- https://github.com/mwaskom/seaborn-data/blob/master/planets.csv

3.3 Pandas Graphics

Pandas is a useful library for working with data. It relies on storing data in data frames. Instead of using a set of ordered pairs, a data frame in Pandas is similar to an Excel Spreadsheet or an SQL data frame and supports multiple rows and columns in a tabular fashion.

Visualization for Panda's data frames are an important tool for understanding the data set. Panda's visualization tools are based off of Matplotlib, so many of the plotting functions are the same.

3.3.1 Installation

To install Pandas, please use the command:

```
$ pip install pandas
```

3.3.2 Import Statements

The user will need to import both Pandas and Matplotlib in order to create and visualize data frames. In addition, Python's `numpy` may be a useful library for mathematical procedures on the data.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

3.3.3 Bar Chart

Creating a bar chart with data frames is similar to creating bar charts with Matplotlib, with a couple differences in how you manipulate the data. In the following program, we use the same data and modifications as the Matplotlib bar chart example that can be found on Github.

```
import matplotlib.pyplot as plt
import pandas as pd

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz':
   ↪ 25}
categories = data.keys()
count = data.values()

df = pd.DataFrame({'Count':count, 'Categories':categories})

# Creating the bar chart
df.plot.bar(
        x='Categories',
        y='Count',
        align='center',
        color='orange',
        width=0.9,
        edgecolor="black",
        linewidth=2)

# Editing the bar chart's title, x, and y axes
plt.xlabel("Genre of Music")
plt.ylabel("Number of songs in the genre")
plt.title("Distribution of Genres in My Liked Songs")
plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *barchart*.

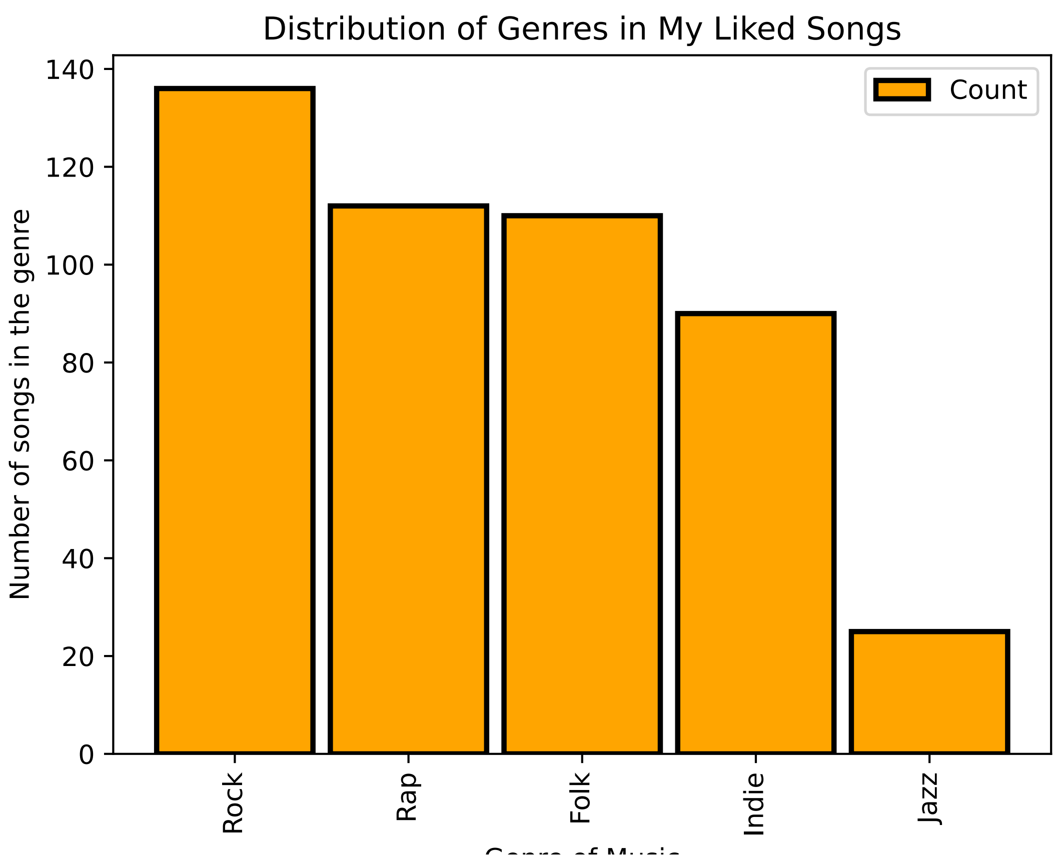


Figure 16: barchart

Figure *barchart*: Barchart created from data from Spotify.

Note the differences in creating the chart. Since data frames support multiple dimensions of data, the x and y we want to graph must be specified in `df.plot.bar()`. However, editing the title and axes are the same as in Matplotlib.

3.3.4 Line Chart

A line chart is typically used for time series data and non-categorical data. Pandas supports line chart visualization with `plot.line()`. We use the same data and modifications as the Matplotlib line chart example that can be found on Github. Note that since this data relies on random number generation the graphs will look slightly different each time.

