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.

2 GRAPH VIZUALIZATION

2.1 Python Graphics

TODO: Remove contractions (I, you, we, etc.) TODO: Grammarly for sections

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

2.1.1 Matplotlib

Matplotlib is a library that allow 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.

2.1.1.1 Installation To install matplotlib, please use the command:

```
$ pip install matplotlib
```

2.1.1.2 Bar Chart In matplotlib it is easy to create bar charts. We demonstrate a simple example using data from a user from 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='edge',
        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*.

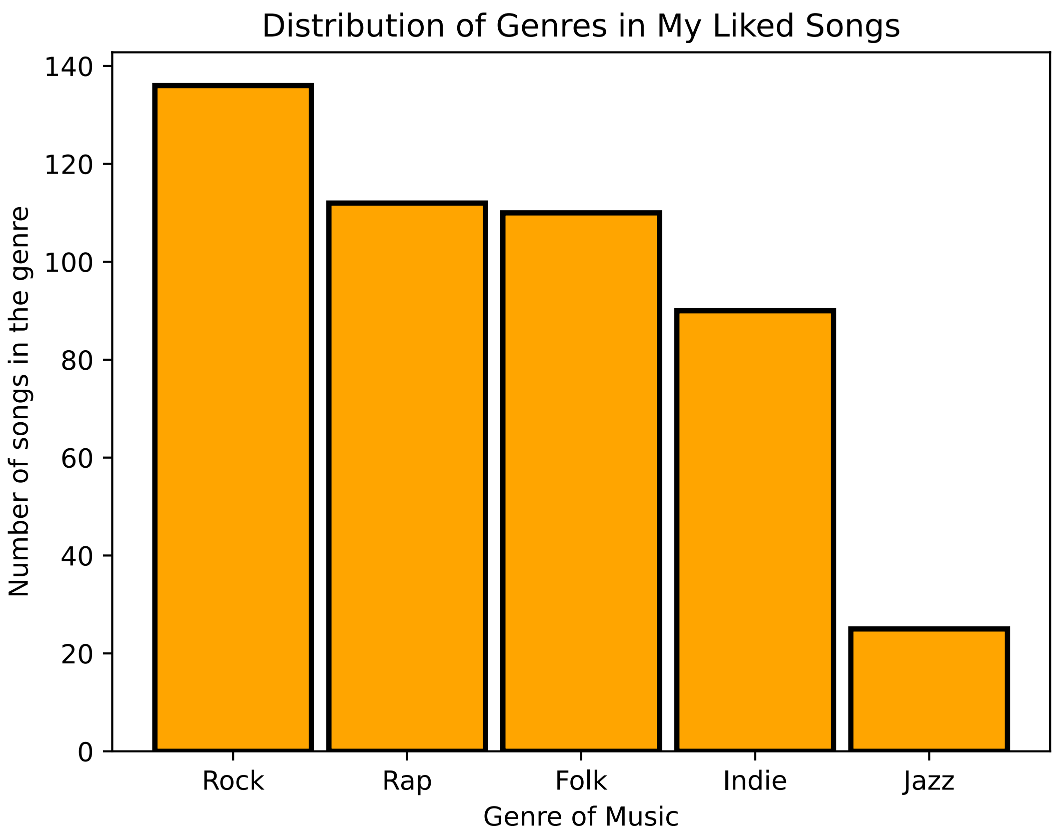


Figure 1: barchart

Figure *barchart*: Barchart created from data from Spotify

The bar chart is a graph that visualizes data by displaying the quantity of several variables through different sized rectangles. Matplotlib essentially creates the bar chart object as a figure, and then displays that figure on the computer. plt.barchart takes in a multitude of parameters.

