VIBE PROJECT

Virtual Biomedical and STEM/STEAM Education

2021-1-HU01-KA220-HED-000032251

Co-funded by the European Union

Erasmus+

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

PÉCSI TUDOMÁNYEGYETEM
UNIVERSITY OF PÉCS

U.PORTO

Politechnika Śląska

DEX innovation centre

VIBE PROJECT

PATTERN RECOGNITION IN BIOMEDICAL ENGINEERING

SOFTWARE TOOLS AND IMPLEMENTATION

Lecture plan

- Software tools and implementation
- Introduction to software tools
- Libraries and frameworks for pattern recognition

Software Tools

MAIN SOFTWARE

ADDITIONAL SOFTWARE

Visual Studio Code



Lecture plan

- Software tools and implementation
- Introduction to software tools
- Libraries and frameworks for pattern recognition

Frameworks for pattern recognition

NumPy

Software tools

Introduction to Python and R for Data Analysis

Python and **R** have emerged as two of the most popular programming languages for data analysis. Both offer a rich ecosystem of libraries and tools, making them powerful and versatile choices for a wide range of data-driven tasks.

Choosing between Python and R often depends on personal preference and specific project requirements.

Python's versatility and general-purpose nature can be advantageous for projects that involve a mix of data analysis and other programming tasks.

R's statistical focus and specialized packages make it a strong choice for projects that require in-depth statistical analysis.

Ultimately, the best language for data analysis is the one that you are most comfortable with and that provides the necessary tools and libraries for your specific needs.

Introduction to Python and R for Data Analysis

Python is renowned for its readability and versatility. It's a general-purpose language that has gained significant traction in the data science community due to its extensive libraries like:

-NumPy: For numerical computations and arrays **-Pandas:** For data manipulation and analysis
-Scikit-learn: For machine learning algorithms

-Matplotlib: For data visualization

Python's versatility extends beyond data analysis, making it a valuable skill for various programming tasks.

R is specifically designed for statistical computing and data analysis. It provides a comprehensive set of statistical

functions and tools, making it a popular choice among statisticians and data scientists. Key R libraries include:

-dplyr: For data manipulation
-ggplot2: For data visualization
-caret: For machine learning
-tidyverse: A collection of packages for data science workflows

R's strengths lie in its statistical capabilities and its active community, which constantly contributes to new packages and advancements.

WinPython

WinPython is a free Python distribution for Windows that includes a comprehensive set of popular data science packages.

It's designed to provide a convenient and self-contained environment for: - data analysis, - scientific computing,

- and machine learning tasks.

Advantages of WinPython Over Official Distribution:

-Pre-installed Packages: WinPython comes with a wide range of pre-installed packages, including **NumPy, SciPy, Pandas, Matplotlib, Scikit-learn**, and more. This saves you the time and effort of manually installing and configuring these packages. **-Portable Environment:** WinPython is portable, meaning you can carry it on a USB drive and use it on any Windows computer without requiring installation. This makes it ideal for students, researchers, or anyone who needs to work on multiple machines. **-Multiple Python Versions:** WinPython offers support for multiple Python versions, allowing you to choose the version that best suits your project requirements. **-Package Manager:** WinPython includes a package manager, making it easy to install additional packages that are not included in the base distribution. **-Integrated Development Environment (IDE):** Some WinPython distributions include an integrated development environment (IDE), such as Spyder, which provides a userfriendly interface for coding and debugging.

Finding and Installing WinPython:

WinPython

1.Visit the WinPython website: Go to https://winpython.github.io/ or https://github.com/winpython/winpython

2.Download the appropriate version: Choose the version of Python and the set of packages that best suits your needs.

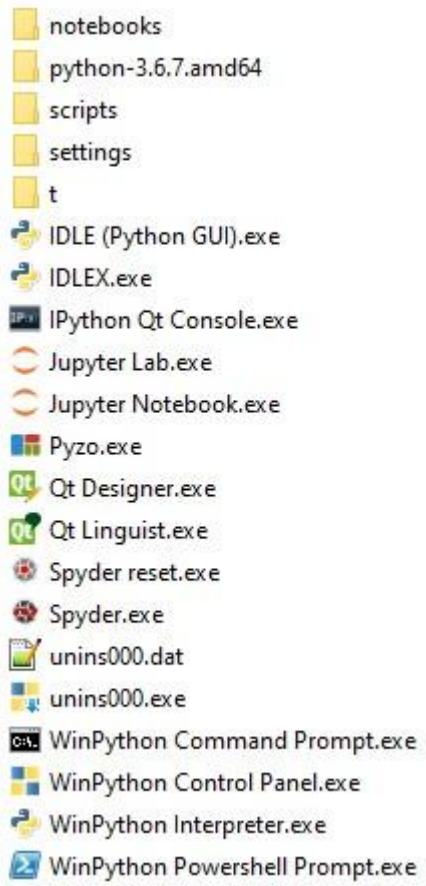
3.Run the installer: Double-click the downloaded installer file and follow the on-screen instructions.

4.Start using WinPython: Once the installation is complete, you can launch WinPython and start using the included packages.

By using WinPython, you can quickly and easily set up a powerful environment for data analysis and scientific computing on Windows.

Jupyter, JupyterLab, Spyder, Qt Designer, and Pyzo

WinPython



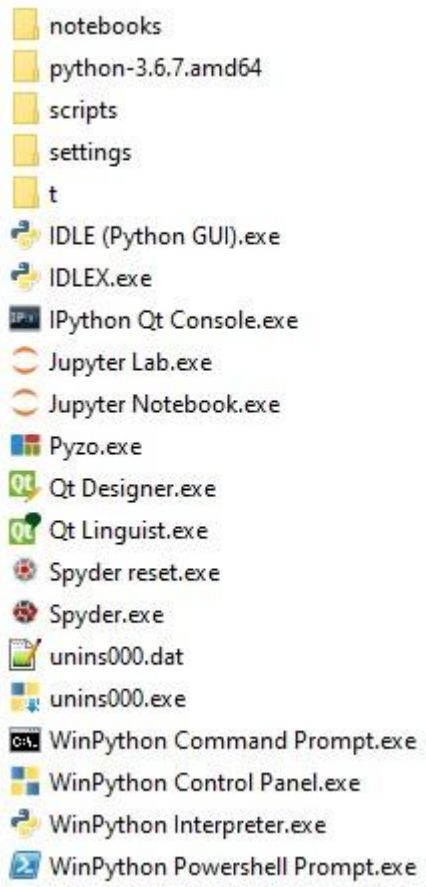
Jupyter and **JupyterLab** are popular interactive environments for data science and scientific computing, providing a web-based interface for creating and running notebooks.

- **Jupyter:** Offers a classic notebook interface with cells for code, text, and visualizations.
- **JupyterLab:** A more modern and modular interface with multiple tabs and panels, allowing for more complex workflows.

Spyder is a powerful scientific Python IDE that provides a comprehensive set of features for data analysis, debugging, and visualization. **Spyder** offers a variable explorer, debugger, and plotting tools, making it a great choice for those who prefer a traditional IDE-like experience.

Jupyter, JupyterLab, Spyder, Qt Designer, and Pyzo

WinPython



Qt Designer is a GUI builder tool that allows you to create graphical user interfaces (GUIs) using the Qt framework. **Qt Designer** is often used to create custom GUIs for Python applications, but it's not specifically designed for data science or scientific computing.

Pyzo is another Python IDE that combines the features of Jupyter and Spyder, offering a hybrid approach with both notebook-style and IDE-style interfaces. **Pyzo** provides a variable explorer, debugger, and plotting tools, while also allowing you to create and run notebooks.

GIT

Git is a free and open-source version control system that is widely used by developers to track changes to their code over time. It is a distributed version control system, which means that each developer has a full copy of the repository on their local machine. This allows for efficient collaboration and offline work.

https://git-scm.com/

Git is a complex tool with many features. However, it is relatively easy to learn the basics. There are many resources available online to help you get started with Git.

GIT

Git is a powerful tool that can be used to manage projects of all sizes. It is particularly useful for projects that involve multiple developers or that need to be versioned over time.

Here are some of the key features of Git:

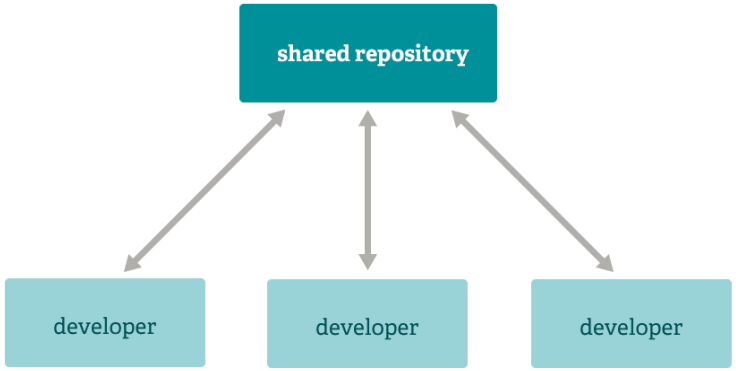
- **Version control:** Git allows you to track changes to your code over time and revert to previous versions if necessary.