```
import matplotlib.pyplot as plt
import pandas as pd
import random

x = []
y = []
for i in range(0, 100):
    x.append(i)
    value = random.random() * 100
    y.append(value)


df = pd.DataFrame({'x':x, 'y':y})

# creating the plot and labeling axes and title
df.plot.line(x='x', y='y')
plt.ylabel("y")
plt.title("Plot Test")

plt.show()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *linechart*.

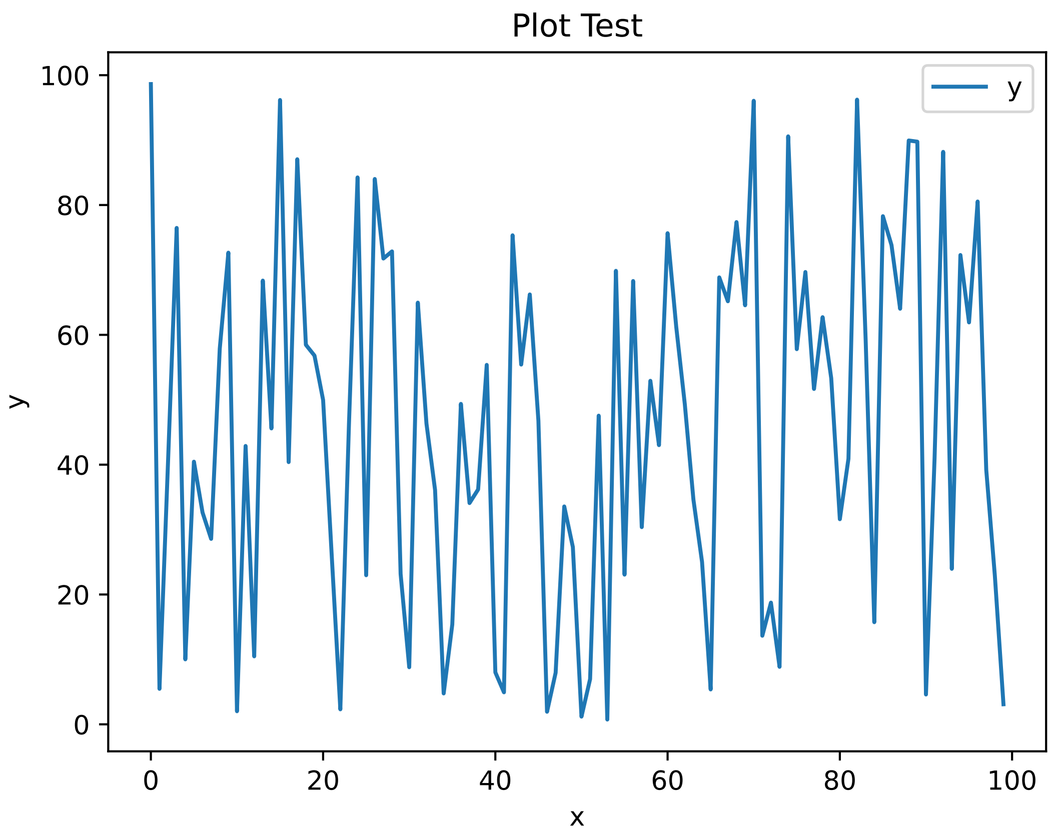


Figure 17: linechart

Figure *linechart*: Barchart created from random number generation.