2.1.1.3 Line Plot The matplotlib library in python allows for comprehensive line plots to be created. Here we created a line plot 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.

```
x = []
for i in range(0, 100):
    value = random.random() * 10000
    x.append(value)
```

```
# creating a list of 100 numbers in order from 0 to 100
y = []
for j in range(0, 100):
    y.append(j)

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

The line chart using the matplotlib library allows for multiple data sets to be contrasted against each other in the same graph. The line chart is positioned based on the np.linspace which takes in two total parameters that determine the starting point and the end point and an optional parameter that defines the total generated sample between the start/end points. With this, the variables and data sets can be plotted and modified to shape the steepness of the curve and its growth rate–exponential, polynomial, logarithmic, s-curved etc.

```
x = np.linspace(start,end,samples between start-finish)
plt.plot(x, y)
```

Instead of only plotting a linear line, there is a choice to include multiple points of x-values that relate to its corresponding y-values. As mentioned, functions can be incorporated to adjust the line's properties either by addition, subtraction, division or multiplication. Optional parameters for better visualization in a line chart includes the modification of the linestyle, and can be adjusted based on the viewer's preference–dotted line, dashed, dashed with dots or none).

2.1.1.4 Pie Plot A pie plot 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 `pie()` function creates a pie plot. In the following code example, a user's Spotify data will be displayed as a pie plot.

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

The pie chart is a graph that visually displays multiple quantities of data as a proportion to the total amount, represented as the whole circle, with each quantity shown as a proportional slice of it. Matplotlib has the ability to display data through a pie chart as a figure after data is inputted. The command plt.pie takes in many parameters. Here are some of the parameters used in plt.pie, from matplotlib API online, not all of them are shown here.

```
plt.pie(x, labels, colors, normalize, startangle, radius, center)
```

Here, the first parameter `x` is the parameter that consists of the data being plotted, which should be in the form of a list or dictionary as it be multiple quantities of data. Each slice of the pie can be labeled. To do so, labels must be in the form of a list of strings in the same corresponding order as the data. The sequence of colors of the slices can be set using the command `plt.get_cmap("Colors")`. There is also the choice of making the pie chart a full pie or not using normalize. Setting it to True, which is the default, makes it a full pie, False makes it not a full pie. The angle of the start of the pie, set counterclockwise from the x-axis can be set using startangle. The radius of the pie can be set using radius and setting it to a float. The coordinates of the center of the chart can be set in the form `(float, float)`.

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

```
#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
fig, ax = plt.subplots()
ax.contour(x,y,z, levels=levels)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Function of z(x,y)")
plt.show()
```

A contour plot allows data and equations consisting of three variables to be plotted through plotting 3D surfaces as 2D slices on an xy plane. Matplotlib has the ability to 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.

2.1.2 Titles, Labels, and Legends

2.1.2.1 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").
```

2.1.2.2 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")
```

2.1.2.3 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()
```

2.1.2.4 Display The very last command you should put in your code is plt.show(), as this command displays the graph that you made. To show, simply type:

```
plt.show()
```

2.1.3 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.

2.1.3.1 Installation

2.1.3.2 Import Statements

2.1.3.3 Bokeh Plotting Interface Bokeh.plotting is the library's main interface. It allows you to generate plots easily by providing parameters such as axes, grids, 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)
```

2.1.3.4 Some useful parameters from figure TODO: change section header

- 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)

2.1.3.5 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, you will be able to create different types of plots utilizing the figure created with the figure function.

2.1.3.6 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.

```
# Circle
p.circle([1,2], [3,4], size = 10)

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

The list of all possible marker types and the functions used to create them can be found here: http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html

2.1.3.7 Line Graphs

TODO: Make line graphs/plots a consistent term

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. You can create a simple linear line graph connecting the points (1,1), (2,2) and (3,3) with the following.

```
# The line_width parameter sets the width of the line graph.
x = [1,2,3]
y = [1,2,3]
p.line(x, y, line_width = 1)
```

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

2.1.3.8 Bar Graphs

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

```
# The line_color parameter sets the color of the bar graph.
x = [1,2,3]
y = [1,2,3]
p.hbar(x, y, line_color = 'black')
```

2.1.4 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 dataframes (generated from Pandas) that are passed into its functions as the data.

