Ensembler Python GUI Project
Program Documentation

Mr. Mohamed Mahmoud

2024-08-31

Introduction

This document provides a comprehensive explanation of the Python GUI program developed as a conversion from the bioMechZoo program originally in MATLAB. The GUI is divided into three main sections: the Left Section, Middle Section, and Right Section. Each section is responsible for different aspects of the program's functionality. This document also provides guidance for future developers on how to add more features to each section.

How to Run the Program

To run the program, the user must execute the following command in the terminal:

Listing 1: Command to Run the Program

```
python3 ensembler.py
```

This command will call the `gui.py` file and launch the GUI. Make sure that you are in the directory containing the `ensembler.py` file before running the command.

Left Section: File Organization

The Left Section of the GUI is primarily responsible for organizing files. This section includes functionalities that allow users to organize data files based on different criteria such as participants, conditions, or other categories.

Overview

This section provides the interface and logic for organizing files. It allows users to categorize and move files into appropriate directories based on their naming conventions.

Adding More Functionalities

To add more functionalities to the Left Section, such as additional methods for organizing files, developers should:

- Modify the `organize_folder` function to include additional organization methods. For example, to organize files by date, you could add:

Listing 2: New Method for Organizing Files by Date

```
def organize_by_date(path):
    for filename in os.listdir(path):
        if not os.path.isdir(os.path.join(path, filename)):
            date_str = filename.split('_')[2]  # Assuming the date is the third
            date_folder = os.path.join(path, date_str)
            os.makedirs(date_folder, exist_ok=True)
            shutil.move(os.path.join(path, filename), os.path.join(date_folder,
```

- Update the `organize_folder` function to include a new option:

Listing 3: Updating organize$_{f}olderFunction$

```
def organize_folder(path, method):
    if method == "Participants":
        organize_by_participants(path)
    elif method == "Conditions":
        organize_by_conditions(path)
    elif method == "Date":
        organize_by_date(path)
    elif method == "Other":
        organize_by_other(path)
```

- Update the GUI layout to include new buttons or dropdowns that allow users to select the new organization methods.

Middle Section: Data Processing & Plotting

The Middle Section is the core of the GUI, handling data processing, visualization, and interaction. This section provides the graphical plot area where data is displayed after processing.

Overview

This section is responsible for drawing and updating plots based on the processed data. Users interact with this section to view and analyze data, apply filters, and modify plot parameters.

Adding More Functionalities

To extend the functionalities in the Middle Section, developers can:

- Add new data processing functions in the `processing` module. For instance, if you want to add a smoothing function:

Listing 4: New Smoothing Function in processing Module

```
def smooth_data(data, window_size=5):
    return np.convolve(data, np.ones(window_size)/window_size, mode='valid')
```

- Modify the `update_plot` function to handle additional types of data or to implement new visualization techniques:

Listing 5: Updating update$_plotFunction$

```
def update_plot(fig, ax, data, smooth=False):
    ax.clear()
    if smooth:
        data['y'] = smooth_data(data['y'])
    ax.plot(data['x'], data['y'])
    ax.set_xlabel("Time")
    ax.set_ylabel("Value")
    fig.canvas.draw()
```

- Enhance the user interaction capabilities by adding new controls or options within the plot area. For example, add a checkbox for smoothing:

Listing 6: Adding Checkbox for Smoothing in GUI

```
layout = [
    # Other GUI elements
    [sg.Checkbox('Smooth Data', key='smooth')],
    [sg.Button('Update Plot')],
    # Other GUI elements
]

event, values = window.read()

if event == 'Update Plot':
    smooth = values['smooth']
    update_plot(fig, ax, data, smooth=smooth)
```

Right Section: Settings & Utilities

The Right Section is where users can control various settings and utilities of the program. This section includes options for configuring data processing parameters, selecting organization methods, and triggering specific events.

Overview

This section allows users to fine-tune the program's behavior by adjusting settings and parameters. It also provides utilities for managing events and other auxiliary tasks.

Adding More Functionalities

To add more functionalities to the Right Section, developers should:

- Expand the `Settings` module to include new configurable parameters that can be adjusted via the GUI. For example, adding a parameter for plot color:

Listing 7: Adding Plot Color Parameter in Settings Module

```
class Settings:
    def __init__(self):
        self.plot_color = 'blue'  # Default color

    def set_plot_color(self, color):
        self.plot_color = color
```

- Implement additional event tracking or logging capabilities in the `EventTracker` class and provide corresponding GUI elements for user control. Example:

Listing 8: Adding Event Logging in EventTracker Class

```python
class EventTracker:
    def __init__(self):
        self.event_log = []

    def log_event(self, event):
        self.event_log.append(event)
        print(f"Event Logged: {event}")
```

- Update the GUI layout to accommodate new settings or utilities, ensuring that the user interface remains intuitive and accessible. For example, adding a dropdown for plot color:

Listing 9: Adding Dropdown for Plot Color in GUI

```python
layout = [
    # Other GUI elements
    [sg.Text('Plot Color'), sg.Combo(['blue', 'green', 'red'], key='plot_color'
    [sg.Button('Apply Settings')],
    # Other GUI elements
]

event, values = window.read()

if event == 'Apply Settings':
    settings.set_plot_color(values['plot_color'])
```

Conclusion

This document serves as a guide for understanding the current functionality of the Python GUI program and provides detailed instructions for future developers to extend and enhance the program's capabilities. By following the guidelines provided, developers can ensure that the program remains robust and adaptable to future requirements.