Course 2: Tools for Data Science

Module 1: Overview of Data Science Tools

In this module, you will learn about the different types and categories of tools that data scientists use and popular examples of each. You will also become familiar with Open Source, Cloud-based, and Commercial options for data science tools.

- Describe the components of a Data Scientist's toolkit and list various tool categories
- List examples of Open Source, Commercial, and Cloud-based tools in various categories

Module 2: Languages of Data Science

This module will bring awareness about the criteria determining which language you should learn. You will learn the benefits of Python, R, SQL, and other common languages such as Java, Scala, C++, JavaScript, and Julia. You will explore how you can use these languages in Data Science. You will also look at some sites for more information about the languages.

Learning Objectives

- Identify the criteria and roles for determining the language to learn.
- Identify the users and benefits of Python.
- Identify the users and uses of the R language.
- Define SQL elements and list their benefits.
- Review languages such as Java, Scala, C++, JavaScript, and Julia.
- List the global communities for connecting with other users.

Module 3: Packages, APIs, Data Sets and Models

This module will give you in-depth knowledge of different libraries, APIs, dataset sources and models used by data scientist.

Learning Objectives

- List examples of the various libraries: scientific, visualization, machine learning, and deep learning.
- Define REST API to request and respond.
- Describe data sets and sources of data.
- Explore open data sets on the Data Asset eXchange.
- Describe how to use a learning model to solve a problem.
- List the tasks that a data scientist needs to perform to build a model.
- Explore ML models in the Model Learning eXchange.

Module 4: Jupyter Notebooks and JupyterLab

This module introduces the Jupyter Notebook and JupyterLab. You will learn how to work with different kernels and the basic Jupyter architecture. In addition, you will identify the tools in an Anaconda Jupyter environment. Finally, the module overviews cloud-based Jupyter environments and their data science features.

Learning Objectives

- Describe how to use the notebooks in JupyterLab.
- Describe how to work in a notebook session.
- Describe the basic Jupyter architecture.
- Describe how to work with kernels.
- Identify tools in Anaconda Jupyter environments.

- Describe cloud-based Jupyter environments and their data science features.

Module 5: RStudio and GitHub
This module will start with an introduction to R and RStudio and will end up with Github usage. You will learn about the different R visualization packages and how to create visual charts using the plot function.
Further in the module, you will develop the essential conceptual and hands-on skills to work with Git and GitHub. You will start with an overview of Git and GitHub, creating a GitHub account and a project repository, adding files, and committing your changes using the web interface. Next, you will become familiar with Git workflows involving branches, pull requests (PRs), and merges. You will also complete a project at the end to apply and demonstrate your newly acquired skills.
Learning Objectives

- Describe R capabilities and RStudio environment.
- Use the inbuilt R plot function.
- Explain version control and describe the Git and GitHub environment.
- Describe the purpose of source repositories and explain how GitHub satisfies the needs of a source repository.
- Create a GitHub account and a project repository.
- Demonstrate how to edit and upload files in GitHub.
- Explain the purpose of branches and how to merge changes.

Module 6: Final Project and Assessment
In this module, you will work on a final project to demonstrate some of the skills learned in the course. You will also be tested on your knowledge of various components and tools in a Data Scientist's toolkit learned in the previous modules.
Learning Objectives

- Create a Jupyter Notebook with markdown and code cells
- List examples of languages, libraries and tools used in Data Science
- Share your Jupyter Notebook publicly on GitHub
- Evaluate notebooks submitted by your peers using the provided rubric
- Demonstrate proficiency in Data Science toolkit knowledge

Module7: IBM Watson Studio
This is as an optional module if you are interested in learning about and working with data science tools from IBM such as Watson Studio.
Learning Objectives

- Find common resources in Watson Studio and IBM Cloud Pak for Data.
- Create an IBM Cloud account, service, and project in Watson Studio.
- Create and share a Jupyter Notebook.
- Use different types of Jupyter Notebook templates and kernels on IBM Watson Studio.
- Describe how to connect a Watson Studio account and publish a notebook in GitHub.

