ENEZA Data Science Residential Training Programme Schedule

Date: June 30th, 2025 - August 1, 2025

Venue:

- ✓ **First 3 Weeks:** Online
- ✓ **Final 2 Weeks:** International Centre of Insect Physiology and Ecology (ICIPE), Kenya
- ✓ **Session Times:** 9:00 AM – 5:00 PM daily

1. Orientation and Introduction to Data Science

- ✓ Structure of the program.
- ✓ Training objectives,
- ✓ expected outcomes
- ✓ Role of data science in healthcare,

Installation of Software

- ❖ Preparing a Linux environment:
  - Terminal basics.
  - Package installation (e.g., Python, R).
- ❖ Software installation:
  - Jupyter Notebook setup.
  - IDEs for coding: VSCode and RStudio.

2. Linux/Bash Basics

- ❖ **Introduction to Linux**
  - What is Linux? Overview of Linux distributions (e.g., Ubuntu, CentOS, Debian)
  - Key components of Linux: Kernel, shell, file system
  - Importance of Linux in data science, bioinformatics, and IT
- ❖ **Linux File System and Navigation**
  - Directory structure in Linux (e.g., `/home`, `/etc`, `/var`)
  - Navigating directories using commands (`cd`, `ls`, `pwd`)
  - Understanding relative vs. absolute paths
- ❖ **Essential Linux Commands**
  - File and directory operations: `mkdir`, `cp`, `mv`, `rm`, `touch`

  - Viewing and editing files: `cat`, `less`, `nano`, `vim`
  - Disk usage and storage: `df`, `du`, `lsblk`
  - Process monitoring: `top`, `ps`, `kill`

- **Managing Permissions and Ownership**
  - Understanding file permissions (`rwx`, `chmod`)
  - Changing ownership with `chown`
  - Managing groups and user accounts
- **Text Processing Tools**
  - Searching files with `grep` and `find`
  - Sorting and filtering data using `sort`, `uniq`, and `awk`
  - Editing text with `sed`
- **Networking Basics**
  - Checking network status: `ping`, `ifconfig`, `netstat`
  - File transfer tools: `scp`, `rsync`, `wget`, `curl`
  - SSH for remote access and server management
- **Bash Scripting Basics**
  - Introduction to Bash scripting and its benefits
  - Writing and executing simple scripts
  - Variables, loops (`for`, `while`), and conditionals (`if`, `case`)
  - Scheduling tasks with `cron`
- **Software Installation and Management**
  - Installing packages using package managers (e.g., `apt`, `yum`, `zypper`)
  - Updating and upgrading software
  - Managing dependencies in scientific environments
- **File Compression and Archiving**
  - Creating and extracting archives: `tar`, `zip`, `unzip`, `gzip`
  - Managing large datasets with efficient compression techniques
- **Troubleshooting and System Monitoring**
  - Checking system logs (`dmesg`, `journalctl`, `syslog`)
  - Identifying and killing resource-hogging processes
  - Monitoring CPU and memory usage with `htop`, `vmstat`
- **Hands-On Exercises**
  - Navigating the file system and creating a directory structure

- Writing a basic Bash script to automate a simple task
- Compressing and decompressing files using `tar` and `gzip`
- Using `grep` and `awk` to filter and process log files

3. Git/GitHub and Markdown

❖ Introduction to Version Control

- What is version control, and why is it important?
- Overview of Git: Distributed version control system
- Comparison of Git with other version control systems (e.g., SVN, Mercurial)

❖ Installing and Configuring Git

- Setting up Git on Linux, macOS, or Windows
- Configuring user details (`git config --global`)
- Understanding `.gitconfig`

❖ Basic Git Workflow

- Initializing a repository (`git init`)
- Adding and committing changes (`git add`, `git commit`)
- Checking the status of the repository (`git status`)
- Viewing the commit history (`git log`)

❖ Branching and Merging

- Creating and switching branches (`git branch`, `git checkout`)
- Merging branches (`git merge`)
- Resolving merge conflicts
- Understanding branching strategies (e.g., feature branching, GitFlow)