GIT

- **Collaboration:** Git makes it easy for multiple developers to work on the same project simultaneously.

- **Branching and merging:** Git allows you to create branches of your code, which can be used to experiment with new features or fix bugs without affecting the main codebase.

- **Distributed:** Git is a distributed version control system, which means that each developer has a full copy of the repository on their local machine. This allows for efficient collaboration and offline work.



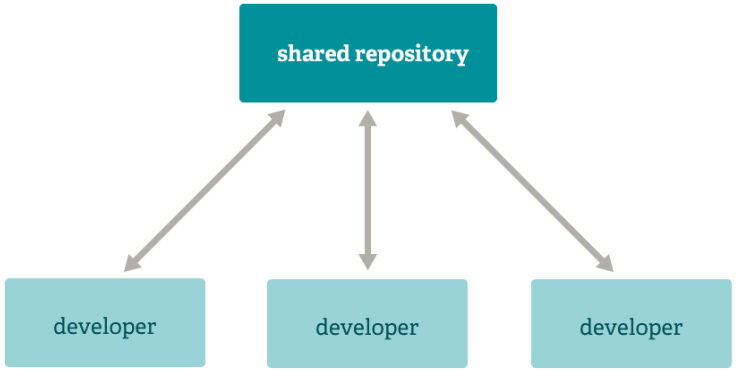
GIT

Distributed Git means that every developer has a complete copy of the entire repository, including its history. This is different from centralized version control systems like SVN, where there is a single central server that stores the repository, and developers only have a working copy.

In essence, distributed Git empowers developers with more flexibility, independence, and resilience.

https://git-scm.com/about/distributed

GIT



Key advantages of distributed Git:

-Offline work: Developers can work on their local repositories even without an internet connection.

-Faster operations: Many Git operations, such as committing and branching, can be performed locally, without needing to communicate with a remote server.

-Increased reliability: If the central server goes down, developers can still access their local repositories and continue working.

GIT

-Better collaboration: Distributed Git makes it easier for teams to collaborate, as each member has a full copy of the repository and can contribute changes independently.

https://git-scm.com/about/distributed

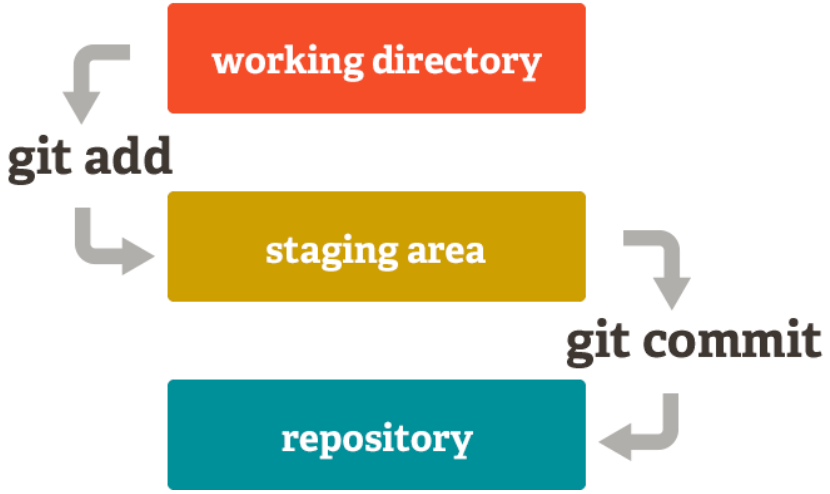
Staging Area in Git

The staging area in Git is a temporary storage area where you prepare changes to be committed. It acts as a buffer between your working directory (where you make changes to your files) and the repository's commit history.

Here's how it works:

- **Make changes:** You modify files in your working directory.
  Stage changes: Use the git add command to add specific files or all changes to the staging area. This marks them as ready to be committed.

- **Commit changes:** Use the git commit command to create a new commit, taking the staged changes from the staging area and adding them to the repository's history.



https://git-scm.com/about/staging-area

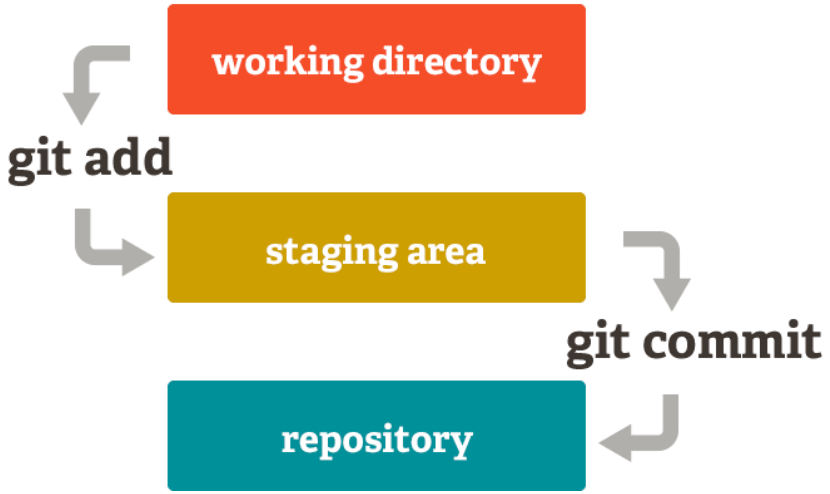
GIT

Staging Area in Git

Here are the basic commands for using the staging area in Git:

Adding files to the staging area:

git add <filename>: Adds a specific file to the staging area. git add .: Adds all changes in the current directory to the staging area.



Removing files from the staging area:

git reset HEAD <filename>: Removes a specific file from the staging area. git reset HEAD .: Removes all changes from the staging area.

Checking the staging area:

git status: Shows the current status of the repository, including which files are staged, unstaged, and untracked.

Staging Area in Git

https://git-scm.com/about/staging-area

GIT

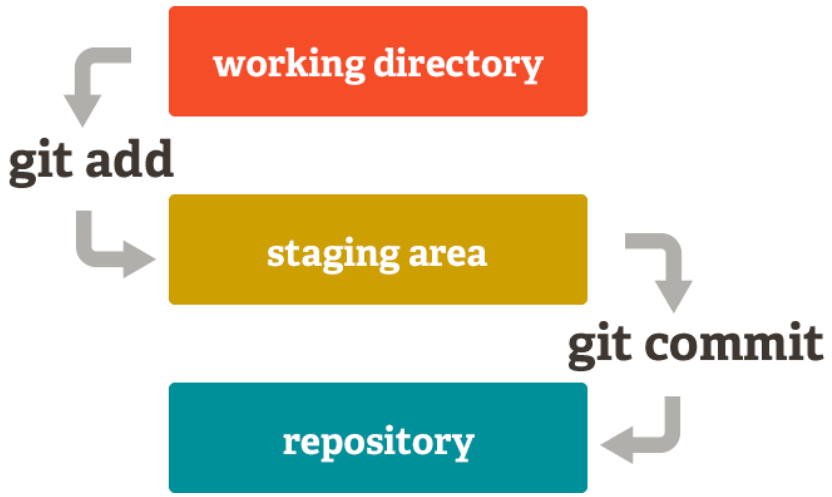
```
# Example:

# Make some changes to a file # Stage
the changes

>git add my_file.txt

# Check the staging area git status
# Remove the file from the staging
area

>git reset HEAD my_file.txt
```



https://git-scm.com/about/staging-area

GIT

Example:

```
>cd my-project
>git init
>git add .
>git commit -m "Initial commit"
>git remote add origin https://github.com/your-username/my-project.git
>git push origin master
```

Explanation of Commands:

- **git init:** Initializes a new Git repository in the current directory.
- **git add <filename>:** Stages the specified file for the next commit.
- **git commit -m "message":** Creates a new commit with the specified message.
- **git remote add origin <url>:** Adds a remote repository named "origin" with the specified URL.
- **git push origin master:** Pushes the current branch ("master") to the remote repository named "origin".

Example: Creating a Git Repository…

1. Initialize a Git Repository:

GIT

- **Navigate to your project directory:** Open your terminal or command prompt and use the cd command to change to the directory where you want to create your repository.
- **Initialize a Git repository:** Run the following command:

```
>git init
```

This creates a hidden .git directory in your project directory, which contains all the necessary files for Git to track changes.