Module 1: Overview of Data Science Tools

หมวดหมู่ของงานด้านวิทยาศาสตร์ข้อมูล

1. **การจัดการข้อมูล (Data Management)**
    - กระบวนการเก็บรวบรวม จัดเก็บ และเรียกใช้ข้อมูลอย่างปลอดภัย มีประสิทธิภาพ และคุ้มค่า
    - แหล่งข้อมูลมาจากหลายที่ เช่น Twitter, Flipkart, สื่อ, และเซ็นเซอร์
    - เก็บข้อมูลในที่เก็บถาวรเพื่อให้สามารถเรียกใช้งานได้ทุกเมื่อ
2. **การรวมและแปลงข้อมูล (Data Integration and Transformation)**
    - กระบวนการ Extract, Transform, Load (ETL)
    - การดึงข้อมูลจากแหล่งต่าง ๆ มารวมไว้ในคลังข้อมูล (Data Warehouse)
    - การแปลงค่าของข้อมูลให้เป็นรูปแบบที่ต้องการ เช่น แปลงข้อมูลน้ำหนักและส่วนสูงเป็นหน่วยเมตริก
3. **การแสดงผลข้อมูล (Data Visualization)**
    - การแสดงข้อมูลในรูปแบบกราฟฟิก เช่น แผนภูมิแท่ง แผนภูมิเส้น แผนที่ ฯลฯ
    - ช่วยให้ผู้ตัดสินใจเห็นภาพรวมและเข้าใจข้อมูลได้ดียิ่งขึ้น
4. **การสร้างโมเดล (Model Building)**
    - การฝึกฝนข้อมูลและวิเคราะห์รูปแบบด้วยอัลกอริทึมการเรียนรู้ของเครื่อง (Machine Learning)
    - ใช้โมเดลนี้เพื่อทำนายข้อมูลใหม่ ๆ ที่ไม่เคยเห็นมาก่อน
5. **การปรับใช้โมเดล (Model Deployment)**
    - การรวมโมเดลที่พัฒนาแล้วเข้าสู่สภาพแวดล้อมการผลิต
    - ให้ผู้ใช้งานสามารถเข้าถึงและใช้งานข้อมูลผ่านแอพพลิเคชันของบุคคลที่สาม
6. **การตรวจสอบและประเมินผลโมเดล (Model Monitoring and Assessment)**
    - การตรวจสอบคุณภาพของโมเดลเพื่อให้แน่ใจว่าแม่นยำ ยุติธรรม และแข็งแรง
    - การใช้เครื่องมือประเมินผลเช่น F1 score และ True Positive Rate

เครื่องมือที่สนับสนุนการทำงานด้านวิทยาศาสตร์ข้อมูล

1. **การจัดการสินทรัพย์โค้ด (Code Asset Management)**
    - การจัดการเวอร์ชันของโค้ด เช่น GitHub
    - ช่วยในการแชร์และปรับปรุงโค้ดร่วมกันในทีม
2. **การจัดการสินทรัพย์ข้อมูล (Data Asset Management)**
    - การจัดเก็บและจัดการข้อมูลที่สำคัญจากแหล่งต่าง ๆ อย่างมีระบบ
    - สนับสนุนการทำสำเนา การสำรองข้อมูล และการจัดการสิทธิ์การเข้าถึง
3. **สภาพแวดล้อมการพัฒนา (Development Environments)**
    - Integrated Development Environments (IDEs) เช่น IBM Watson Studio
    - ให้เครื่องมือสำหรับการพัฒนา ทดสอบ และจำลองการทำงานของโค้ด
4. **สภาพแวดล้อมการปฏิบัติการ (Execution Environments)**
    - มีไลบรารีในการคอมไพล์โค้ดและทรัพยากรระบบสำหรับการดำเนินการและตรวจสอบโค้ด
    - สภาพแวดล้อมบนคลาวด์ที่ไม่ผูกติดกับฮาร์ดแวร์หรือซอฟต์แวร์ใด ๆ โดยเฉพาะ