❖ Remote Repositories and GitHub

- What is GitHub, and why use it?
- A walk through the GitHub GUI
- Creating a remote repository on GitHub
- Pushing changes to a remote repository (`git push`)
- Cloning repositories (`git clone`)
- Pulling changes from a remote repository (`git pull`)

❖ Collaborative Workflows

- Forking repositories and creating pull requests

- Reviewing and merging pull requests on GitHub
- Using GitHub issues and discussions for project management
- Best practices for collaboration

❖ **Advanced Git Features**

- Stashing changes (`git stash`)
- Rebasing vs. merging (`git rebase`, `git merge`)
- Tagging specific commits (`git tag`)
- Undoing changes (`git reset`, `git revert`)

❖ **Markdown Basics**

- What is Markdown, and why use it?
- Syntax for headers, lists, links, images, and code blocks
- Creating tables in Markdown
- Writing README files for repositories

❖ **GitHub Pages and Documentation**

- Hosting websites with GitHub Pages
- Writing effective project documentation using Markdown
- Customizing Markdown with emojis, checklists, and other features

❖ **GitHub Actions and Automation**

- Introduction to GitHub Actions for CI/CD
- Setting up a simple GitHub Action workflow
- Automating testing, deployment, and notifications

❖ **Hands-On Exercises**

- Setting up a Git repository and making initial commits
- Creating and merging branches in a small project
- Writing a README.md file using Markdown
- Collaborating on a GitHub repository with pull requests
- Publishing a personal website using GitHub Pages

4. Containerization (Conda, Docker, and Singularity)

❖ **Introduction to Containerization**

- What is containerization?
- Benefits of containerization: Portability, scalability, and reproducibility
- Difference between containers and virtual machines

- Key containerization tools (e.g., Docker, Singularity, Conda)

- **Understanding Conda**
  - ✓ What is Conda? Overview of Anaconda and Miniconda
  - ✓ Managing environments: Creating, activating, deactivating, and deleting environments
  - ✓ Installing and managing packages within a Conda environment
  - ✓ Sharing environments using `environment.yml` files
  - ✓ Best practices for reproducibility using Conda
- **Introduction to Docker**
  - What is Docker, and why is it widely used?
  - Key components of Docker: Images, containers, Dockerfiles, and registries
  - Installing Docker and setting up a Docker environment
  - Basic Docker commands (`docker run`, `docker ps`, `docker stop`, `docker rm`)
- **Working with Docker Images**
  - Understanding Docker images and containers
  - Pulling images from Docker Hub
  - Building custom images using Dockerfiles
  - Best practices for writing Dockerfiles
- **Managing Containers with Docker**
  - Running, stopping, and restarting containers
  - Managing persistent data with volumes
  - Networking in Docker: Connecting containers and exposing ports
  - Cleaning up unused containers, images, and volumes
- **Introduction to Singularity**
  - Why Singularity? Differences between Docker and Singularity
  - Setting up Singularity on Linux systems
  - Building and running Singularity containers
  - Sharing and using Singularity images
- **Applications of Containerization in Data Science and Bioinformatics**
  - Running reproducible bioinformatics workflows (e.g., pipeline execution)

- Sharing complex environments across systems
- Using containers in cluster and cloud computing environments
- Integration with machine learning frameworks (e.g., TensorFlow, PyTorch)

❖ **Best Practices for Containerization**

- Creating lightweight and efficient containers
- Version control for container images
- Avoiding common pitfalls (e.g., including sensitive data in images)
- Documenting and sharing containerized workflows

❖ **Security in Containerization**

- Understanding container security risks
- Best practices for securing Docker and Singularity containers
- Managing user permissions and privileges within containers

❖ **Hands-On Exercises**

- Creating and managing Conda environments for a sample project
- Writing a Dockerfile to containerize a simple application
- Pulling and running a bioinformatics Docker image
- Building and running a Singularity container for a specific task
- Sharing a containerized workflow with colleagues or on a repository

5. R Basics, Statistical Analysis, and Plotting

❖ **Introduction to R**

- What is R, and why is it popular in data analysis?
- Installing R and RStudio
- Overview of RStudio interface: Console, script editor, environment, and plots panes

