My Data Visualization Portfolio

Quarto, Virtual Environments & Data Visualization

Tech Setup Challenge - Part 2: Quarto & Python Environment

Prerequisites

Before starting Part 2, make sure you've completed Part 1:

- ☐ Cursor AI installed and configured
- ☐ Git set up in Cursor AI (Source Control)
- ☐ GitHub account created
- ☐ Course materials forked and cloned
- ☐ Student folder created with README.md
- ☐ Initial pull request submitted

Part 2: Quarto & Python Environment Setup

Why Quarto?

Quarto is essential for creating **professional reports** that combine:

- **Text** (like this document)
- **Code** (Python/R analysis)
- **Visualizations** (charts, graphs)
- **Interactive elements** (dashboards)

It's not just for code - it's for creating beautiful, reproducible documents that are perfect for business presentations and reports.

1. Install Quarto

- Go to quarto.org/docs/get-started/
- Download and install Quarto for your operating system
- Verify installation: Open terminal/command prompt and type `quarto --version`
- **Recommended:** Install the Quarto extension in Cursor AI for better editing experience:
  - Press `Ctrl+Shift+X` to open Extensions (from within Cursor AI)
  - Search for "Quarto"
  - Install the "Quarto" extension by Quarto
  - **Why?** This gives you syntax highlighting, live preview, and better editing support for `.qmd` files

2. Create Your Own GitHub Repository

Why Create a Separate Repository?

Creating your own repository for Quarto projects is a best practice that: - Keeps your work organized and separate from course materials - Allows you to showcase your portfolio publicly - Teaches you proper project management - Enables GitHub Pages for hosting your documents

Create a New Repository on GitHub:

1. **Go to GitHub.com** and make sure you're logged in
2. **Click the "+" icon** in the top right corner → "New repository"
3. **Repository settings:**
   - **Repository name:** `your-nickname-quarto-portfolio` (e.g., `johnD-quarto-portfolio`)
   - **Description:** "My Quarto data visualization portfolio"
   - **Visibility:** Public (so you can use GitHub Pages)
   - **Initialize with:** Check "Add a README file"
4. **Click "Create repository"**

3. Clone Your New Repository

Clone to Your Computer:

1. **In Cursor AI:** Press `Ctrl+Shift+P` → "Git: Clone"
2. **Paste your repository URL:** `https://github.com/YOUR_USERNAME/your-nickname-quarto-portfol`
3. **Choose location:** Save it somewhere easy to find (like `C:\` or your Documents folder)
4. **Open the cloned repository** in Cursor AI

4. Set Up Python Virtual Environment

Why Virtual Environments? Virtual environments keep your Python packages organized and prevent conflicts between different projects. Each project can have its own set of packages without interfering with others.

Create a Virtual Environment:

1. **Open Terminal in Cursor AI:**
   - Press 'Ctrl + " (backtick key, usually under the Esc key) to open the integrated terminal
   - OR go to the top menu: **Terminal → New Terminal**
2. **You should already be in your repository folder** (if not, navigate to it)
3. **Create Virtual Environment:**

   ```
   python -m venv venv
   ```

   - This creates a folder called `venv` with a clean Python environment
4. **Activate Virtual Environment:**

   First, identify your terminal type:

   - **Command Prompt (cmd):** Prompt shows `C:\>` or similar with backslash paths
   - **PowerShell:** Prompt shows `PS C:\>` with "PS" at the beginning
   - **Git Bash:** Prompt shows `user@computer MINGW64` or similar
   - **Mac/Linux Terminal:** Prompt shows `user@computer:~$` or similar

   Then use the appropriate command:

   - **Windows Command Prompt (cmd):**

     ```
     venv\Scripts\activate
     ```

   - **Windows PowerShell:**

     ```
     .\venv\Scripts\Activate.ps1
     ```

     - **Note:** If you get an execution policy error, run: `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`
   - **Windows Git Bash:**

     ```
     source venv/Scripts/activate
     ```

- **Mac/Linux Terminal:**

```
source venv/bin/activate
```

Success indicator: You should see `(venv)` at the beginning of your terminal prompt

5. **Install Required Packages:**

```
pip install matplotlib pandas jupyter
```

6. **Verify Installation:**

```
python -c "import matplotlib, pandas, jupyter; print('All packages installed successfull
```

- You should see: "All packages installed successfully!"
- If you get an error, try: `pip install --upgrade pip` then reinstall the packages

5. Configure Cursor to Auto-Activate Environment

Important: Keep your terminal open with the virtual environment activated while you complete the next steps.