[ความหมายของ Data warehousing] Data warehousing คือ กระบวนการรวบรวมข้อมูลจากแหล่งต่าง ๆ แล้วจัดเก็บไว้ในระบบที่มีการออกแบบมาเพื่อการจัดเก็บและจัดการข้อมูลในลักษณะที่เป็นประโยชน์ต่อการวิเคราะห์และรายงานข้อมูลในเชิงธุรกิจ การจัดเก็บข้อมูลนี้จะเป็นแบบโครงสร้าง (structured) และสามารถนำมาใช้ในการวิเคราะห์ข้อมูลเชิงลึก การทำรายงาน การวิเคราะห์แนวโน้ม หรือการทำ Business Intelligence (BI)

องค์ประกอบหลักของ Data Warehousing มีดังนี้:

1. การสกัดข้อมูล (ETL - Extract, Transform, Load): เป็นกระบวนการที่ประกอบด้วยการสกัดข้อมูลจากแหล่งต่าง ๆ (เช่น ฐานข้อมูล, ระบบ ERP, ระบบ CRM), การเปลี่ยนรูปข้อมูลให้อยู่ในรูปแบบที่เหมาะสม และการโหลดข้อมูลเข้าสู่ Data Warehouse
2. ที่เก็บข้อมูล (Storage): ข้อมูลที่ถูกโหลดจะถูกเก็บในรูปแบบที่จัดโครงสร้างไว้เพื่อให้สามารถเรียกใช้งานได้ง่ายและรวดเร็ว เช่น เก็บใน Data Marts หรือ OLAP Cubes
3. การจัดการข้อมูล (Data Management): การจัดการข้อมูลใน Data Warehouse รวมถึงการปรับปรุงคุณภาพของข้อมูล การรักษาความปลอดภัย และการจัดการการเข้าถึงข้อมูล
4. การเข้าถึงและการวิเคราะห์ข้อมูล (Data Access and Analysis): เครื่องมือที่ใช้สำหรับการเข้าถึงข้อมูลและการวิเคราะห์ เช่น SQL Queries, Reporting Tools, และ BI Tools
5. การสำรองข้อมูล (Backup and Recovery): การทำสำรองข้อมูลเพื่อให้มั่นใจว่าข้อมูลจะไม่สูญหายและสามารถกู้คืนได้ในกรณีที่มีปัญหาเกิดขึ้น

Open-Source tools for Data Science

- **Data Management Tools**: MySQL, PostgreSQL, MongoDB, Apache CouchDB, Apache Cassandra, Hadoop File System, Ceph, Elasticsearch.
- **Data Integration and Transformation Tools**: Apache AirFlow, KubeFlow, Apache Kafka, Apache Nifi, Apache SparkSQL, NodeRED.
- **Data Visualization Tools**: Pixie Dust, Hue, Kibana, Apache Superset.
- **Model Deployment Tools**: Apache PredictionIO, Seldon, Kubernetes, Redhat OpenShift, Mleap, TensorFlow Serving, TensorFlow Lite, TensorFlow.js.
- **Model Monitoring Tools**: ModelDB, Prometheus, IBM AI Fairness 360, IBM Adversarial Robustness 360 Toolbox, IBM AI Explainability 360.
- **Code Asset Management Tools**: Git, GitHub, GitLab, Bitbucket.
- **Data Asset Management Tools**: Apache Atlas, ODPi Egeria, Kylo.