2. Add Files to the Repository:

Stage files: Use the git add command to stage files that you want to include in your next commit:

```
>git add <filename>
```

You can also add all files in the current directory using git add ..

…Creating a Git Repository

3. Commit Changes:

- **Create a commit:** Use the git commit command to create a commit with a message describing the changes:

```
>git commit -m "Initial commit"
```

Replace "Initial commit" with a meaningful message that explains the changes you've made.

4. Create a Remote Repository (Optional):

- **If you want to collaborate with others or have a backup of your repository, create a remote repository on a platform like GitHub, GitLab, Bitbucket or Heroku.**
- **Add the remote repository:** Use the git remote add command to add a remote repository to your local repository:

```
>git remote add origin <remote_repository_url>
```

Replace <remote_repository_url> with the URL of your remote repository.

5. Push Changes to the Remote Repository (Optional):

- **Push your local commits to the remote repository:** Use the git push command:

```
>git push origin master
```

Replace "master" with the name of the branch you want to push to repository named "origin".

VSC

Visual Studio Code

Visual Studio Code (VSCode) is a popular and versatile code editor developed by Microsoft. It's designed to provide a powerful and customizable environment for developers working with a variety of programming languages, including Python.

Advantages of VSCode for Data Analysis and Machine Learning in Python:

- **Rich Ecosystem of Extensions:** VSCode boasts a vast marketplace of extensions that can significantly enhance its capabilities for data analysis and machine learning.

These extensions offer features like:

- Code completion and auto-suggestions
- Debugging tools
- **Integration with popular libraries like NumPy, Pandas, Matplotlib, and Scikit-learn**
- **Jupyter Notebook support**
- **Git integration**

- **Customization:** VSCode is highly customizable, allowing you to tailor the environment to your preferences. You can customize themes, fonts, keyboard shortcuts, and more.

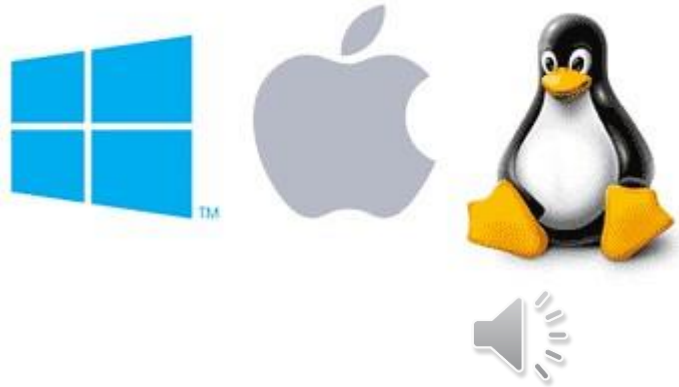
VSC

Visual Studio Code

Visual Studio Code

- **Lightweight and Fast:** VSCode is known for its speed and performance, even when working with large projects. This makes it a great choice for data analysis and machine learning tasks that often involve working with large datasets.

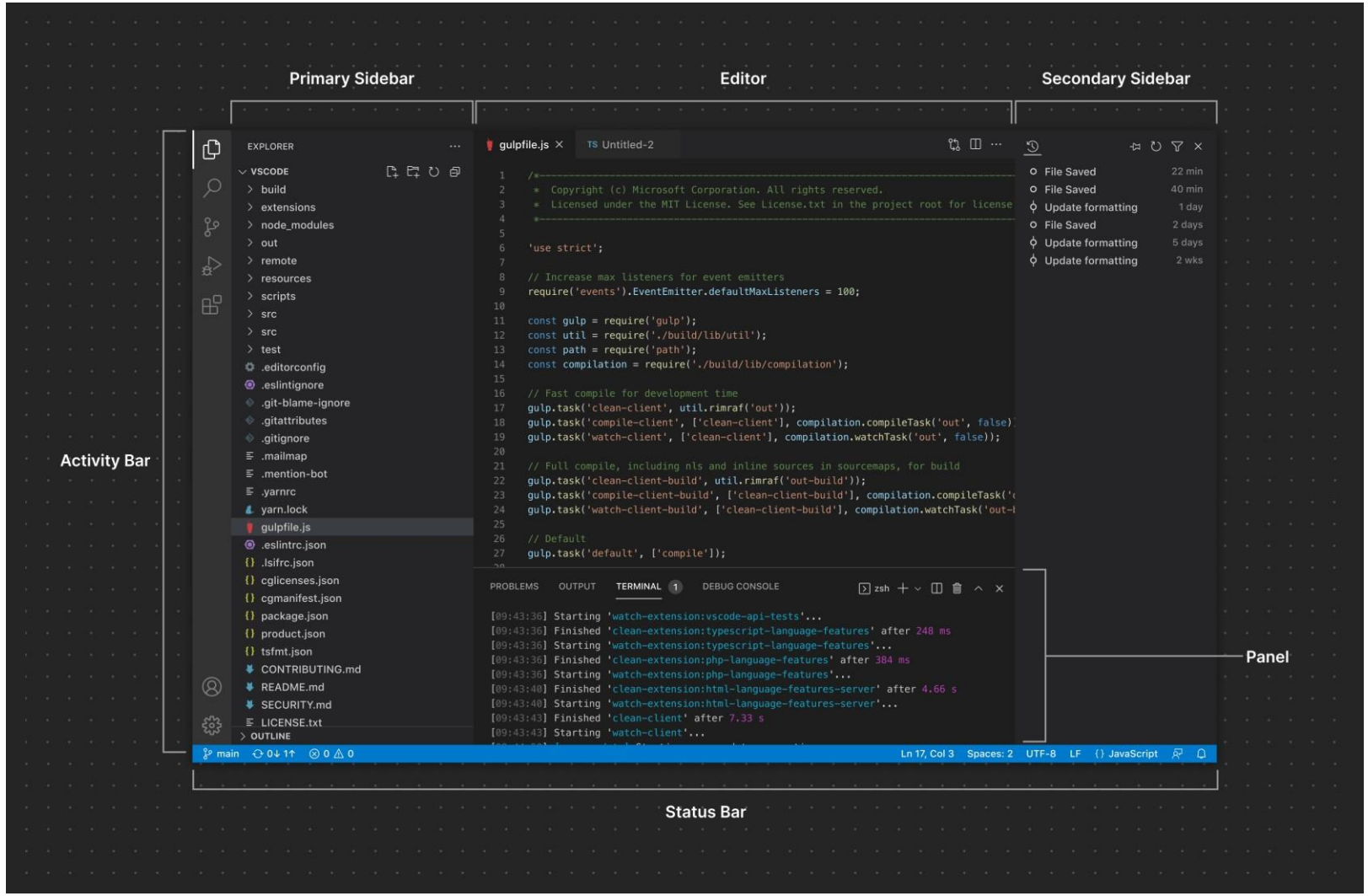
- **Cross-Platform Compatibility:** VSCode runs on Windows, macOS, and Linux, making it a versatile choice for developers working on different operating systems.



- **Integration with Other Tools:** VSCode integrates well with other tools and services commonly used in data science and machine learning, such as GitHub, Docker, and cloud platforms.