3.3.5 Pie Chart

A pie chart is useful for showing a divison of a whole. Data that can be represented by a pie chart can also be used to make a bar chart, since both typically use categorical counts. Pandas uses `plot.pie()` to make a pie chart, in a manner similar to `plot.bar()`. We use the same data used to create the line chart for this visualization.

```
import matplotlib.pyplot as plt
import pandas as pd

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz':
    ↪ 25}
```

```
categories = data.keys()
count = data.values()

df = pd.DataFrame({'Count':count},index=categories)
plot = df.plot.pie(y='Count',legend=None)
save()
```

This program can be downloaded from GitHub

The output of this program is showcased in Figure *piechart*.

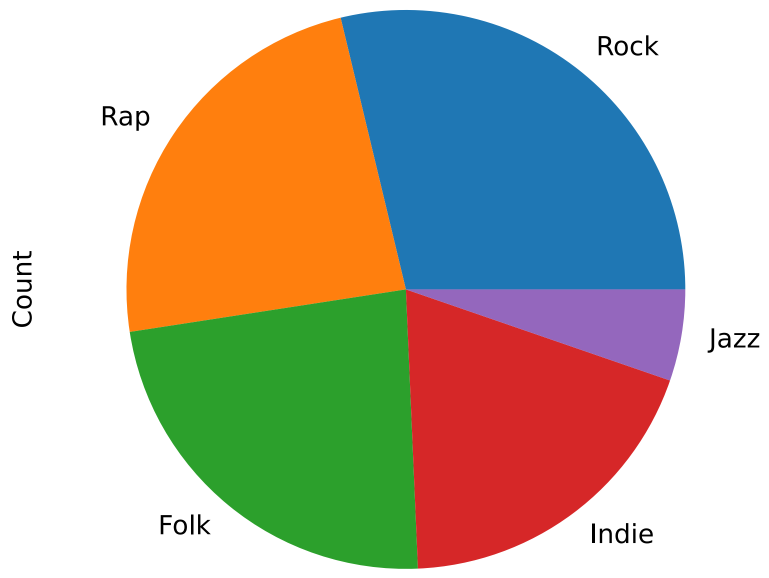


Figure 18: piechart

Figure *piechart*: Barchart created from data from Spotify.

Note that instead of listing both the Categories and the Count as data, we use the categories as index. This gets the proper labeling for our pie chart. In addition, Pandas automatically adds a legend, but this is unnecessary so we can remove the legend by setting the parameter `legend=None` in `plot.pie()`.

3.3.6 Exporting

Note that this is the same as the Matplotlib tutorial found here on Github.

To export your graph as an image file, you can use the Matplotlib function `savefig("fname.x")`. You can specify the file type by filling in `.x` with `.pdf`, `.png`, `svg`, etc.

The parameter `dpi` sets the DPI (Dots per Inch) of the image being saved. Specify this number in the form of a float. For example, set `dpi=300`.

Additionally, there is another way to save files that may be faster than calling a specific method for each file. The following code showcases this:

```
import matplotlib.pyplot as plt
import os
from matplotlib import pyplot


def save():
    name = os.path.basename(__file__).replace(".py", "")
    plt.savefig(f'/filepath/{name}.png')
    plt.savefig(f'filepath/{name}.pdf')
    plt.savefig(f'filepath/{name}.svg')
    plt.show()
```

This code can be accessed on GitHub

The very last command is `plt.show()`, as this command displays the graph that you made. To show, simply type:

```
plt.show()
```

3.3.7 Sources

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.pie.html

4 PYTHON

4.1 Cloudmesh StopWatch

Improve the documentation of Cloudmesh StopWatch if needed

https://github.com/cloudmesh/cloudmesh-common/blob/main/cloudmesh/common/StopWatch.py

Please be reminded that we could develop a program that read the documentation form the docstring Then we can safe it to a file, so we could autogenerate the md file from the docstring.