For Testing the knowledge about tools:
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0130EN-SkillsNetwork/storyline/Open%20Source%20Tools/story.html?origin=www.coursera.org

Module 1 Summary
Congratulations! You have completed this module. At this point in the course, you know:

- The Data Science Task Categories include:
    - Data Management - storage, management and retrieval of data
    - Data Integration and Transformation - streamline data pipelines and automate data processing tasks
    - Data Visualization - provide graphical representation of data and assist with communicating insights

    - Modelling - enable Building, Deployment, Monitoring and Assessment of Data and Machine Learning models
- Data Science Tasks support the following:
    - Code Asset Management - store & manage code, track changes and allow collaborative development
    - Data Asset Management - organize and manage data, provide access control, and backup assets
    - Development Environments - develop, test and deploy code
    - Execution Environments - provide computational resources and run the code

The data science ecosystem consists of many open source and commercial options, and include both traditional desktop applications and server-based tools, as well as cloud-based services that can be accessed using web-browsers and mobile interfaces.

Data Management Tools: include Relational Databases, NoSQL Databases, and Big Data platforms:

- MySQL, and PostgreSQL are examples of Open Source Relational Database Management Systems (RDBMS), and IBM Db2 and SQL Server are examples of commercial RDBMSes and are also available as Cloud services.
- MongoDB and Apache Cassandra are examples of NoSQL databases.
- Apache Hadoop and Apache Spark are used for Big Data analytics.

Data Integration and Transformation Tools: include Apache Airflow and Apache Kafka.

Data Visualization Tools: include commercial offerings such as Cognos Analytics, Tableau and PowerBI and can be used for building dynamic and interactive dashboards.

Code Asset Management Tools: Git is an essential code asset management tool. GitHub is a popular web-based platform for storing and managing source code. Its features make it an ideal tool for collaborative software development, including version control, issue tracking, and project management.

Development Environments: Popular development environments for Data Science include Jupyter Notebooks and RStudio.

- Jupyter Notebooks provides an interactive environment for creating and sharing code, descriptive text, data visualizations, and other computational artifacts in a web-browser based interface.
- RStudio is an integrated development environment (IDE) designed specifically for working with the R programming language, which is a popular tool for statistical computing and data analysis.

Module 2: Languages of Data Science

Python:

- **Users of Python**: Python is used by both experienced programmers and beginners due to its clear and readable syntax. It is widely used in data science, AI, machine learning, web development, and IoT. Large organizations like IBM, Google, NASA, and Amazon use Python.
- **Benefits of Using Python**: Python allows for the development of programs with less code compared to other languages. It has a vast global community and extensive documentation, making it accessible for beginners and professionals. Python's

libraries, such as Pandas, NumPy, SciPy, Matplotlib, TensorFlow, PyTorch, Keras, Scikit-learn, and NLTK, are highly valuable for data science and AI tasks.

- **Diversity and Inclusion in the Python Community**: The Python community, supported by the Python Software Foundation, promotes diversity and inclusion. Initiatives like PyLadies focus on creating inclusive environments and encouraging more women to participate in the Python community.

Python's extensive standard library supports a variety of tasks including databases, automation, web scraping, text and image processing, and data analytics. The language's strong community and support for diversity make it a leading choice in the tech industry.

R Language:

- **Comparison of Open Source and Free Software**: Both are free to use, support collaboration, and often use similar licenses like the General Public License (GNU). The Open-Source Initiative (OSI) champions open source, which is business-focused, while the Free Software Foundation (FSF) defines free software, which is value-focused.
- **Users of R**: R is used by statisticians, mathematicians, and data miners for developing statistical software, graphing, and data analysis. It is popular in academia and is used by companies like IBM, Google, Facebook, Microsoft, Bank of America, Ford, TechCrunch, Uber, and Trulia.
- **Benefits of Using R**: R is free software that allows private, commercial, and public collaboration. Its array-oriented syntax makes it easy for beginners with minimal programming background. R is highly compatible with other languages like C++, Java, C, .Net, and Python, and it supports complex mathematical operations. R also has strong object-oriented programming facilities and offers more than 15,000 publicly released packages for data analysis.
- **Global Communities for R**: To connect with other R users, you can join communities like useR, WhyR, SatRdays, and R-Ladies, or check the R project website for conferences and events.