VSC

Visual Studio Code User Interface

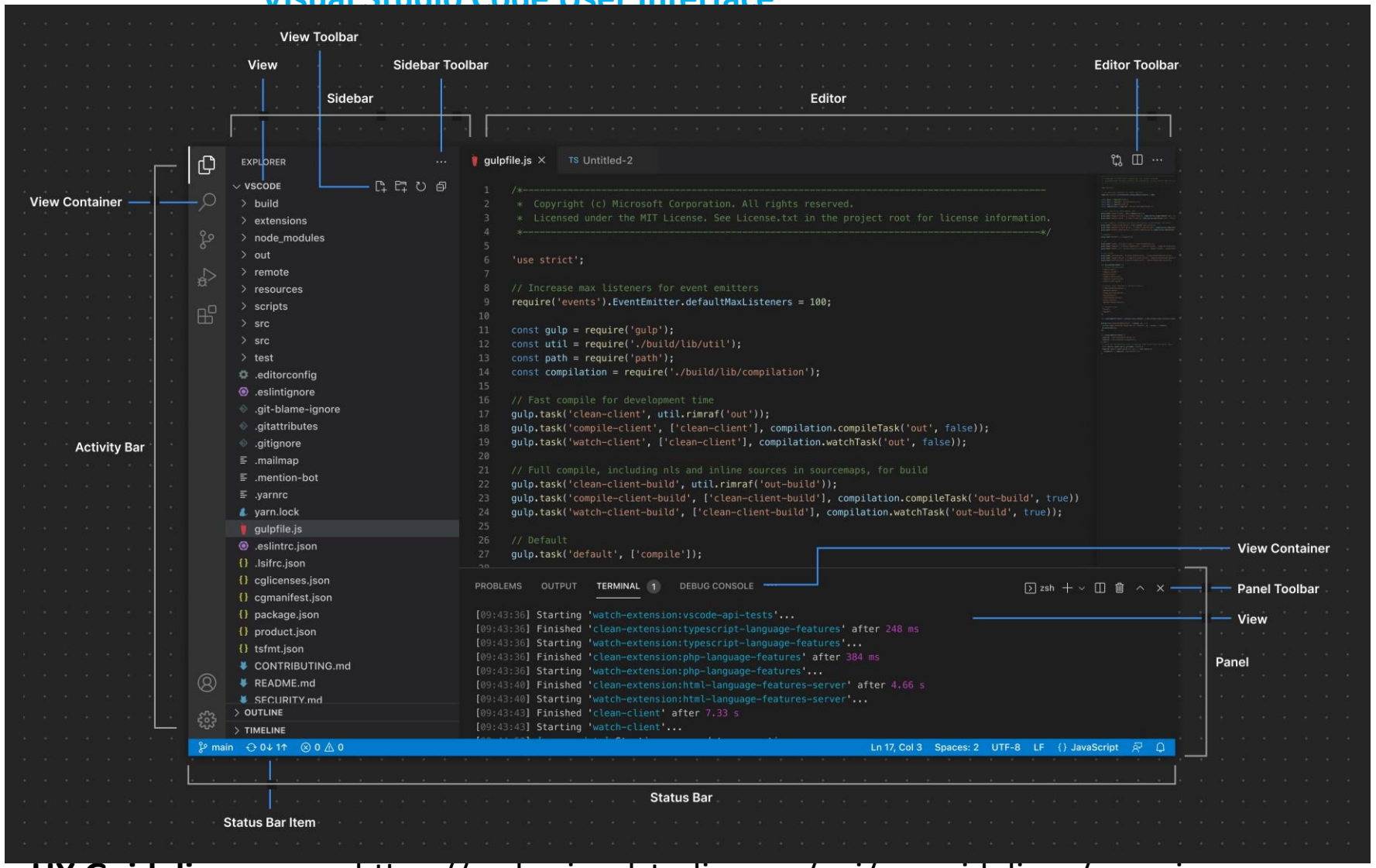


UX Guidelines https://code.visualstudio.com/api/ux-guidelines/overview

VSC

Visual Studio Code

Visual Studio Code User Interface

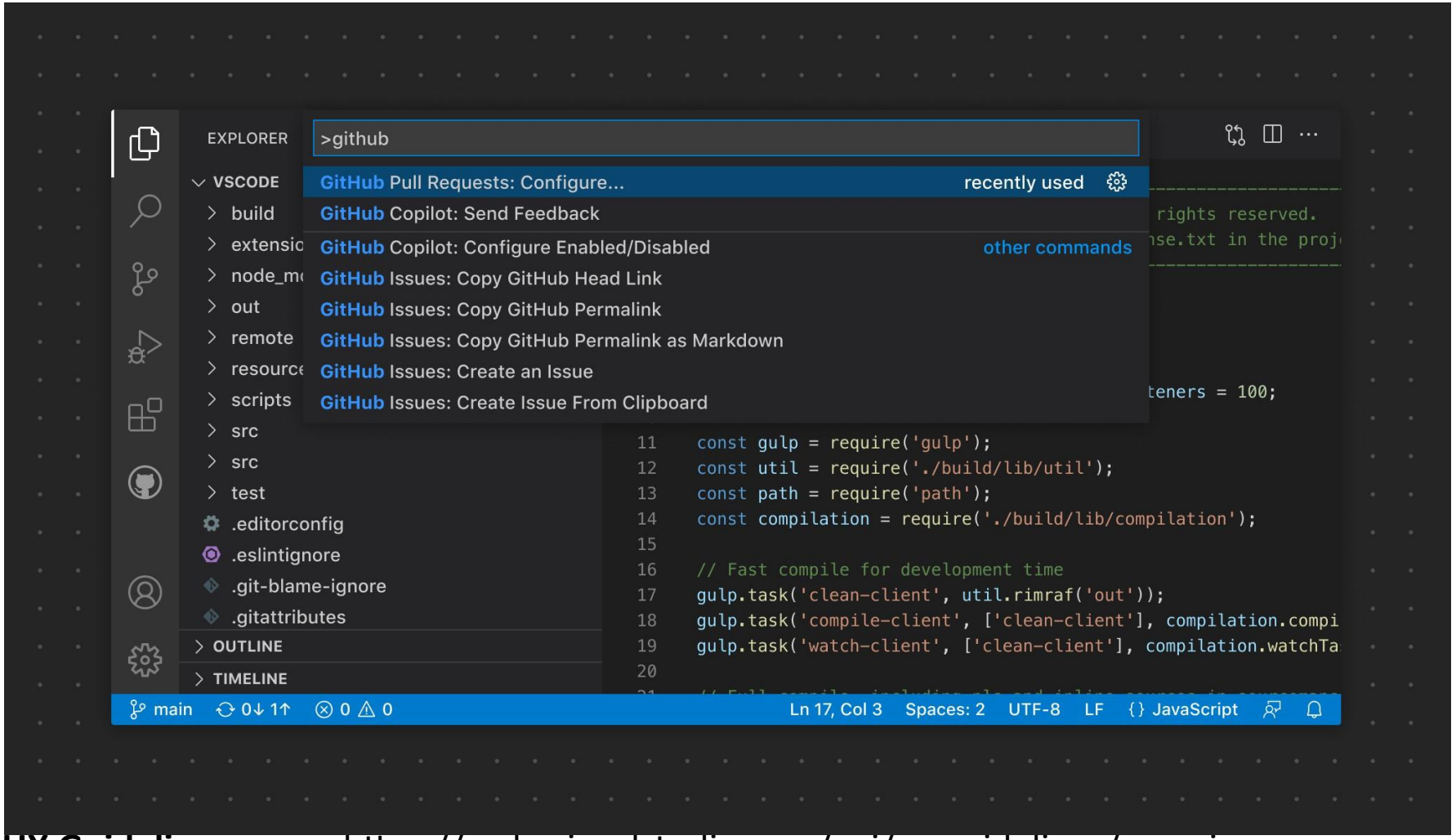


UX Guidelines https://code.visualstudio.com/api/ux-guidelines/overview

VSC

Visual Studio Code User Interface

Visual Studio Code



Command Palette

UX Guidelines https://code.visualstudio.com/api/ux-guidelines/overview

VSC

Visual Studio Code User Interface

Visual Studio Code (VSCode) has a clean and intuitive user interface, designed to enhance productivity and streamline coding workflows. Here are the main elements of the VSCode user interface:

1. **Title Bar:** Located at the top of the window, the title bar displays the name of the open file or project, the VSCode icon, and basic window controls (minimize, maximize, close).

2. **Menu Bar:** The menu bar, found directly below the title bar, contains menus for various actions and settings, such as File, Edit, View, Go, Run, Debug, Terminal, Help, and Extensions.

3. **Activity Bar:** The activity bar is located on the left side of the window and provides quick access to common tasks like opening files, searching, debugging, and managing extensions.

4. **Explorer:** The Explorer panel, typically found on the left side of the window, displays the file structure of your project, allowing you to navigate through files and folders.

VSC

Visual Studio Code User Interface

5. Editor Area: This is the main workspace where you write and edit your code. It can display multiple files simultaneously in tabs.

https://code.visualstudio.com/docs/getstarted/userinterface

UX Guidelines https://code.visualstudio.com/api/ux-guidelines/overview

Visual Studio Code User Interface

Visual Studio Code

6. **Status Bar:** Located at the bottom of the window, the status bar provides information about the current file, line number, character position, and other relevant details. It also displays indicators for language mode, Git status, and other settings.

7. **Sidebar:** The sidebar, which can be toggled on or off, provides additional panels for tasks like searching, debugging, and version control.

8. **Extensions Panel:** This panel, accessible from the Activity Bar, allows you to browse, install, and manage extensions that add new features and functionality to VSCode.