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

We have created examples to demonstrate the abilities of Seaborn. These examples draw on a GitHub repository made by the creator of Seaborn (listed in the sources section). The data ("mpg") used looks at different variables in different cars such as displacement, horsepower, miles per gallon, etc.

2.1.4.1 Installation

Seaborn can be installed in the exact 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
```

2.1.4.2 Import Statements

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

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

It is easy to set up the data to be used because the `load_dataset` method draws directly from GitHub:

```
data = sns.load_dataset("mpg")
sns.set_theme()

dependent_1 = data.horsepower
dependent_2 = data.mpg

independent_1 = data.displacement
independent_2 = data.weight
independent_3 = data.acceleration

hue_1 = data.origin
hue_2 = data.model_year
```

2.1.4.3 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. A hued line plot can be created easily with Seaborn.

```
sns.lineplot( x=independent_1 , y=dependent_1, hue=hue_1)
plt.show()
```

Which produces:

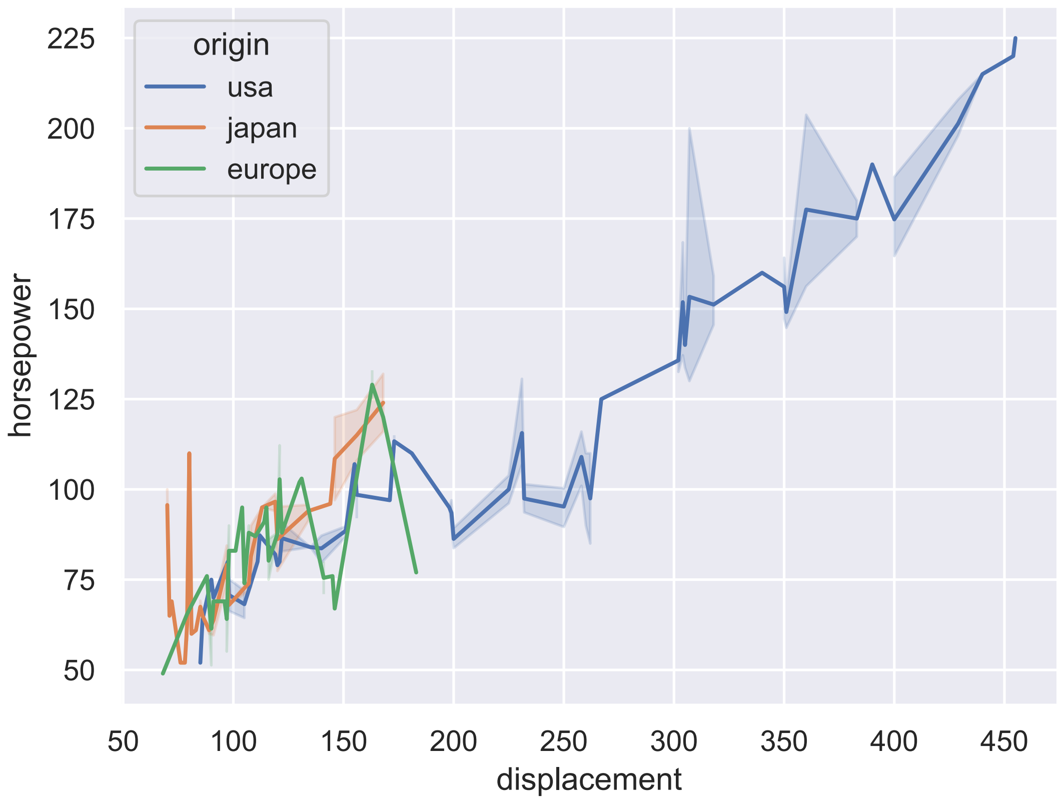


Figure 2: lineplot

2.1.4.4 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 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=independent_2, y=dependent_2, hue=hue_1)
plt.show()
```