Set Python Interpreter (Project-Specific):

1. **Make sure you have your repository folder open** in Cursor (not just a single file)
2. Press `Ctrl+Shift+P` to open the command palette
3. Type "Python: Select Interpreter"
4. Choose the interpreter from your virtual environment:
   - **Windows:** Look for `.\venv\Scripts\python.exe`
   - **Mac/Linux:** Look for `./venv/bin/python`
   - **Important:** Make sure it shows the path to YOUR project's `venv` folder
5. **Why?** This tells Cursor to use your virtual environment's Python and packages for THIS project only

Pro Tip: Cursor will remember this setting for this specific project folder. When you open this repository in the future, it will automatically use the correct virtual environment!

Verify Setup:

- Open a new terminal in Cursor
- You should see `(venv)` in the prompt automatically
- Type `python -c "import matplotlib; print('Matplotlib installed successfully!')"`

6. Install Live Preview Extension

Why Live Preview?

Live Preview allows you to see your HTML documents update in real-time as you make changes, making development much faster and more interactive.

Install the Extension:

1. **Open Extensions:** Press `Ctrl+Shift+X` in Cursor
2. **Search for:** "Live Preview"
3. **Install:** "Live Preview" by Microsoft
4. **Why?** This extension lets you preview HTML files directly in Cursor with live updates

7. Create Your First Data Visualization

Create a Quarto Document with Matplotlib:

1. **Create a new file** in your repository called `index.qmd` (this will be your main portfolio page)
2. **Add this content** (replace bracketed text with your nickname):

> **i** YAML Front-Matter: The Secret Sauce
>
> **What you're about to see** (between the `---` lines) is called **YAML front-matter** - and yes, YAML stands for "YAML Ain't Markup Language" (because the original acronym was "Yet Another Markup Language" and they got meta with it!).
> Think of it as the **control panel** for your Quarto document - it's where you tell Quarto how to dress up your content. Want HTML? PDF? Should code run or just show? It's all configured here in this little language-within-a-language that makes your markdown document go from "meh" to "WOW!"

Copy and paste this content into your `index.qmd` file:

Welcome to My Portfolio!

Hello! My nickname is [YourNickNameLastInitial]. This is my Quarto data visualization portfolio showcasing my Python and matplotlib skills.

About This Portfolio

This portfolio demonstrates how Quarto can seamlessly integrate Python code and visualizations into professional documents. Each chart below is created using matplotlib and embedded directly into this document.