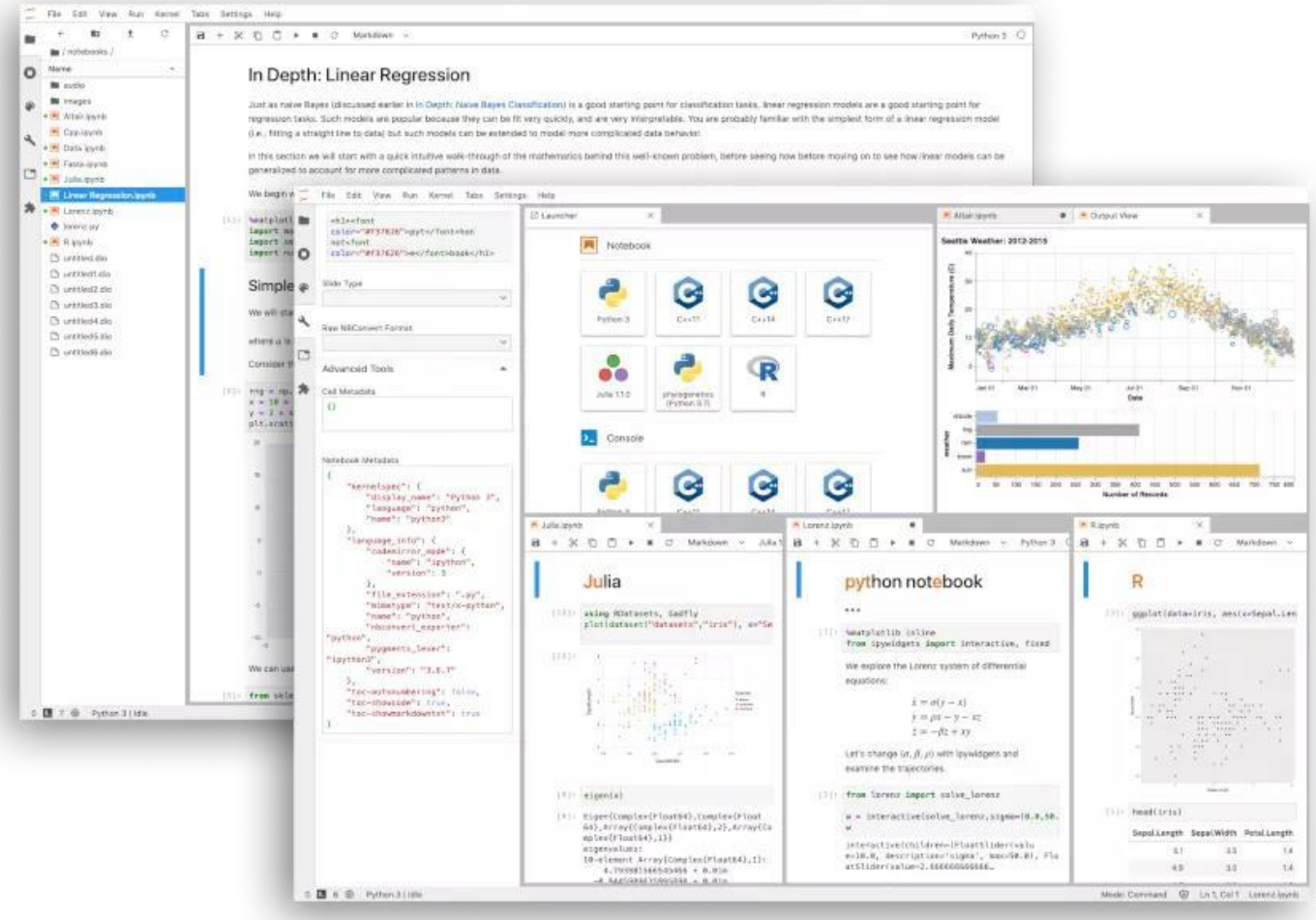
9. **Command Palette:** The Command Palette, accessed by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS), provides a quick way to search for and execute commands within VSCode.

These elements work together to create a customizable and efficient coding environment that can be tailored to your specific needs and preferences.

Jupyter

https://code.visualstudio.com/docs/getstarted/userinterface

Jupyter



https://jupyter.org/

Jupyter

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It's a popular tool for data scientists, researchers, and educators due to its interactive nature and ability to combine code with explanatory text.

Key Features of Jupyter Notebook:

-Interactive Execution: Run code cells one by one or all at once, providing immediate feedback and making it easy to experiment and iterate.

-Multiple Programming Languages: Supports a wide range of programming languages, including Python, R, Julia, Scala, and many more.

-Rich Text Formatting: Use Markdown to format text, add headings, lists, images, and links, creating well-structured and informative documents.

-Mathematical Equations: Write equations using LaTeX syntax, making it ideal for scientific and technical work.

-Data Visualization: Create plots, charts, and other visualizations using libraries like Matplotlib, Seaborn, and Plotly.

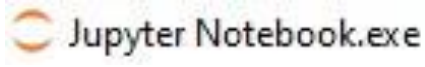
-Collaboration: Share notebooks with others and collaborate in real-time, making it a great tool for teamwork and knowledge sharing.

Jupyter

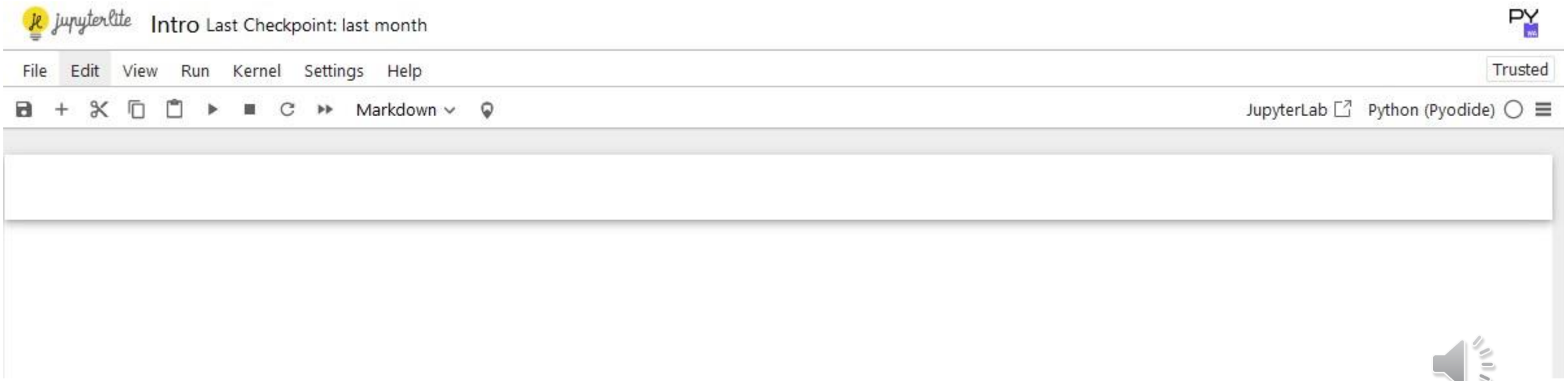
You can test Jupyter online! Without installation!

BTW. If you use WinPython then you have Jupyter ready to use.

Just start it by double-click on „Jupyter Notebook.exe” file in WinPython folder



Jupyter



https://jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb

Frameworks
for data analysis and pattern recognition

Pandas is a powerful Python library for data analysis and manipulation.

It provides data structures and operations for efficiently handling large datasets, making it a valuable tool for data scientists, researchers, and analysts.

Key features:

-Series: One-dimensional labeled array of data, similar to NumPy's array.

-DataFrame: Two-dimensional labeled data structure with rows and columns, similar to a spreadsheet.

-Indexing and selection: Flexible methods for accessing and selecting data based on labels or positions.

-Data cleaning and preparation: Functions for handling missing values, duplicates, and outliers.

-Data analysis: Tools for statistical analysis, aggregation, grouping, and filtering.

-Visualization: Integration with plotting libraries like Matplotlib for creating informative charts and graphs.

Basic example:

```
# Import library
import pandas as pd

# Create a Series s =
pd.Series([1, 2, 3, 4])

# Create a DataFrame df = pd.DataFrame({'A': [1, 2, 3],
'B': ['a', 'b', 'c']})

# Access data
print(s[0])
print(df['A'])

# Perform operations print(s.sum())  print(df.mean())
```

Scikit-learn is a powerful and versatile Python library for **machine learning**.

It provides a wide range of algorithms and tools for building and training various machine learning models, making it a popular choice for data scientists and researchers.

It provides a comprehensive set of supervised and unsupervised learning algorithms, covering areas such as:

-Classification: Identifying which category an object belongs to.
-Regression: Predicting a continuous-valued attribute associated with an object.
-Clustering: Automatic grouping of similar objects into sets, with models like k-means.
-Dimensionality Reduction: Reducing the number of attributes in data for summarization, visualization and feature selection, with models like Principal Component Analysis (PCA).
-Model Selection: Comparing, validating and choosing parameters and models.