❖ **R Basics**

- Writing and executing R scripts
- Variables and data types in R: Numeric, character, logical, and factors
- Basic operations and arithmetic in R
- Working with vectors, matrices, and lists

❖ **Data Import and Export**

- Reading data from CSV, Excel, and text files (`read.csv`, `readxl`)

- Writing data to files (`write.csv`, `write.xlsx`)
- Loading and saving R data objects (`save`, `load`)
- Introduction to the `tidyverse` package for data manipulation

❖ **Data Wrangling with `dplyr` and `tidyr`**

- Selecting and filtering data using `select()` and `filter()`
- Summarizing data with `group_by()` and `summarize()`
- Pivoting data: `pivot_longer()` and `pivot_wider()`
- Joining datasets: `left_join()`, `inner_join()`, and more

❖ **Statistical Analysis**

- Descriptive statistics: Mean, median, mode, variance, and standard deviation
- Hypothesis testing: t-tests, chi-square tests, and ANOVA
- Correlation analysis and regression models (linear and logistic)
- Non-parametric tests (e.g., Wilcoxon rank-sum, Kruskal-Wallis)

❖ **Data Visualization Basics**

- Introduction to `ggplot2` for data visualization
- Creating basic plots: Histograms, bar charts, scatterplots, and line graphs
- Customizing plots: Titles, labels, themes, and legends
- Saving plots in different formats (e.g., PNG, PDF)

❖ **Advanced Plotting Techniques**

- Creating multi-faceted plots with `facet_wrap()` and `facet_grid()`
- Using color scales and gradients for categorical and continuous data
- Adding annotations, labels, and error bars to plots
- Combining multiple plots using `patchwork` or `cowplot` packages

❖ **Statistical Plotting**

- Visualizing distributions (e.g., boxplots, density plots)
- Creating regression plots and diagnostic plots for models
- Plotting time-series data and trends

❖ **R Packages for Specialized Analyses**

- Overview of useful R packages: `caret` (machine learning), `phyloseq` (microbial data), `DESeq2` (genomics)

- Installing and managing R packages (`install.packages()`, `library()`)
- Troubleshooting package installations and dependencies

❖ **Reproducible Research with R**

- Introduction to R Markdown for creating dynamic reports
- Combining R code, results, and text in R Markdown documents
- Exporting reports to HTML, PDF, or Word formats
- Sharing R projects and scripts

❖ **Hands-On Exercises**

- Loading and exploring a sample dataset in R
- Performing basic statistical tests on real-world data
- Creating a publication-quality plot using `ggplot2`
- Writing an R Markdown document summarizing results and visualizations

6. Python Basics and Data Visualization

❖ **Introduction to Python**

- What is Python, and why is it widely used?
- Installing Python and setting up environments (e.g., Anaconda, virtualenv)
- Overview of IDEs: Jupyter Notebook, VS Code, and PyCharm
- Understanding Python's role in data analysis and visualization

❖ **Python Basics**

- Writing and executing Python scripts
- Variables, data types, and basic operations
- Control structures: Loops (`for`, `while`) and conditionals (`if-else`)
- Functions: Writing and using reusable code blocks
- Importing and using Python libraries

❖ **Data Structures in Python**

- Working with lists, tuples, sets, and dictionaries
- Understanding and using NumPy arrays for numerical computations
- Indexing, slicing, and manipulating data structures

❖ **Data Manipulation with Pandas**

- Importing datasets with Pandas (`read_csv`, `read_excel`)

- Exploring data: `head()`, `info()`, `describe()`
- Filtering, sorting, and selecting data
- Handling missing data and duplicates
- Grouping and aggregating data with `groupby()`

❖ **Introduction to Data Visualization**

- Overview of Python visualization libraries: Matplotlib, Seaborn, and Plotly
- Creating basic plots: Line plots, bar charts, scatter plots, and histograms
- Customizing plots: Titles, labels, legends, and themes