4.2 cms sys command generate

locate in the book how to use cms sys command generate. Generate a command with your username. No commit of this is necessary, but we need to make sure you understand how to create a command.

4.3 Linux

The book has some introduction material to linux, please contribute to the book. Make sure you do not duplicate wht others have done or are doing, coordinate. Create a pull riquest with your contribution.

4.4 pandas

See if we have already a chapter Pandas in the book and learn about it. Improve or add a new features that you found.

4.5 Python

find a topic that is not yet in the book and create a description for others. Do not duplicate efforts. Create a pull request with your contribution.

4.5.1 Queue

A queue is a data structure. Essentially, it is a list that has order, meaning that the queue has an object to be removed first and an object to be removed last.

There are three types of queues. The first is FIFO, which stands for first in, first out. This means that the first element that is added to the data structure is the first element that is pulled from the data structure. The second type if LIFO, which stands for Last in, First out. This means that the element that was last added to the queue is the element that will be removed first. Finally, there is a priority queue. In a priority queue, the element that is removed first is the element with the lowest value for a priority (meaning the highest priority).

Fortunately, python has a built-in queue module. To access, simply type:

```
import queue
```

Then, the data structures can be built as follows:

4.5.1.1 FIFO Queue

4.5.1.2 LIFO Queue

4.5.1.3 Priority Queue

4.6 FastAPI

FastAPI is a Python framework that allows developers to use the RestAPI interface to call functions that implement applications. RestAPI is used to call the common building block of an application. ## FastAPI Install

There are two ways to install the FastAPI: either completely with the `uvicorn` or partially with both the `FastAPI` and the `uvicorn`.

```
$ pip install "fastapi[all]"
$ pip install "uvicorn[standard]"
```

4.6.1 FastAPI Example

The simplest FastAPI file could look like this:

```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

Copy it in a file called `main.py`.

Start the live server as follows:

```
$ uvicorn main:app --reload

INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to
   ↪ quit)
INFO:     Started reloader process [28720]
INFO:     Started server process [28722]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

There's a line that output something like:

```
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to
   ↪ quit)
```

This line displays the URL where your app is served on your local machine.

Navigate to http://127.0.0.1:8000 in your browser.

The JSON response will appear as:

```
{"message": "Hello World"}
```

- Go to http://127.0.0.1:8000/docs.
- OpenAPI :

Using the `OpenAPI` standard for defining APIs, FastAPI creates a `schema` with all of your APIs.

You can see it directly at: http://127.0.0.1:8000/openapi.json.

It will display a JSON that starts with:

```
{
    "openapi": "3.0.2",
    "info": {
        "title": "FastAPI",
        "version": "0.1.0"
    },
    "paths": {
        "/items/": {
            "get": {
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {
```

4.6.1.1 Step 2: create a FastAPI instance

```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

The variable `app` will be a "instance" of the class `FastAPI` in this case.

This will be the primary point of contact for all API creation.

This is the same app that `uvicorn` refers to in the command:

```
$ uvicorn main:app --reload

INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to
   ↪ quit)
```

If you create your app like:

```
from fastapi import FastAPI

my_awesome_api = FastAPI()


@my_awesome_api.get("/")
async def root():
    return {"message": "Hello World"}
```

And copy it in a file `main.py` then you would call `uvicorn` like:

```
$ uvicorn main:my_awesome_api --reload

INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to
   ↪ quit)
```

4.6.1.2 Step 3: create a path operation

Path refers to the last part of the URL beginning with the first. So, in a URL like: `https://example.com/purple/flc` …the path would be: `/purple/flc`

"When developing an API, the"path" is the primary means of separating "concerns" and "resources."

Operation The term "operation" refers to one of the HTTP "methods."

When creating APIs, these specific HTTP methods are typically used to perform a specific action. Normally, you would use: * POST: to generate data * GET: data retrieval. * PUT: updates data. * DELETE: deletes data. The HTTP protocol allows you to communicate with each path by using one (or more) of these "methods." As a result, in OpenAPI, each HTTP method is referred to as a "operation."

4.6.1.2.1 Define a path operation decorator¶

```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

4.6.1.3 Step 4: define the path operation function

Our "path operation function" is as follows:

path: is /. operation: is get. function: the function that comes after the "decorator" (below [@app.get]("/")).

```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

This is an example of a Python function.

It will be called by FastAPI whenever it receives a GET operation request to the URL "/".

It is an async function in this case.

Instead of async def, you could define it as a normal function:

```
from fastapi import FastAPI

app = FastAPI()

Reference: <https://fastapi.tiangolo.com/tutorial/first-steps/>
@app.get("/")
def root():
    return {"message": "Hello World"}
```

4.6.1.4 Step 5: return the content

```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

You can return a dict, a list, or singular values such as str, int, and so on.