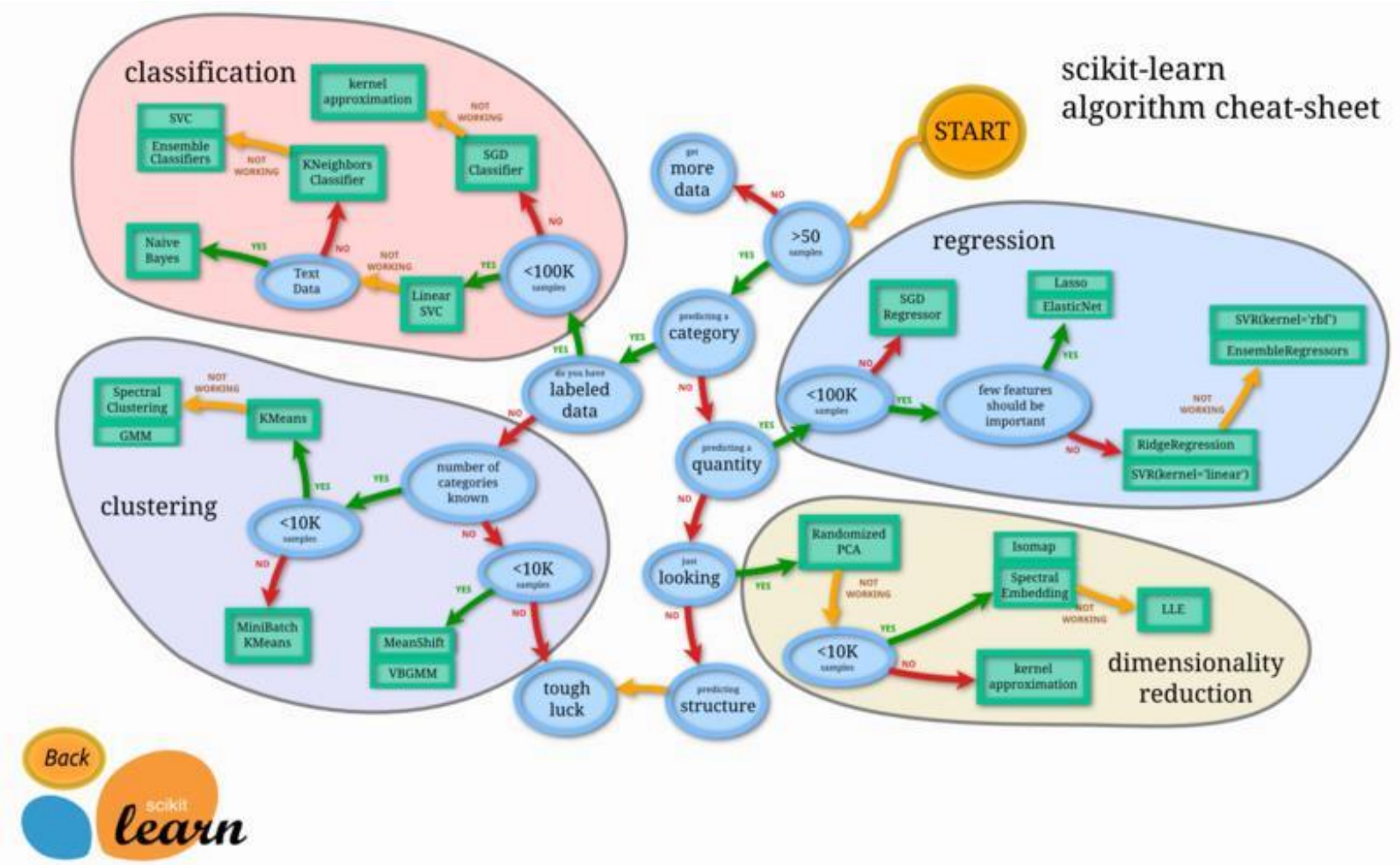
Source:

https://domino.ai/data-science-

-Pre-processing: Feature extraction and normalization, including defining

arn%3F,data%20modeling%20library%20for%20Pythodictionary/sklearn#:~:text=What%20is%20Scikit%2Dle attributes in image and text data. n.

Machine learning algorithms implemented in Scikit-Learn


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
Text Data
NOT WORKING
Linear SVC
YES
<100K samples
YES
NO
more data
NO
>50 samples
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
predicting a category
YES
NO
<100K samples
YES
few features should be important
YES
NO
NOT WORKING
RidgeRegression
SVR(kernel='linear')
do you have labeled data
YES
NO
predicting a quantity
YES
NO
clustering
Spectral Clustering
GMM
NOT WORKING
KMeans
number of categories known
YES
YES
<10K samples
NO
NO
<10K samples
YES
NO
MiniBatch KMeans
MeanShift
VBGMM
just looking
YES
NO
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
<10K samples
YES
NO
kernel approximation
dimensionality reduction
tough luck
predicting structure
Back
scikit learn

https://en.wikipedia.org/wiki/Scikit-learn

https://pypi.org/project/pandasgui/
github.com/fonsecajr/pandasgui

PandasGUI

PandasGUI is not a standard package, nor is it part of the Pandas library. To use it you have to install it for yourself.

Here is where you can find it:

github.com/fonsecajr/pandasgui

https://pypi.org/project/pandasgui/

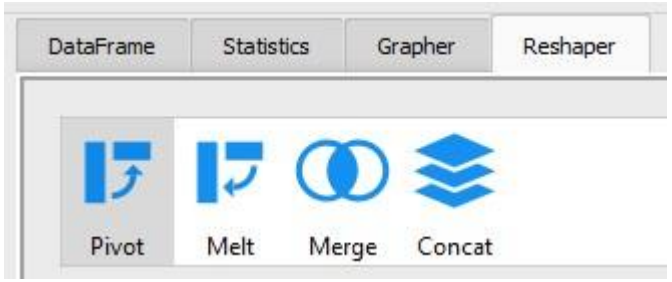
Pandas GUI

This package provides Graphical User Interface (GUI) tools to help Jupyter users generate code to analyze, plot and fit tabulated data that has been loaded into Pandas DataFrames.

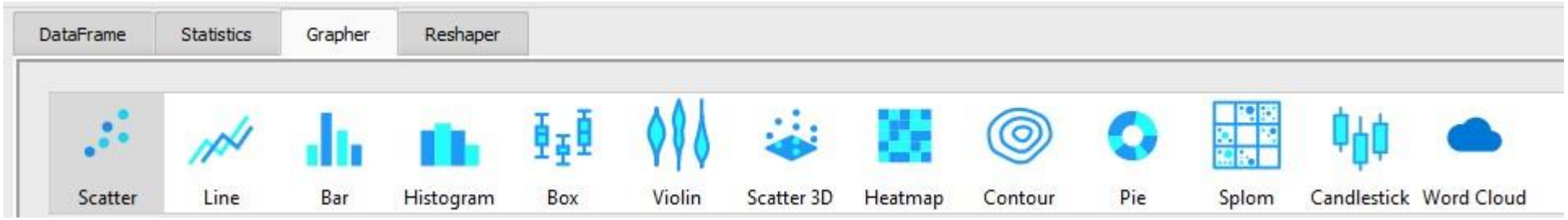
You can see: Pandas website for more about Pandas.

https://jupyterphysscilab.github.io/jupyter_Pandas_GUI/Pandas_GUI_Doc_Home.html

Menu optionsfor



PandasGUIPandasGUI provides the user with many useful tools visual data analysis. It is a very good tool for EDA (Exploratory Data Analysis).