❖ **Advanced Data Visualization with Seaborn**

- Visualizing distributions: Boxplots, violin plots, and density plots
- Correlation heatmaps and pairplots for exploring relationships
- Creating multi-faceted plots with `FacetGrid`
- Customizing color palettes and styles

❖ **Interactive Visualizations with Plotly**

- Introduction to Plotly for dynamic visualizations
- Creating interactive line and scatter plots
- Visualizing geographical data with maps
- Exporting interactive visualizations to HTML

❖ **Time-Series Data Visualization**

- Basics of time-series data: Parsing dates and timestamps
- Plotting trends and seasonality in time-series data
- Highlighting key events with annotations

❖ **Data Storytelling**

- Principles of effective data visualization
- Choosing the right chart type for the data
- Using color and design to convey insights clearly
- Avoiding common pitfalls in data visualization

❖ **Hands-On Exercises**

- Loading a dataset and performing exploratory analysis using Pandas
- Creating a multi-panel visualization using Matplotlib and Seaborn
- Building an interactive dashboard with Plotly Express
- Visualizing correlations and trends in a real-world dataset

7. Introduction to Molecular Biology/Genomics/Bioinformatics

✓ **Fundamentals of Molecular Biology**

- Structure and function of DNA, RNA, and proteins
- Central Dogma: Transcription and Translation
- Genetic code and codons
- Gene structure: Exons, introns, and regulatory regions

✓ **Genomics Basics**

- What is a genome?
- Genomic organization in prokaryotes vs. eukaryotes
- Coding vs. non-coding regions
- Introduction to Next-Generation Sequencing (NGS)

✓ **Bioinformatics Overview**

- Definition and scope of bioinformatics
- Key applications in genomics and healthcare
- Role of databases in bioinformatics (e.g., NCBI, EMBL, UniProt)

✓ **Tools and Software in Bioinformatics**

- Sequence alignment tools (e.g., BLAST, Clustal Omega)
- Genome annotation tools
- Databases for gene and protein information

✓ **Basics of Sequence Analysis**

- DNA, RNA, and protein sequence formats
- Concepts of sequence alignment: Local vs. global
- Evaluating sequence similarity and identity

✓ **Introduction to Phylogenetics**

- Importance of phylogenetic analysis
- Constructing phylogenetic trees
- Applications in evolutionary studies

✓ **Applications of Molecular Biology in Healthcare**

- Personalized medicine and genomics
- Gene editing technologies (e.g., CRISPR)
- Role of molecular diagnostics

✓ **Key Ethical and Legal Considerations**

- Privacy and data sharing in genomics
- Ethical implications of gene editing
- Understanding intellectual property in bioinformatics

✓ **Emerging Trends in Genomics and Bioinformatics**

- Single-cell genomics
- Metagenomics and microbiome studies
- Integration of AI in genomics

✓ **Hands-On Exercises**

- Using online tools like NCBI for sequence retrieval
- Aligning sequences using BLAST
- Annotating a gene or protein sequence

8. Machine Learning

✓ **Fundamentals of Machine Learning**

- Definition and core concepts: Supervised, unsupervised, and reinforcement learning
- Machine learning workflow: Data collection, preprocessing, training, testing, and evaluation
- Types of machine learning models (e.g., regression, classification, clustering)

✓ **Supervised Learning**

- Basics of regression (e.g., linear regression, logistic regression)
- Classification techniques (e.g., decision trees, random forests, support vector machines)
- Use cases in healthcare (e.g., disease prediction, diagnostic tools)

✓ **Unsupervised Learning**

- Clustering algorithms (e.g., k-means, hierarchical clustering)
- Dimensionality reduction (e.g., PCA, t-SNE)
- Applications in data exploration and anomaly detection

✓ **Introduction to Neural Networks**

- Basics of neural network architecture: Layers, neurons, and activation functions

  - Deep learning concepts and frameworks (e.g., TensorFlow, PyTorch)
  - Applications in image analysis and speech recognition

- ✓ **Feature Engineering**
  - Importance of feature selection and extraction
  - Techniques for handling categorical and numerical data
  - Role of domain knowledge in feature creation