R has become the world's largest repository of statistical knowledge, making it a powerful tool for data analysis and statistical computing.

SQL:

- **Explanation of SQL and Relational Databases**: SQL stands for "Structured Query Language" and is pronounced "ess cue el" or "sequel." It is a non-procedural language designed for querying and managing data in relational databases, which consist of two-dimensional tables formed by columns and rows. Although primarily for relational databases, SQL is also used with NoSQL and big data repositories due to its widespread use and ease of learning.
- **SQL Elements**: SQL is subdivided into several elements, including Clauses, Expressions, Predicates, Queries, and Statements.
- **Benefits of Using SQL**:
    - Essential for data science, business and data analysis, and data engineering.

- Direct data access speeds up workflows.
- ANSI standard allows for easy application of SQL knowledge across different databases.
- Compatible with many databases like MySQL, IBM DB2, PostgreSQL, SQLite, Oracle, MariaDB, and Microsoft SQL Server.
- The SQL syntax may vary slightly between different relational database management systems.

Other language:

- **Java**: A general-purpose, object-oriented language known for speed and scalability. Used in enterprise settings. Key data science tools: Weka, Java-ML, Apache MLlib, Deeplearning4, and Hadoop.
- **Scala**: Supports functional programming and runs on the JVM. Designed to overcome Java's limitations. Key data science tool: Apache Spark, which includes Shark, MLlib, GraphX, and Spark Streaming.
- **C++**: Enhances processing speed and system programming. Key data science tools: TensorFlow (foundation), MongoDB, and Caffe. Often used for performance-critical tasks in combination with high-level languages.
- **JavaScript**: Not related to Java, used both client-side and server-side (Node.js). Key data science tools: TensorFlow.js, brain.js, machinelearn.js, and R-js.
- **Julia**: Designed for high-performance numerical analysis and computational science. Combines the development ease of Python/R with the speed of C/Fortran. Key data science tool: JuliaDB.

Each language has compelling use cases in data science, offering various tools and libraries to address different needs and enhance data processing and analysis.

Module 2 Summary

Congratulations! You have completed this module. At this point in the course, you know:

- You should select a language to learn depending on your needs, the problems you are trying to solve, and whom you are solving them for.
- The popular languages are Python, R, SQL, Scala, Java, C++, and Julia.
- For data science, you can use Python's scientific computing libraries like Pandas, NumPy, SciPy, and Matplotlib.
- Python can also be used for Natural Language Processing (NLP) using the Natural Language Toolkit (NLTK).
- Python is open source, and R is free software.
- R language's array-oriented syntax makes it easier to translate from math to code for learners with no or minimal programming background.
- SQL is different from other software development languages because it is a non-procedural language.
- SQL was designed for managing data in relational databases.

- If you learn SQL and use it with one database, you can apply your SQL knowledge with many other databases easily.
- Data science tools built with Java include Weka, Java-ML, Apache MLlib, and Deeplearning4.
- For data science, popular program built with Scala is Apache Spark which includes Shark, MLlib, GraphX, and Spark Streaming.
- Programs built for Data Science with JavaScript include TensorFlow.js and R-js.
- One great application of Julia for Data Science is JuliaDB.