Pydantic models can also be returned.

Many more objects and models will be automatically converted to JSON (including ORMs, etc). Try using your favorites; they are almost certainly already supported.

References: https://fastapi.tiangolo.com/tutorial/first-steps/

5 RIVANNA

5.1 Rivanna

Logging in to Rivanna via web interface

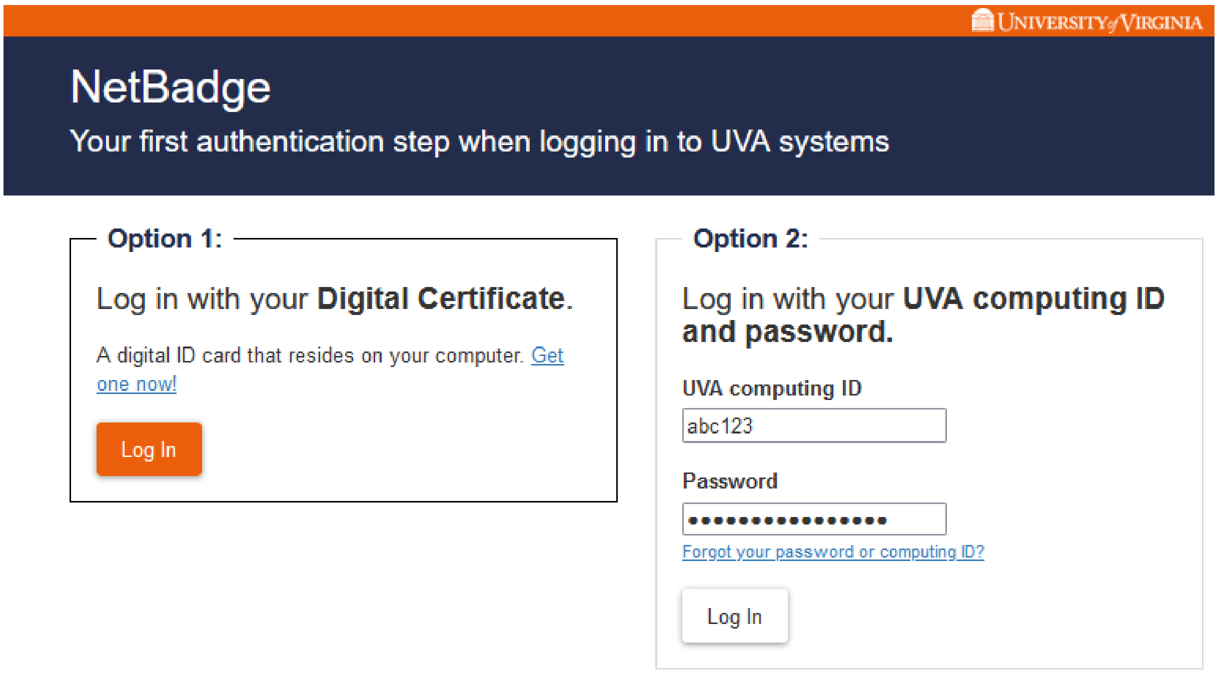
Documentation: https://www.rc.virginia.edu/userinfo/rivanna/login/#web-based-access

Login: https://rivanna-portal.hpc.virginia.edu/

UVA VPN: https://in.virginia.edu/vpn

Shell access: https://rivanna-portal.hpc.virginia.edu/pun/sys/shell/ssh/rivanna.hpc.virginia.edu

JupyterLab: https://rivanna-portal.hpc.virginia.edu/pun/sys/dashboard/batch_connect/sys/jupyter_lab/session_contexts/new


UNIVERSITY of VIRGINIA

NetBadge

Your first authentication step when logging in to UVA systems

Option 1:

Log in with your **Digital Certificate**.