- ✓ **Model Evaluation and Metrics**
  - Accuracy, precision, recall, F1-score, and ROC-AUC
  - Overfitting and underfitting
  - Cross-validation techniques

- ✓ **Applications of Machine Learning in Healthcare**
  - Predictive modeling in patient care
  - Medical image analysis (e.g., detecting tumors in radiology images)
  - NLP for clinical data (e.g., analyzing electronic health records)
  - Drug discovery and genomics

- ✓ **Emerging AI Tools in Healthcare**
  - Role of generative AI (e.g., ChatGPT, DALL-E)
  - Federated learning for privacy-preserving ML in healthcare
  - Explainable AI and ethical considerations

- ✓ **Data Preparation for Machine Learning**
  - Data cleaning and preprocessing
  - Dealing with missing data and outliers
  - Data normalization and scaling

- ✓ **Hands-On Exercises**
  - Building a basic classification model using Python (e.g., Scikit-learn)
  - Training and evaluating a machine learning model
  - Visualizing model performance (e.g., confusion matrices, ROC curves)

9. NLP for Text Mining and Clinical Data Analysis

- ✓ **Fundamentals of NLP**
  - Definition and scope of Natural Language Processing (NLP)

    - Key challenges in NLP (e.g., ambiguity, context understanding)
    - The role of NLP in data science and healthcare

- ✓ **Basic Text Processing**
    - Tokenization: Breaking text into sentences, words, or subwords
    - Stopword removal and stemming/lemmatization
    - N-grams and their applications in text analysis
    - Regular expressions for text parsing
- ✓ **Feature Extraction from Text**
    - Bag-of-Words (BoW) model
    - Term Frequency-Inverse Document Frequency (TF-IDF)
    - Word embeddings (e.g., Word2Vec, GloVe)
- ✓ **NLP Tools and Libraries**
    - Overview of popular libraries: NLTK, SpaCy, and Hugging Face Transformers
    - Using Python for NLP tasks
    - Introduction to pre-trained models for NLP (e.g., BERT, GPT)
- ✓ **Sentiment Analysis**
    - Concept and importance in clinical data (e.g., patient feedback)
    - Building sentiment analysis models
    - Applications in healthcare and beyond
- ✓ **Named Entity Recognition (NER)**
    - Identifying entities like diseases, drugs, and symptoms in clinical text
    - NER tools and techniques
    - Applications in extracting meaningful insights from healthcare records
- ✓ **Text Classification and Categorization**
    - Basics of text classification (e.g., spam detection, document labeling)
    - Applications in healthcare (e.g., triaging medical reports, identifying health-related topics)
    - Model evaluation metrics for text classification
- ✓ **NLP Applications in Healthcare**
    - Analyzing Electronic Health Records (EHRs)

  - Clinical decision support systems
  - Identifying adverse drug reactions through text mining
  - Literature mining for drug discovery and genomics
- ✓ **Emerging Trends in NLP**
  - Advances in transformer models (e.g., ChatGPT, T5, BERT)
  - Multilingual NLP for global healthcare applications
  - Explainable NLP and ethical considerations
- ✓ **Hands-On Exercises**
  - Tokenizing and processing text data with NLTK or SpaCy
  - Building a simple sentiment analysis model using Python
  - Extracting entities from clinical text using pre-trained NER models
  - Analyzing clinical datasets (e.g., MIMIC-III or PubMed abstracts)

10. Blockchain Fundamentals and Use Cases in Healthcare

- ✓ **Introduction to Blockchain Technology**
  - Definition and core concepts of blockchain
  - How blockchain works: Blocks, transactions, and chains
  - Key features: Decentralization, immutability, transparency, and security
  - Difference between blockchain and traditional databases
- ✓ **Blockchain Architecture**
  - Components of a blockchain: Nodes, ledgers, consensus mechanisms
  - Types of blockchains: Public, private, consortium, and hybrid
  - Overview of consensus algorithms (e.g., Proof of Work, Proof of Stake, Practical Byzantine Fault Tolerance)
- ✓ **Cryptography Basics**
  - Hash functions and their role in securing data
  - Public and private key cryptography
  - Digital signatures and verification
- ✓ **Smart Contracts**
  - Definition and purpose of smart contracts
  - How smart contracts automate processes
  - Popular platforms for smart contracts (e.g., Ethereum, Hyperledger)