Module 3: Packages, APIs, Data Sets and Models

Libraries for Data Science:

- **Scientific Computing Libraries in Python**: Pandas offers data manipulation and analysis with two-dimensional tables called Data Frames. NumPy provides mathematical functions for arrays and matrices, serving as the foundation for Pandas.
- **Visualization Libraries in Python**: Matplotlib is popular for creating customizable graphs and plots. Seaborn, based on Matplotlib, generates heat maps, time series, and violin plots.
- **High-Level Machine Learning and Deep Learning Libraries in Python**: Scikit-learn provides tools for statistical modeling like regression, classification, and clustering. Keras offers a high-level interface for building standard deep learning models quickly. TensorFlow is a low-level framework for large-scale production of deep learning models. PyTorch simplifies experimentation for researchers.
- **Libraries in Other Languages**: Apache Spark, a cluster-computing framework, processes data in parallel across multiple computers. It has functionalities similar to Pandas, NumPy, and Scikit-learn. Scala libraries like Vegas provide statistical data visualizations, often used in data engineering and science. R has built-in machine learning and visualization capabilities, with popular libraries like ggplot2 for visualization and interfaces for Keras and TensorFlow.

Dataset:

1. **Definition of a Dataset:** A dataset is a structured collection of data, which can include text, numbers, images, audio, or video files. Tabular datasets are common, often represented as CSV files, where each row represents an observation and each column contains information about that observation.
2. **Types of Data Structures:** Data can be organized hierarchically in a tree-like format or as a network, represented by a graph. Raw data files, such as images or audio, can also be part of a dataset.
3. **Data Ownership:** Traditionally, datasets were private due to containing proprietary or confidential information. However, many public and private entities now provide open datasets, allowing free access to information on various topics.
4. **Open Data Sources:** Governments, scientific institutions, organizations, and companies contribute to open data by publishing datasets covering a wide range of fields such as economy, society, healthcare, and transportation. These datasets are valuable resources for data scientists, researchers, and analysts.

5. **Community Data License Agreement (CDLA):** The CDLA, created by the Linux Foundation, provides licenses for open data distribution. The CDLA-Sharing license allows data use and modification but requires sharing modifications under the same license terms. The CDLA-Permissive license allows data use and modification without sharing modifications.
6. **Importance of Open Data:** Open data is fundamental to data science, facilitating research, application development, and insights discovery. However, open datasets might not always meet enterprise requirements due to potential business impacts.

Machine Learning Models

Machine Learning Basics:

- **Machine Learning (ML)**: Uses algorithms, also known as models, to identify patterns in data.
- **Model Training**: The process by which a model learns patterns from data. Once trained, a model can make predictions based on new data.

Types of Machine Learning:

1. **Supervised Learning**:
    - **Definition**: A human provides input data and correct outputs. The model identifies relationships between the input data and the output.
    - **Types**:
        - **Regression Models**: Predict numeric values (e.g., predicting home prices based on features like location, size).
        - **Classification Models**: Predict categories (e.g., identifying emails as spam or not).
2. **Unsupervised Learning**:
    - **Definition**: The data is not labeled. The model identifies patterns and structures within the data.
    - **Examples**:
        - **Clustering Models**: Group similar records (e.g., purchase recommendations based on past behavior).
        - **Anomaly Detection**: Identifies outliers (e.g., detecting fraudulent transactions).
3. **Reinforcement Learning**:
    - **Definition**: Models learn through trial and error to maximize rewards. Similar to how a mouse learns to navigate a maze to get cheese.
    - **Applications**: Successful in games like Go, chess, and strategy video games.

Deep Learning:

- **Definition**: A specialized type of machine learning that emulates the human brain's problem-solving approach. It requires large datasets and significant computational power.
- **Applications**: Natural language processing, image and audio analysis, time series forecasting.
- **Frameworks**: TensorFlow, PyTorch, Keras.
- **Model Zoos**: Repositories of pre-trained models (e.g., TensorFlow Model Zoo, PyTorch Hub).

Building a Model:

1. **Data Collection and Preparation**: Gathering and labeling data (e.g., labeling images with bounding boxes around objects).
2. **Model Selection and Training**: Choosing or building a model and training it on the prepared data.
3. **Evaluation and Deployment**: Analyzing training results, refining the model, and deploying it for application use.