A digital ID card that resides on your computer. Get one now!

Log In

Option 2:

Log in with your **UVA computing ID and password.**

UVA computing ID

abc123

Password

••••••••••••••••

Forgot your password or computing ID?

Log In

Figure 19: UVA Login

Figure: UVA Login

The user must install Duo Mobile on smartphone to use as an authentication service to approve logins.

For security reasons we suggest never saving the password within the browser autofill.

After logging in, you will receive an email through your UVA email inbox to create an account on Rivanna. Once completing the sign-up process, it will take around 1 hour for your account creation to be finalized.

If connecting through SSH, then a VPN is required. Follow the instructions to download UVA Anywhere at the following link: https://in.virginia.edu/vpn

To log in to Rivanna, ensure you are connected to UVA Anywhere and issue the following (make sure you replace `abc123` with your UVA id):

```
you@yourcomputer$ ssh-copy-id abc123@rivanna.hpc.virginia.edu
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
   ↪  to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
   ↪  are prompted now it is to install the new keys
```

```
abc123@rivanna.hpc.virginia.edu's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'abc123@rivanna.hpc.
    ↪ virginia.edu'"
and check to make sure that only the key(s) you wanted were added.

you@yourcomputer$ ssh abc123@rivanna.hpc.virginia.edu
Last login: Tue May 31 11:55:43 2022
Authorized Use Only!
-bash-4.2$
```

5.1.1 Notes: superpod

Estimated deployment for testing by the end of this summer.

Hardware Components:

- 10 DGX-A100 (80GB) Servers (8 GPUs)
- 2 DGX-A100 (40GB) Servers (16 GPUs)
- HDR Infiniband (200GB/s) IB network fabric for GPU-to-GPU direct communication
- 500T ESS3200 pure SSD SpectrumScale (aka GPFS) direct-to-GPU storage array

The SuperPod is a collection of GPU servers (Nvidia DGX-A100) integrated into the Rivanna Cluster (on the GPU partition) with an 200Gb/s IB fabric interconnecting the GPUs with each other and with dedicated temporary storage for Nvidia GPUDirect features. The GPU Direct features allow for very fast transfers between the GPUs, storage and also for larger distributed GPU models.

5.1.2 Special DGX Nodes on Rivanna

DGX A100 (udc-an36-1) is now available for your bii_dsc and bii_dsc_community members to test.

Here is the current status:

- The server is NOT YET integrated into the NVIDIA SuperPod because we are still awaiting networking equipment for implementing the SuperPod. We will be in touch if there is a need for a maintenance outage to integrate the server into the SuperPod.
- There is a RAID0 array of NVMe disks mounted locally at /localscratch. The capacity is 27TB. Please keep in mind that /localscratch is not backed up.

- The server is named udc-an36-1 and is currently in the bii-gpu partition with a permanent reservation named bi_fox_dgx for only bii_dsc and bii_dsc_community allocation members to use. To use this reservation for the A100 node, your researchers and students will have to use the following slurm flags:

```
#SBATCH --reservation=bi_fox_dgx
#SBATCH --account=<enter relevant allocation here>
#SBATCH --partition=bii-gpu
#SBATCH --gres=gpu:<number of GPUs to request>
```

For -account, users will enter either bii_dsc or bii_dsc_community depending on which group they belong to. You can find this by running the allocations utility at the commandline. For -gres=gpu:, users should enter the number of GPUs requested.