- Example use cases in healthcare: Automated insurance claims, clinical trial management

✓ **Blockchain Applications in Healthcare**

- Supply chain management for pharmaceuticals (e.g., drug traceability)
- Secure sharing of Electronic Health Records (EHRs)
- Patient consent management and privacy
- Clinical trial data integrity and transparency

✓ **Blockchain Implementation Challenges in Healthcare**

- Scalability and transaction speed
- Data privacy and compliance with regulations (e.g., HIPAA, GDPR)
- Interoperability with existing healthcare systems
- Cost and infrastructure requirements

✓ **Emerging Trends in Blockchain**

- Integration with AI and IoT in healthcare
- The role of blockchain in personalized medicine
- Decentralized Autonomous Organizations (DAOs) for collaborative research
- Blockchain-based digital identity solutions

✓ **Ethical and Legal Considerations**

- Ensuring data privacy and security
- Addressing the risks of centralization within private blockchains
- Regulatory compliance in different regions
- Challenges in managing public trust and adoption

✓ **Blockchain Ecosystem and Tools**

- Overview of popular blockchain platforms (e.g., Ethereum, Hyperledger Fabric, Polkadot)
- Tools for building and deploying blockchain applications (e.g., Solidity, Truffle)
- Blockchain explorers for monitoring transactions

✓ **Hands-On Exercises**

- Setting up a private blockchain network
- Creating and deploying a simple smart contract using Ethereum

  - Simulating a blockchain-based healthcare use case (e.g., tracking pharmaceutical shipments)
  - Analyzing a blockchain ledger for transaction data

11. Open Science and Research Data Management

- ✓ **Introduction to Open Science**
  - Definition and principles of open science
  - The importance of transparency, reproducibility, and accessibility in research
  - Open science frameworks and policies (e.g., FAIR principles, Plan S)
- ✓ **Benefits of Open Science**
  - Enhancing collaboration and innovation
  - Increasing research visibility and impact
  - Fostering trust in scientific findings
  - Reducing duplication of efforts
- ✓ **Research Data Management (RDM) Basics**
  - Definition and importance of RDM
  - The research data lifecycle: Planning, collection, processing, analysis, sharing, and preservation
  - Understanding metadata and its role in data organization
- ✓ **Data Management Plans (DMPs)**
  - What is a DMP, and why is it essential?
  - Key components of a DMP: Objectives, data types, storage, sharing, and preservation plans
  - Tools for creating DMPs (e.g., DMPTool, OpenAIRE)
- ✓ **Data Sharing and Licensing**
  - Understanding open data and open access
  - Data sharing platforms and repositories (e.g., Zenodo, Dryad, Figshare)
  - Licensing data for reuse: Creative Commons and other licensing frameworks
- ✓ **Ethical Considerations in Open Science**
  - Privacy and confidentiality in data sharing

- Balancing openness with intellectual property rights
- Handling sensitive or restricted data

✓ **Tools and Platforms for Open Science**

- Open access publishing platforms (e.g., PLOS, BioRxiv)
- Collaborative research tools (e.g., GitHub, Jupyter Notebooks)
- Electronic Lab Notebooks (ELNs) and their benefits

✓ **Ensuring Data Quality and Integrity**

- Strategies for maintaining high data quality
- Detecting and addressing biases in research data
- Data validation and verification practices

✓ **Challenges in Open Science and RDM**

- Technical barriers: Storage, processing, and sharing infrastructure
- Cultural resistance to open practices
- Addressing interoperability and standardization issues

✓ **Emerging Trends and Future of Open Science**

- Integration of AI and big data in open science
- The role of citizen science in research
- Blockchain and decentralized platforms for research data management
- The shift towards open peer review

✓ **Hands-On Exercises**

- Developing a data management plan for a sample project
- Exploring open data repositories and sharing datasets
- Using GitHub for collaborative research
- Simulating data publication and licensing