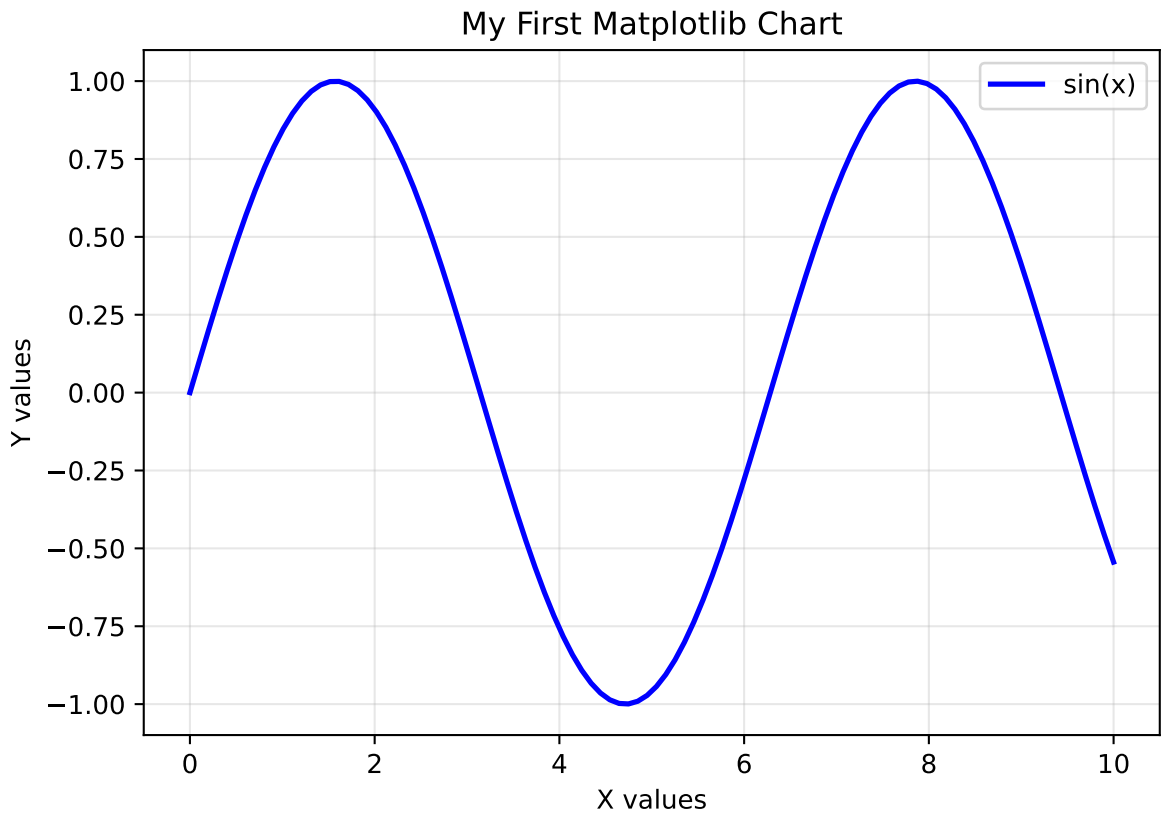
Simple Line Chart

Let's create a simple line chart showing some sample data:

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

# Create sample data
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create the plot
plt.figure(figsize=(7, 4.75))
plt.plot(x, y, 'b-', linewidth=2, label='sin(x)')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.title('My First Matplotlib Chart')
plt.grid(True, alpha=0.3)
plt.legend()
plt.show()
```



Bar Chart Example

Here's a bar chart showing some sample data:

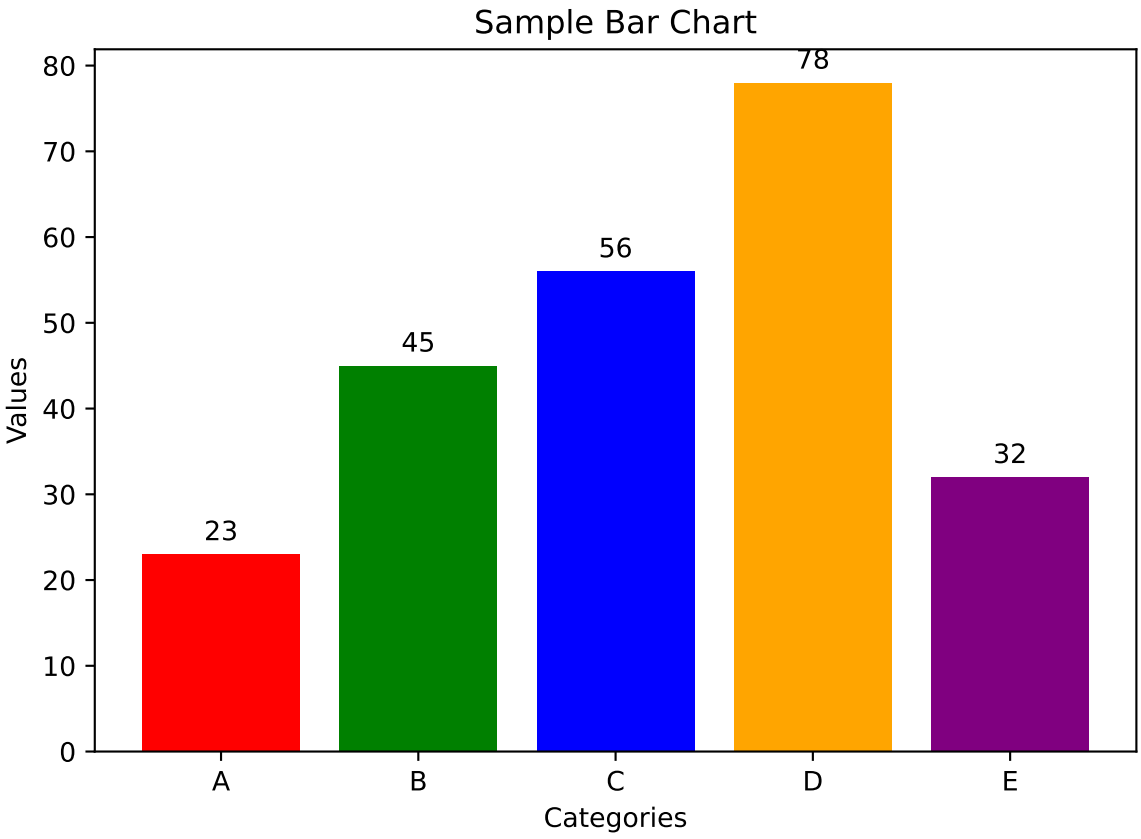
```
# Sample data for bar chart
categories = ['A', 'B', 'C', 'D', 'E']
values = [23, 45, 56, 78, 32]