Conclusion:

- Machine learning uses algorithms to identify data patterns.
- The types of ML include Supervised, Unsupervised, and Reinforcement learning.
- Supervised learning includes regression and classification models.
- Deep learning is a powerful subset of ML emulating the human brain's problem-solving capabilities.

Module 3 Summary

Congratulations! You have completed this module. At this point in the course, you know that:

- Python offers a diverse library ecosystem for data science, covering scientific computing (Pandas, NumPy), visualization (Matplotlib, Seaborn), and high-level machine learning (Scikit-learn). These libraries offer tools for data manipulation, mathematical operations, and simplified machine learning model development.
- Application Programming Interfaces (APIs) facilitate communication between software components. REST APIs, specifically, facilitate internet communication and access resources like storage. Key API terms include client (user or code accessing it), resource (service or data), and endpoint (API's URL).
- Machine learning models analyze data and identify patterns to make predictions and automate complex tasks—the three fundamental types of machine learning are supervised, unsupervised, and reinforcement learning. Supervised learning includes regression and classification models for predictive modeling and pattern recognition. Deep learning, an advanced subset of machine learning, mimics the brain's processing, enabling intricate problem-solving in various domains.
- The Community Data License Agreement (CDLA) facilitates open data sharing by providing clear licensing terms for distribution and use, and the IBM Data Asset eXchange (DAX) site contains high-quality open data sets.
- The Model Asset eXchange (MAX) provides a wealth of pre-trained deep learning models, empowering developers with readily deployable solutions for various business challenges.

Module 4 Summary

Congratulations! You have completed this module. At this point in the course, you know:

- Jupyter Notebooks are used in Data Science for recording experiments and projects.
- Jupyter Lab is compatible with many files and Data Science languages.
- There are different ways to install and use Jupyter Notebooks.
- How to run, delete, and insert a code cell in Jupyter Notebooks.
- How to run multiple notebooks at the same time.
- How to present a notebook using a combination of Markdown and code cells.
- How to shut down your notebook sessions after you have completed your work on them.
- Jupyter implements a two-process model with a kernel and a client.
- The notebook server is responsible for saving and loading the notebooks.
- The kernel executes the cells of code contained in the Notebook.
- The Jupyter architecture uses the NB convert tool to convert files to other formats.
- Jupyter implements a two-process model with a kernel and a client.
- The Notebook server is responsible for saving and loading the notebooks.
- The Jupyter architecture uses the NB convert tool to convert files to other formats.
- The Anaconda Navigator GUI can launch multiple applications on a local device.
- Jupyter environments in the Anaconda Navigator include JupyterLab and VS Code.
- You can download Jupyter environments separately from the Anaconda Navigator, but they may not be configured properly.
- The Anaconda Navigator GUI can launch multiple applications.
- Additional open-source Jupyter environments include JupyterLab, JupyterLite, VS Code, and Google Colaboratory.
- JupyterLite is a browser-based tool.

Module 5: RStudio and GitHub

R Language:

- **Definition:** R is a statistical programming language.
- **Capabilities:** It is a powerful tool for data processing and manipulation, statistical inference, data analysis, and machine learning.
- **Usage:** As of 2017, R is predominantly used by academics, healthcare, and government sectors.
- **Data Import:** R supports importing data from various sources such as flat files, databases, the web, and statistical software like SPSS and STATA.
- **Ease of Use:** R functions are known for their ease of use.
- **Visualizations:** R is renowned for producing excellent visualizations and includes packages for data analysis without requiring additional libraries.

RStudio:

- **Description:** RStudio is a popular integrated development environment (IDE) for developing and running R language code and programs.
- **Productivity:** It enhances productivity with features such as:
    - A syntax-highlighting editor that supports direct code execution and records your work.
    - A Console for typing R commands.