Which produces:

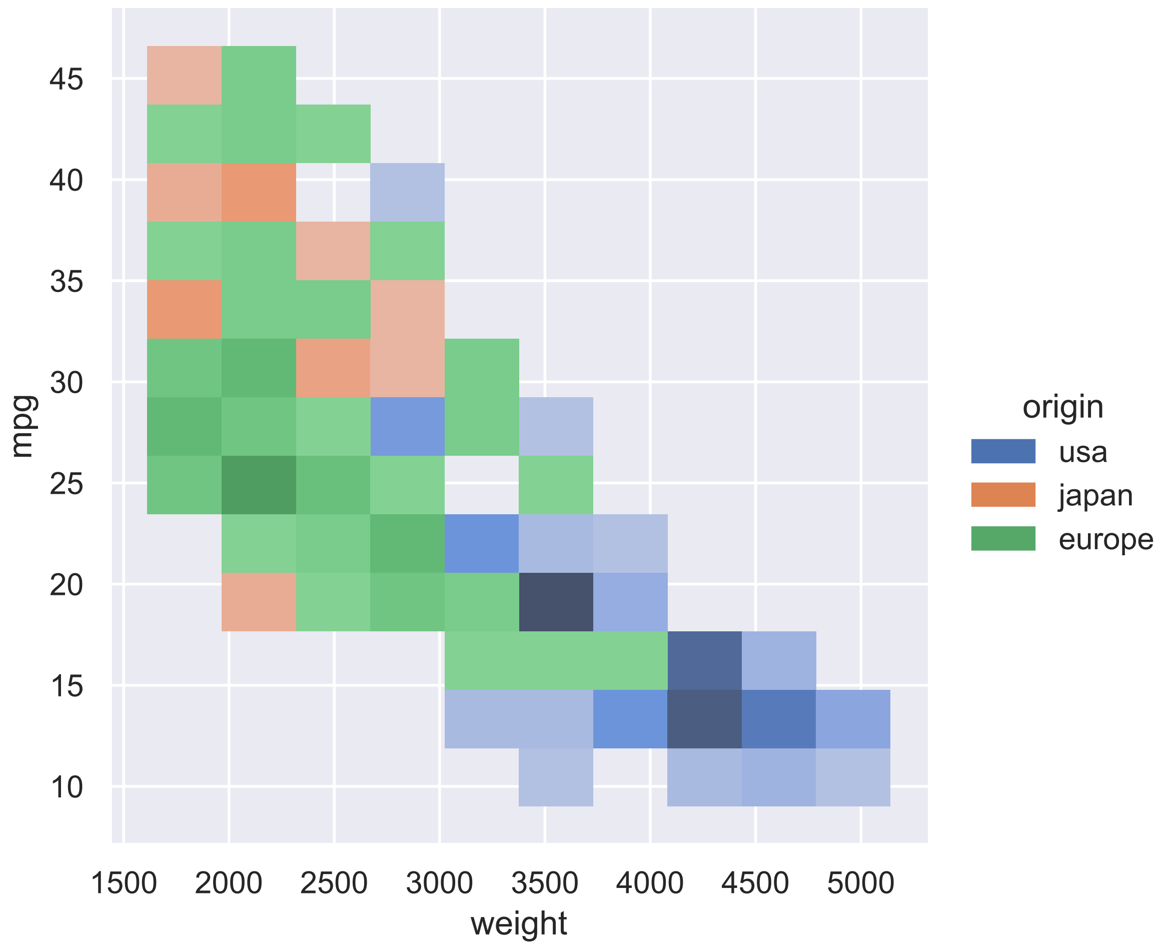


Figure 3: displot

2.1.4.5 Categorical Plots Categorical plots are statistical graphs that help visualize 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. It is necessary to include the `kind` parameter as it specifies the type of categorical plot that will be created.

```
sns.catplot(x="displacement", data=data, kind="count")
plt.show()
```