DataFrame | Statistics | Grapher | Reshaper

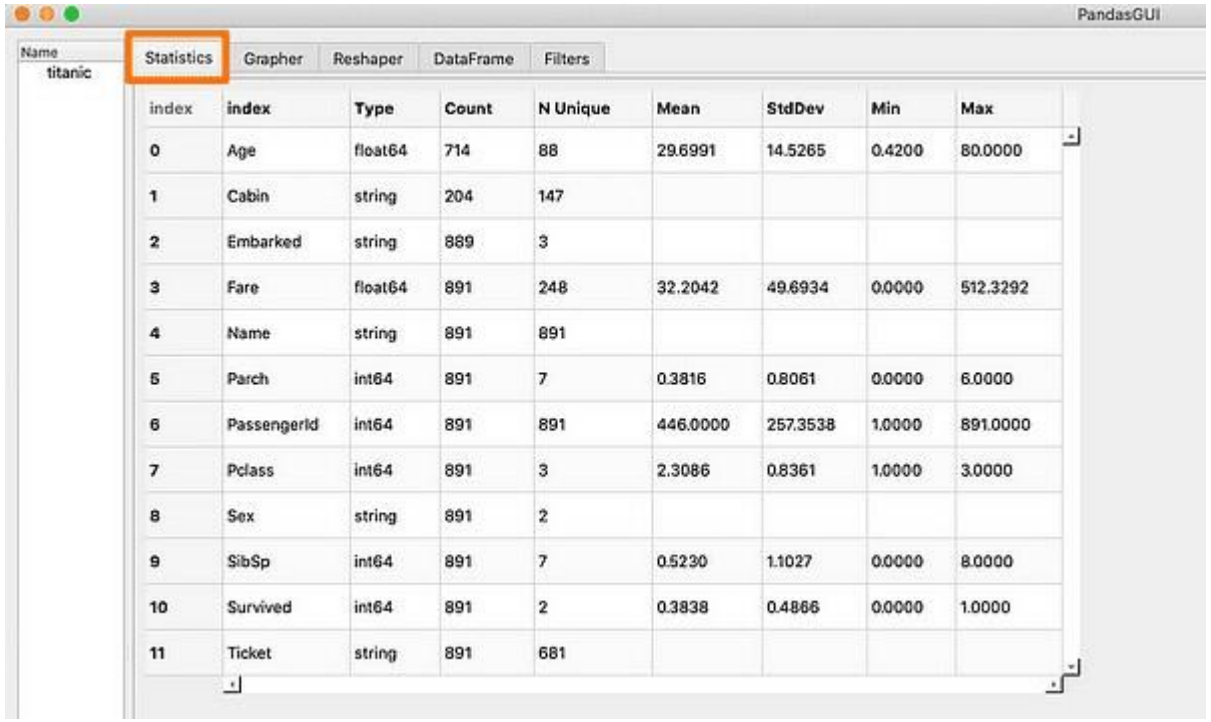
index	Type	Count	N Unique	Mean	StdDev	Min	Max

DataFrame | Statistics | Grapher | Reshaper

index	ıo_eq	zao_casc	ao_li	ao_vol	ao_eq	ao_casc	lm_years	lad_years	cx_years	rca_years	lm_wiek_start
0	0	0.0	0.0	0.0	0.0	0.0	•	•	•	•	•

PandasGUI Capabilities

- Viewing and sorting DataFrames
- Reshaping DataFrames
- DataFrame filtering
- Summary statistics •Interactive plotting



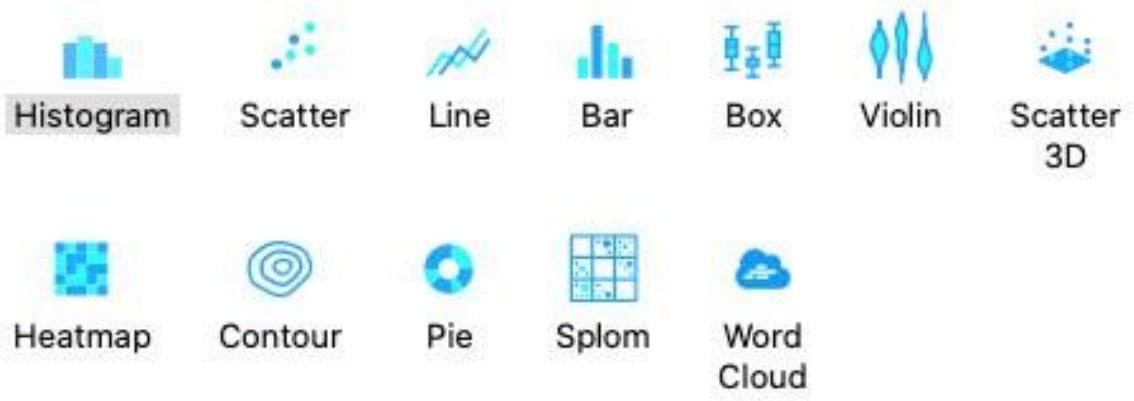
index	index	Type	Count	N Unique	Mean	StdDev	Min	Max
0	Age	float64	714	88	29.6991	14.5265	0.4200	80.0000
1	Cabin	string	204	147				
2	Embarked	string	889	3				
3	Fare	float64	891	248	32.2042	49.6934	0.0000	512.3292
4	Name	string	891	891				
5	Parch	int64	891	7	0.3816	0.8061	0.0000	6.0000
6	PassengerId	int64	891	891	446.0000	257.3538	1.0000	891.0000
7	Pclass	int64	891	3	2.3086	0.8361	1.0000	3.0000
8	Sex	string	891	2				
9	SibSp	int64	891	7	0.5230	1.1027	0.0000	8.0000
10	Survived	int64	891	2	0.3838	0.4866	0.0000	1.0000
11	Ticket	string	891	681				

https://www.kdnuggets.com/2023/06/revolutionizing-data-analysis-pandasgui.html
https://github.com/fonsecajr/pandasg ui

Interactive plotting

Finally, PandasGUI provides powerful interactive plotting options for your dataset which includes:

- Histogram
- Scatter plotting
- Line plotting
- Bar plotting
- Box plotting
- Violin plotting
- 3D scatter plotting
- Heatmap
- Contour plots
- Pie plots
- Splom plot