    - A workspace and History tab that show the list of R objects created during a session and the history of all previous commands.
    - Files tab to show files in the working directory.
    - Plots tab to display and export plots to PDF or image files.
    - Packages tab to display external R packages available locally.
    - Help tab for resources on R, RStudio support, and packages.

Popular R Libraries for Data Science:

- **dplyr:** For data manipulation.
- **stringr:** For string manipulation.
- **ggplot:** For data visualization.
- **caret:** For machine learning.

Skills Network Labs:

- **Virtual Environment:** A virtual RStudio environment is provided for practice, eliminating the need for account creation, downloads, or installations.

Git

In this video, you've learned about Git and GitHub, which are essential tools for version control and collaboration among developers and data scientists.

Git is a free and open-source distributed version control system that allows users to track changes to their documents and projects. It's distributed, meaning users worldwide can have their own copy of a project and sync changes with a remote server.

GitHub, one of the most popular web-hosted services for Git repositories, provides a platform for hosting and sharing code, as well as facilitating collaboration through features like forks and pull requests.

Before diving into Git and GitHub, it's important to understand some basic terms:

- SSH protocol: A secure method for remote login.
- Repository: Contains project folders set up for version control.
- Fork: A copy of a repository.
- Pull request: The process for requesting review and approval of changes.
- Working directory: Contains the files and directories associated with a Git repository.

Key Git commands include:

- `git init`: Initializes a new repository.
- `git add`: Moves changes from the working directory to the staging area.
- `git status`: Shows the state of the working directory and staged changes.
- `git commit`: Commits staged changes to the project.
- `git reset`: Undoes changes in the working directory.
- `git log`: Browses previous changes.
- `git branch`: Creates isolated environments for changes.
- `git checkout`: Views and switches between branches.

- `git merge`: Integrates changes from different branches.

To learn Git effectively and start collaborating, GitHub offers resources like cheat sheets and tutorials on try.github.io. In the upcoming modules, you'll get a crash course on setting up your local environment and initiating projects.

[Optional] Getting Started with Branches using Git Commands
You would typically use Git commands from your own desktop/laptop. However, so you can get started using the commands quickly without having to download or install anything, we are providing an IDE with a Terminal on the Cloud. Simply click the Open Tool button below to launch the Skills Network Cloud IDE and in the new browser tab that launches, follow the instructions to practice the Git commands. After completing this lab you will be able to use git commands to start working with creating and managing your code branches, including:

1. create a new local repository using **git init**
2. create and add a file to the repo using **git add**
3. commit changes using **git commit**
4. create a branch using **git branch**
5. switch to a branch using **git checkout**
6. check the status of files changed using **git status**
7. review recent commits using **git log**
8. revert changes using **git revert**
9. get a list of branches and active branch using **git branch**
10. merge changes in your active branch into another branch using **git merge**

If you are unable to open the lab or view it properly, please click here to view the HTML version full screen in a new browser tab.

Module 5 Summary

Congratulations! You have completed this module. At this point in the course, you know:

- The capabilities of R and its uses in Data Science.
- The RStudio interface for running R codes.
- Popular R packages for Data Science.
- Popular data visualization packages in R.
- Plotting with the inbuilt R plot function.
- Plotting with ggplot.
- Adding titles and changing the axis names using the ggtitle and lab’s function.
- A Distributed Version Control System (DVCS) keeps track of changes to code, regardless of where it is stored.
- Version control allows multiple users to work on the same codebase or repository, mirroring the codebase on their own computers if needed, while the distributed version control software helps manage synchronization amongst the various codebase mirrors.
- Repositories are storage structures that:
    - Store the code

    - Track issues and changes
    - Enable you to collaborate with others
- Git is one of the most popular distributed version control systems.
- GitHub, GitLab and Bitbucket are examples of hosted version control systems.
- Branches are used to isolate changes to code. When the changes are complete, they can be merged back into the main branch.
- Repositories can be cloned to make it possible to work locally, then sync changes back to the original.