Which produces:

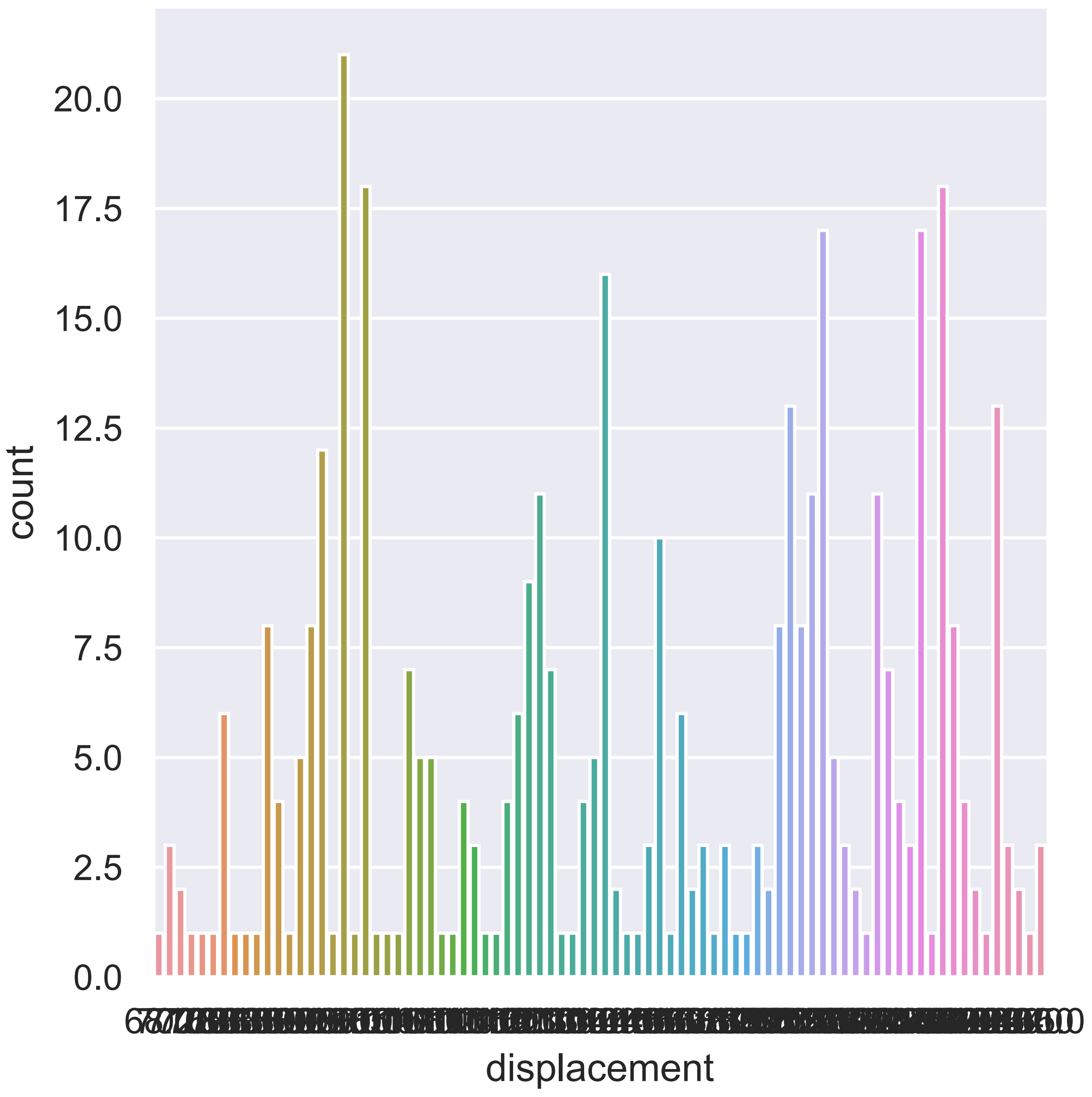


Figure 4: catplot

2.1.4.6 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. Ther regression plots are `lmplot` `regplot` and `residplot`.