The full details of the reservation are below. I named the Slurm reservation “bi_fox_dgx”. It’s not a typo. To change the name of the reservation, I would have to delete the reservation and re-create it and the actual name of the reservation does not affect the reservation’s usability. I’ve successfully tested the ability to use this reservation for all the current bii_dsc and bii_dsc_community members using the Slurm parameters I sent previously.

```
ReservationName=bi_fox_dgx StartTime=2022-06-01T08:37:38 EndTime
   ↪ =2022-06-02T08:37:38 Duration=1-00:00:00
   Nodes=udc-an36-1 NodeCnt=1 CoreCnt=256 Features=(null)
      ↪ PartitionName=bii-gpu Flags=DAILY,SPEC_NODES
   TRES=cpu=256
   Users=(null) Groups=(null) Accounts=bii_dsc,bii_dsc_community
      ↪ Licenses=(null) State=ACTIVE BurstBuffer=(null) Watts=n/a
   MaxStartDelay=(null)
```

5.1.2.1 Starting interactive job on special partition

```
ssh $USERNAME@rivanna.hpc.virginia.edu

$ ijob --reservation=bi_fox_dgx --account bii_dsc --partition=bii-gpu
   ↪  --gres=gpu:1
salloc: Pending job allocation 39263336
salloc: job 39263336 queued and waiting for resources
salloc: job 39263336 has been allocated resources
salloc: Granted job allocation 39263336
salloc: Waiting for resource configuration
```

```
salloc: Nodes udc-an36-1 are ready for job

$ nvidia-smi
Wed Jun  1 17:15:49 2022
+-----------------------------------------------------------------------------+
   ↪
| NVIDIA-SMI 470.103.01   Driver Version: 470.103.01   CUDA Version:
   ↪ 11.4     |
|-------------------------------+----------------------+----------------------+
   ↪
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile
   ↪ Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util
   ↪ Compute M. |
|                               |                      |
   ↪               MIG M. |
|===============================+======================+======================|
   ↪
|   0  NVIDIA A100-SXM...  Off  | 00000000:07:00.0 Off |
   ↪                    0 |
| N/A   29C    P0    54W / 400W |     85MiB / 81251MiB |      0%
   ↪      Default |
|                               |                      |
   ↪ Disabled |
+-------------------------------+----------------------+----------------------+
   ↪

+-----------------------------------------------------------------------------+
   ↪
| Processes:
   ↪
   ↪ |
|  GPU   GI   CI        PID   Type   Process name
   ↪ GPU Memory |
|        ID   ID
   ↪ Usage      |
|=============================================================================|
   ↪
```

```
|    0   N/A  N/A     13486      G   /usr/bin/X
   ↪                             63MiB |
|    0   N/A  N/A     13639      G   /usr/bin/gnome-shell
   ↪                   21MiB |
+-----------------------------------------------------------------------------+
   ↪
```

5.1.3 SSH Config

```
$ cat ~/.ssh/config
host rivanna
        User <USERNAME>
        HostName rivanna.hpc.virginia.edu
        IdentityFile ~/.ssh/id_rsa
```

5.2 Run Python MPI programs on Rivanna

see the book Python MPI

add chapter if not there

6 AI

6.1 DL Timeseries

craete and documenta simple time series command

use cloudmesh sys command generate to create an extension so we can do a commandline tool

```
cms timeseries --config=CONFIG
```

where the config file is a yaml file.

Work with Gregor as he will create a separate github repo for this and create the command template so you can fill it out with content.

7 BIOS

7.1 Bios

Please add here a 2-3 paragrapgh professional Bio. Look up who a professional bio is isn IEEE papers. Write in 3rd person. TOD: provide link example.

Review other peoples bios and improve or give improvement tips where needed.

If it turns out you never contributed to anything, your bio will be removed (as well as your name in this proceedings).

7.1.1 Paul Kiattikhunphan

Paul Kiattikhunphan is a second year at the University of Virginia majoring in computer science.

7.1.2 Alex Beck

Alex Beck has completed his first year at the University of Virginia where he is majoring in electrical engineering. He is set to receive his Bachelor of Science degree in the Spring of 2025. He currently maintains a 3.9 GPA.

Alex is currently working in research this summer at the UVA Biocomplexity Institute's Computing for Global Challenges program under Dr. Gregor Von Laszewski and Dr. Geoffrey Fox where he is planning to gain experience in programming and data science. Prior to that, he has previous experience in sales from working in retail.

At UVA, Alex is involved in a few extracurricular organizations. He is currently active in the Virginia Eta Chapter of Sigma Phi Epsilon, the UVA Climbing Team, and the UVA Chapter of the QuestBridge Scholars Network.