plt.figure(figsize=(7, 4.75))
bars = plt.bar(categories, values, color=['red', 'green', 'blue', 'orange', 'purple'])
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Sample Bar Chart')

# Add value labels on bars
for bar, value in zip(bars, values):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 1,
```

```
            str(value), ha='center', va='bottom')

plt.show()
```



Conclusion

This demonstrates how Quarto can seamlessly integrate Python code and visualizations into professional documents that can be hosted publicly on GitHub Pages!

3. **Save the file:** Press `Ctrl+S` to save your new `index.qmd` file
4. **Experience Cursor Magic:** Let's enhance your document:
   - **Open the AI Chat:** Press `Ctrl+L` to open Cursor's AI chat panel
   - **Ensure correct context:** Make sure your `index.qmd` file is open and active
   - **Ask Cursor to add content:** Try prompts like:
     - "Add a scatter plot showing random data points"

- "Add a section explaining what matplotlib is and why it's useful"
- "Add a pie chart with sample data"

- **Cursor will add the content directly to your file!**

8. Render Your Document

Render the Quarto Document:

1. **Make sure your virtual environment is activated** (you should see `(venv)` in terminal)
2. **Pro Tip - Open Terminal in the Right Location:**
   - **Right-click on your repository folder** in the file explorer (left sidebar)
   - **Select "Open in Integrated Terminal"**
   - This automatically opens a terminal at the correct path - no need to navigate manually!
   - **Alternative:** Make sure you're in your repository folder (you should see your `index.qmd` file when you type `ls` or `dir`)
3. **Render the document:**

   ```
   quarto render index.qmd
   ```

4. **You should see:**
   - Output indicating the file was rendered successfully
   - A new `index.html` file created in your folder

9. Preview Your Website Locally

Using Live Preview Extension:

1. **Right-click on `index.html`** in the file explorer
2. **Select "Open with Live Server"** (this should appear if Live Preview is installed)
3. **Your browser will open** showing your portfolio with live updates
4. **Make changes to `index.qmd`** and re-render to see updates

Alternative Method:

- **Right-click on `index.html`** → "Open with" → "Live Preview"
- Or use the Live Preview icon in the activity bar

Troubleshooting Preview Issues:

- If Live Preview doesn't work, try opening `index.html` directly in your browser
- Make sure you've rendered the document first with `quarto render index.qmd`
- If the page looks broken, check that your virtual environment is activated

10. Commit and Push Your Work

Save Your Progress:

1. **Commit your work:**
   - Press `Ctrl+Shift+G` → click "+" next to `index.qmd` and `index.html`
   - Write commit message: "Add Quarto portfolio with matplotlib visualizations"
   - Click "Commit" (icon)
2. **Push to GitHub:**
   - Click "Sync Changes" or "Push" to upload your work to GitHub
   - Your repository now contains your portfolio files

11. Enable GitHub Pages

Make Your Portfolio Publicly Accessible:

1. **Go to your GitHub repository** in your web browser
2. **Click "Settings"** tab (at the top of your repository page)
3. **Scroll down to "Pages"** in the left sidebar
4. **Under "Source":**
   - Select "Deploy from a branch"
   - Choose "main" branch
   - Select "/ (root)" folder
5. **Click "Save"**
6. **Wait 2-3 minutes** for GitHub to build your site
7. **Your portfolio will be available at:** `https://YOUR_USERNAME.github.io/your-nickname-quarto-po`

12. Share Your Portfolio

Your portfolio is now live! You can:

- **Share the URL** with others to showcase your work
- **Add more Quarto documents** to your repository
- **Update your portfolio** by editing `index.qmd`, re-rendering, and pushing changes
- **Use this as a foundation** for future data visualization projects

Congratulations! You're Done!

You've completed Part 2 of the tech setup challenge! You now have:

- A professional GitHub repository for your portfolio
- A complete Python environment with Quarto
- A publicly accessible website showcasing your data visualizations
- Experience with modern development workflows

Part 2 Submission Checklist

☐ Quarto installed and verified
☐ Quarto extension installed in Cursor AI (recommended)
☐ Live Preview extension installed in Cursor AI
☐ New GitHub repository created for portfolio
☐ Repository cloned to local computer