Regression plots are simple to implement:

```
sns.regplot(x=independent_1, y=dependent_1)
```

```
plt.show()
```

Which produces:

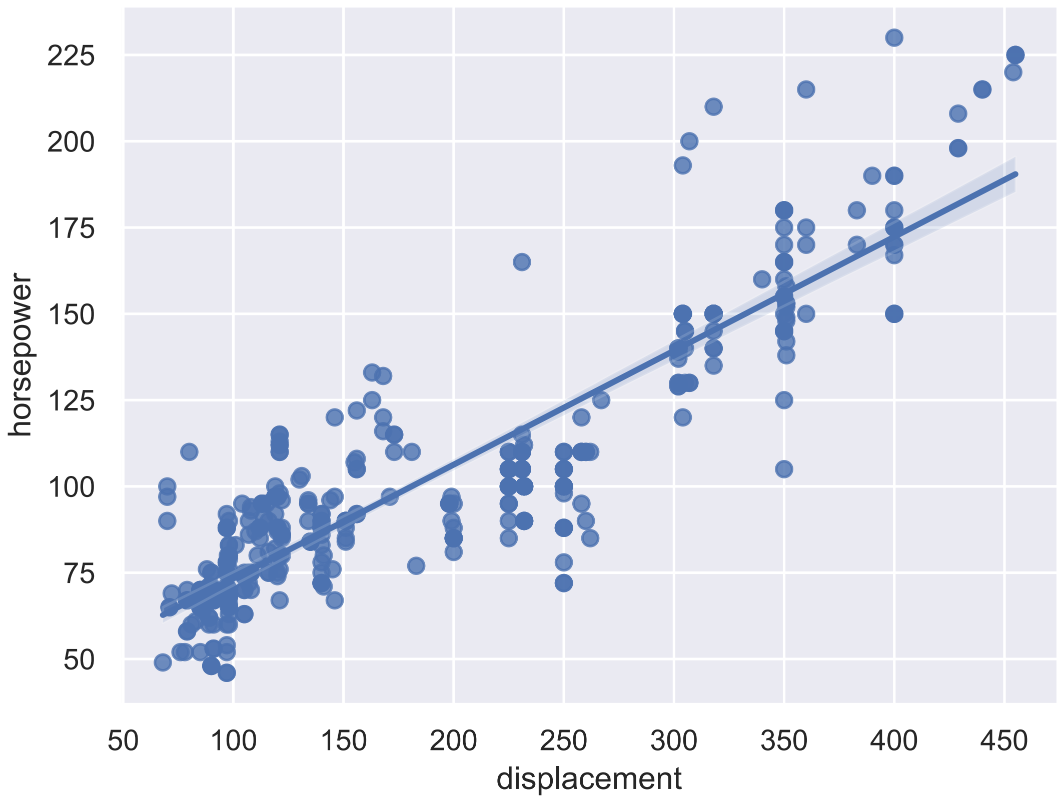


Figure 5: regplot

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

2.1.4.7 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')
```

2.1.5 Sources

2.1.5.1 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/

2.1.5.2 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

2.1.5.3 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

3 INSTALL

3.1 Install

Improve the instalation instructions for python in the book.

3.2 Installing Packages with Choco

3.2.1 emacs

TODO:

3.2.2 make

TODO:

3.3 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}`')
```

3.3.1 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.

3.3.2 windows

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

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.