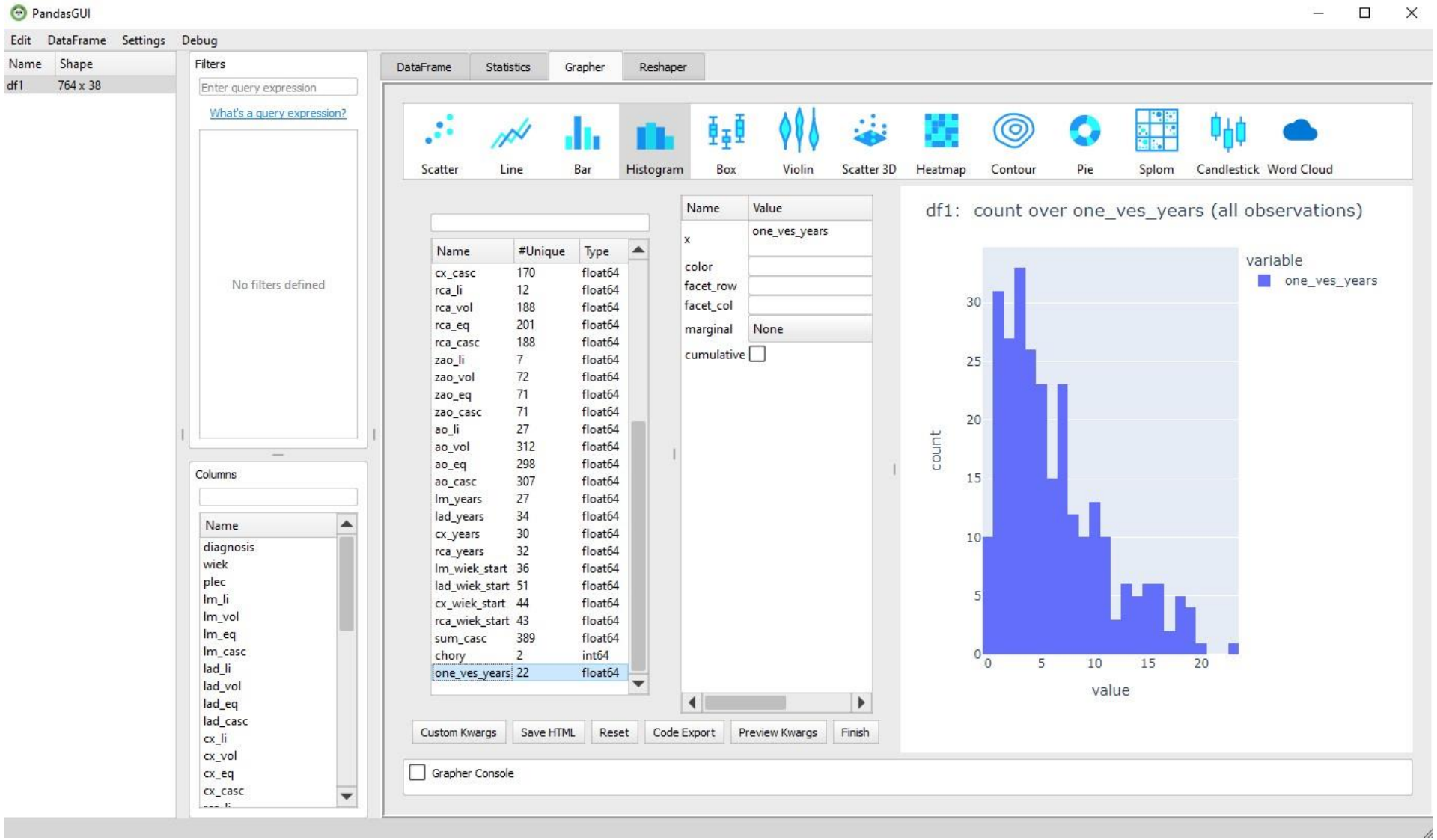


- Word cloud

PandasGUI

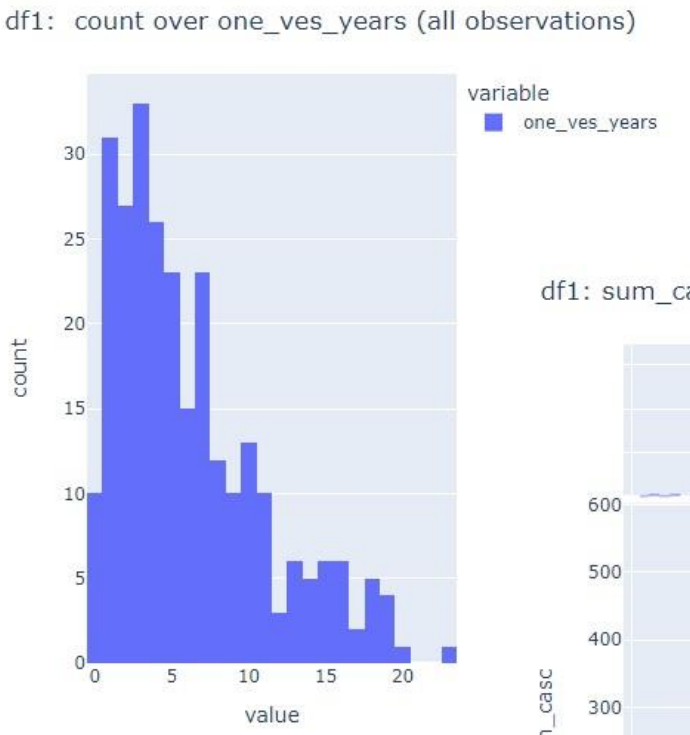
Example Screenshot

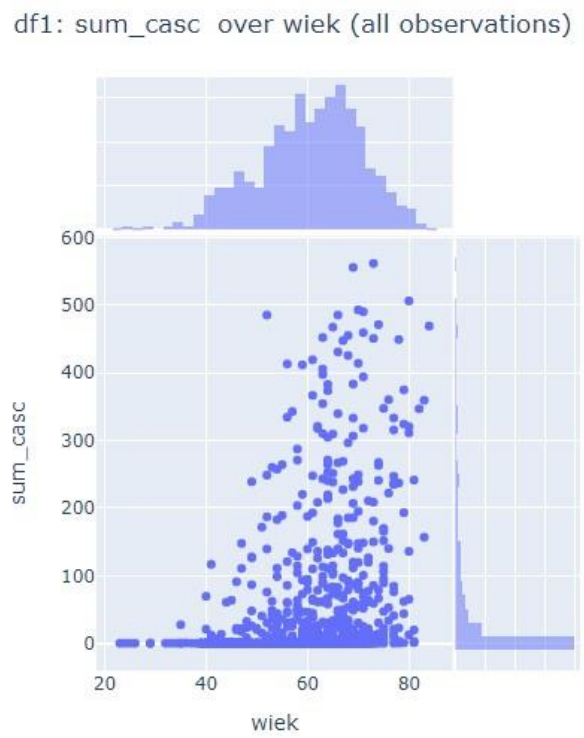
PandasGUI

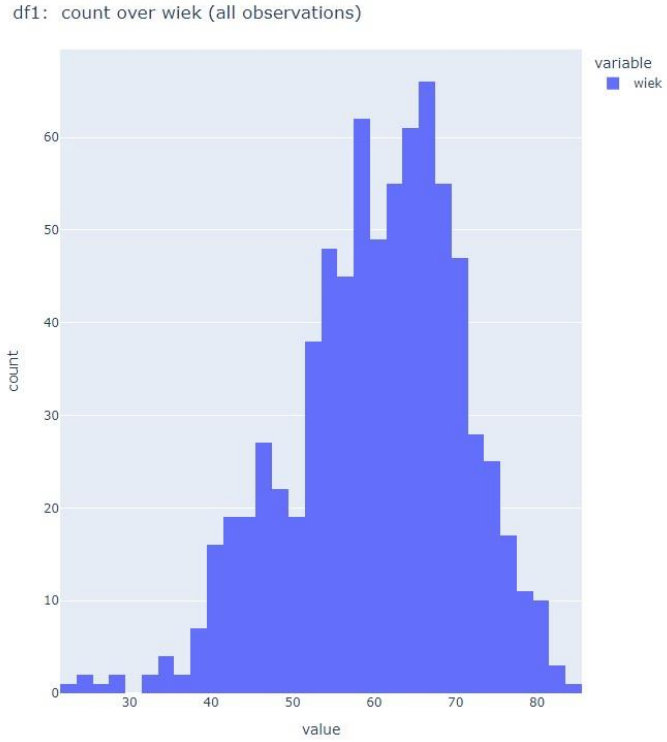

PandasGUI
Edit
DataFrame
Settings
Debug
Name
Shape
df1
764 x 38
Filters
Enter query expression
What's a query expression?
No filters defined
Columns
Name
diagnosis
wiek
plec
lm_li
lm_vol
lm_eq
lm_casc
lad_li
lad_vol
lad_eq
lad_casc
cx_li
cx_vol
cx_eq
cx_casc
DataFrame
Statistics
Grapher
Reshaper
Scatter
Line
Bar
Histogram
Box
Violin
Scatter 3D
Heatmap
Contour
Pie
Splom
Candlestick
Word Cloud
Name
#Unique
Type
cx_casc 170 float64
rca_li 12 float64
rca_vol 188 float64
rca_eq 201 float64
rca_casc 188 float64
zao_li 7 float64
zao_vol 72 float64
zao_eq 71 float64
zao_casc 71 float64
ao_li 27 float64
ao_vol 312 float64
ao_eq 298 float64
ao_casc 307 float64
lm_years 27 float64
lad_years 34 float64
cx_years 30 float64
rca_years 32 float64
lm_wiek_start 36 float64
lad_wiek_start 51 float64
cx_wiek_start 44 float64
rca_wiek_start 43 float64
sum_casc 389 float64
chory 2 int64
one_ves_years 22 float64
Name
Value
x
one_ves_years
color
facet_row
facet_col
marginal
None
cumulative
df1: count over one_ves_years (all observations)
variable
one_ves_years
count
value
0
5
10
15
20
25
30
Custom Kwargs
Save HTML
Reset
Code Export
Preview Kwargs
Finish
Grapher Console

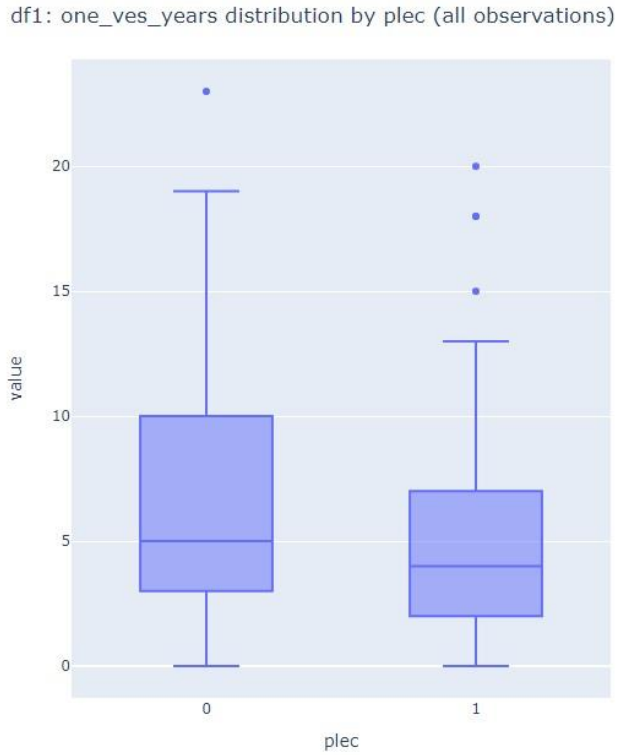
PandasGUI

Example Plots (histograms, boxplots)









PandasGUI

You can use colors to represent the “extra dimension” of the data.

PandasGUI

Name	Value
y	sum_casc
x	wiek
color	one_ves_years
facet_row	
facet_col	
symbol	
size	
trendline	None
marginal	None