☐ Python virtual environment created in repository
☐ Virtual environment activated and packages installed (matplotlib, pandas, jupyter)
☐ Cursor AI configured to use virtual environment Python interpreter
☐ Portfolio Quarto document (`index.qmd`) created
☐ Matplotlib charts added to Quarto document (line chart and bar chart)
☐ AI-generated content added using Cursor AI
☐ Quarto document rendered to HTML successfully
☐ Local preview tested using Live Preview extension
☐ All files committed and pushed to GitHub
☐ GitHub Pages enabled for public hosting
☐ Portfolio URL shared and verified working

What You've Learned

Technical Skills:

- **Repository Management** = Creating and managing your own GitHub repositories
- **Virtual Environments** = Isolated Python environments for clean package management
- **Quarto** = Professional document creation with embedded code and visualizations
- **Matplotlib** = Python library for creating charts and graphs
- **Live Preview** = Real-time HTML preview during development
- **GitHub Pages** = Free web hosting for static sites
- **Cursor AI Integration** = Using AI to enhance your coding and documentation

Workflow Skills:

- **Project Organization** = Proper repository structure and file management
- **Environment Management** = Setting up and managing Python environments
- **Document Rendering** = Converting Quarto documents to HTML
- **Version Control** = Committing and pushing changes to GitHub
- **Web Development** = Creating and hosting professional websites
- **AI-Assisted Development** = Using Cursor AI to enhance your work

Troubleshooting Common Issues

Quarto Installation Issues

- **If `quarto --version` doesn't work:** Make sure Quarto is in your system PATH. Restart your terminal/command prompt after installation.
- **If rendering fails:** Check that your virtual environment is activated and packages are installed.

Virtual Environment Issues

- **If activation fails:** Make sure you're in the correct directory and the `venv` folder exists.
- **If packages won't install:** Try `pip install --upgrade pip` first, then reinstall packages.
- **If Python interpreter not found:** Make sure Python is installed and accessible from command line.

PowerShell Issues

- **Execution policy error:** Run `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser` in PowerShell as Administrator.
- **If activation script doesn't work:** Try using Command Prompt instead of PowerShell.

GitHub Pages Issues

- **If your site doesn't appear:** Wait 5-10 minutes after enabling Pages, then check the URL.
- **If changes don't show:** Make sure you've committed and pushed your changes to GitHub.

Need Help?

- **Quarto Documentation:** quarto.org/docs
- **Matplotlib Tutorials:** matplotlib.org/tutorials
- **Python Virtual Environments:** docs.python.org/3/tutorial/venv.html
- **GitHub Pages:** docs.github.com/en/pages
- **Live Preview Extension:** marketplace.visualstudio.com/items?itemName=ms-vscode.live-server
- **Cursor Help:** Built-in help and AI assistance. Note: Cursor is built on VS Code, so search "VS Code [your question]" for solutions
- **Ask questions in class or office hours!**

Total time to complete Part 2: ~60 minutes

Why These Tools Matter

In the Real World:

- **GitHub Repositories** = Industry standard for project management and collaboration
- **Virtual Environments** = Industry standard for Python project management
- **Quarto** = Professional reporting and documentation (used by data scientists, analysts, researchers)
- **Matplotlib** = Most popular Python visualization library
- **GitHub Pages** = Free hosting for portfolios, documentation, and project websites
- **Live Preview** = Modern development workflow for web projects
- **AI-Assisted Development** = Modern approach to coding and documentation

For This Course:

- **All future challenges** will use Quarto for reports
- **Data visualizations** will be created with matplotlib and other Python libraries
- **Virtual environments** will keep your projects organized and reproducible
- **Professional presentation** of your analytical work
- **Portfolio development** for showcasing your skills to employers

You're now equipped with the essential tools for modern data analytics and have a professional portfolio to showcase your work!

Pro Tip: This repository and workflow will serve as the foundation for all your future analytics projects in this course and beyond!