5 RIVANNA

5.1 Rivanna

Logging in to Rivanna via web interface

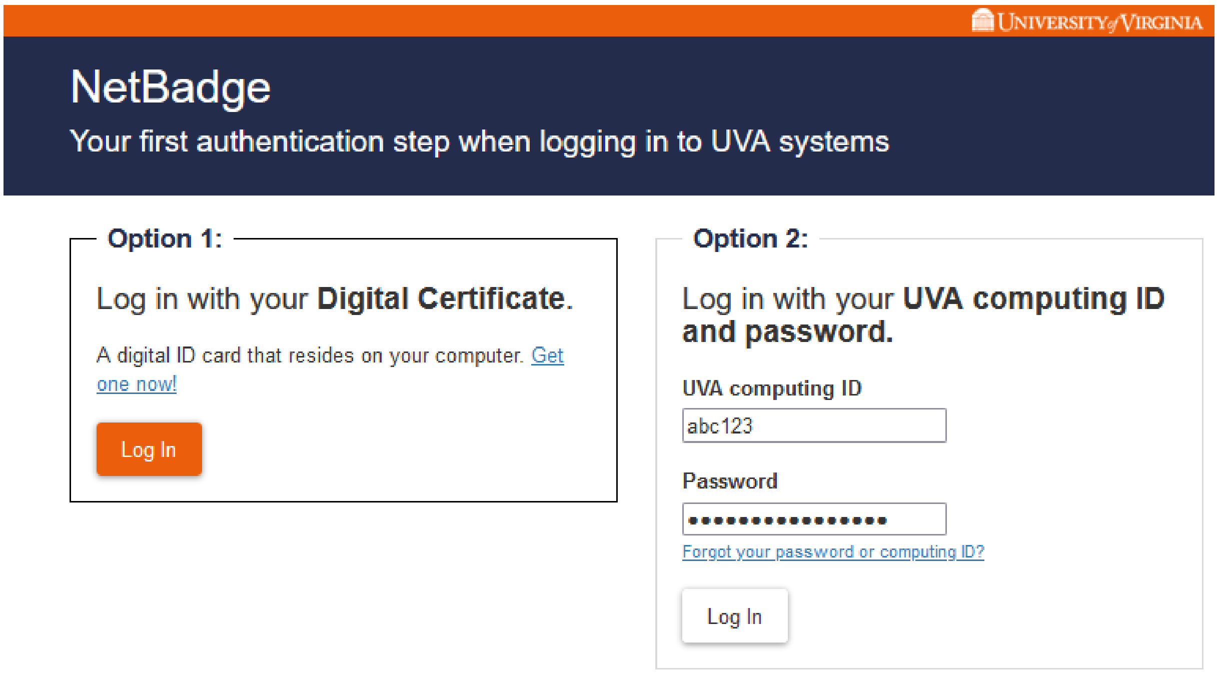
Documentation: https://www.rc.virginia.edu/userinfo/rivanna/login/#web-based-access

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



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 6: 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 BIOS

6.1 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).

6.1.1 Paul Kiattikhunphan

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

6.1.2 Alex Beck

Alex is a first year student at the University of Virginia majoring in electrical engineering.

6.1.3 Alison Lu

Alison Lu has completed her second year at the University of Virginia pursuing a double major in CS and Chemistry with a minor in Japanese. She is currently participating in research.

6.1.4 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 worked in a variety of different environments: restaurant, coffee shop, environmental science organization, and nonprofit. He is currently working in research at the UVA Biocomplexity Institute where he is planning to learn as much as possible about programming, data science, and academic research.

At UVA, Jackson is active in organizations that he is passionate about. He volunteers for the National Alliance on Mental Illness in the outreach committee, served as recruitment chair for Phi Delta Theta and treasurer of Phi Alpha Delta, and volunteers with Madison House in the Casa Alma organization.

6.1.5 Jacques Fleischer

Figure 7: 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.

6.1.6 Eric He

TBD

6.1.7 Abdulbaqiy Diyaolu

AbdulBaqiy Diyaolu is a Computer science and Mathematics Major from Mississippi Valley State University.

7 REFERENCES