7.1.3 Alison Lu

Alison Lu has completed her second year (Class of 2024) at the University of Virginia pursuing a double major in CS and Chemistry with a minor in Japanese. She is conducting research with the physics department studying quantum computing and photon resolution using machine learning. In addition, she works with UVA's Repair Lab to study gentrification in Norfolk, VA.

She is currently participating in the Biocomplexity Institute's C4GC REU program. Her interests include computer architecture, machine learning, and quantum computing alongside quantum mechanics.

7.1.4 Jackson Miskill

Jackson Miskill has completed his second year at the University of Virginia where he is studying Computer Science and Cognitive Science. He will receive a Bachelor of Arts degree from UVa in Spring of 2024. Jackson has studied python and java in his courses, delving into concepts from basic syntax to data structures and algorithms.

Jackson is currently working at the UVa Biocomplexity Institute under Dr. Gregor von Laszewski as a part of the Computing for Global Challenges program. He is studying the intersection between python and cloud computing. In the future, Jackson plans to continue research.

7.1.5 Jacques Fleischer

Figure 20: Jacques's Picture

Jacques Fleischer is a sophomore at the Miami Dade Honors College. He is set to receive his associate degree in computer science in summer 2022. He received the Miami Dade Honors College Fellows Award and currently maintains a 4.0 GPA on the Dean's List.

In the summer of 2021, he participated in the Florida-Georgia Louis Stokes Alliance for Minority Participation REU Data Science and AI Research Program; his research focused on predicting the price of cryptocurrency using artificial intelligence. This was done in conjunction with faculty from Florida A&M University and Indiana University. He presented his findings at the Miami Dade College School of Science Symposium in October 2021. Jacques was accepted to the 2022 Emerging Researchers National (ERN) Conference in STEM after applying with his abstract on cryptocurrency time-series. Additionally, he was one of four Miami Dade College students to be nominated for the Barry Goldwater Scholarship due to his research findings.

Jacques is active in extracurriculars; for instance, he is the current Vice President of the MDC Computer Club. There, he hosts virtual workshops on how to use computer software, including Adobe Premiere

Pro and PyCharm. He is also a member of Phi Theta Kappa. Furthermore, he is an active contributor to Cloudmesh: an open-source, all-in-one grid-computing solution written in Python. He presently participates in the University of Virginia's Computing for Global Challenges program with Dr. Gregor von Laszewski and Dr. Geoffrey C. Fox to find high performance computing solutions using Raspberry Pis.

7.1.6 Eric He

Junyang (Eric) He completed his first year of study at the University of Virginia majoring in Computer Science. He anticipates to receive his B.S. in Computer Science in 2025. He is currently a member of the Engineering Student Council and the Chinese Student and Scholars Society at UVA.

In the summer of 2021, Eric worked as a Data Analyst intern at Nint (Shanghai) Co., Ltd, a company that provides market data analysis products for E-commerce His work included time series data cleaning and natural language processing.

Eric is currently conducting research with Prof. Geoffrey C. Fox on a Deep Learning model based on LSTM networks trained to predict hydrological features like streamflow, precipitation, and temperature at different locations in the US. He focused primarily on the possibilities of extending the model to countries outside of the US such as Chile and UK.

7.1.7 Abdulbaqiy Diyaolu

AbdulBaqiy Diyaolu is a Computer science and Mathematics Major from Mississippi Valley State University. He will be receiving his bachelor's degrees in both Computer science and Mathematics in the year 2025. AbdulBaqiy is currently a presidential scholar at Mississippi Valley State University and he maintains a 4.0 GPA.

AbdulBaqiy currently works at Fedex Logistics at MVSU. He helps in data entry and data Analysis. He is hoping to polish his data analysis skills with this opportunity. In the summer of 2022, he joins the Bio complexity research program at UVA where he will be able to use his skills in support of different researches, and also learn more research skills along the way.

Abdulbaqiy participates in several extracurricular activities in MVSU he is a member of African Student Union(ASU), National Society of Black engineers (NSBE), and the google developer's club. He is also a Strada scholar at MVSU where he participates in several leadership development activities.

8 REFERENCES