USE OF PYTHON AND COMPLETE BLOOD COUNT PARAMETERS FOR COST-EFFECTIVE THALASSEMIA SCREENING IN RESOURCE-LIMITED SETTINGS: DEVELOPMENT AND VALIDATION OF A SCREENING PROGRAM

ABHISHEK SAMANTA1, NANDAN BHATTACHARYYA2

1Department of Zoology, Panskura Banamali College, P.O., Panskura R.S., West Bengal, India. 2Department of Biotechnology, Panskura Banamali College, P.O., Panskura R.S., West Bengal, India.

*Corresponding author: Nandan Bhattacharyya; Email: bhattacharyya_nandan@rediffmail.com

Received: 22 May 2023, Revised and Accepted: 19 September 2023

INTRODUCTION

High-performance liquid chromatography (HPLC) is the gold standard for thalassemia screening, demonstrating its ability to detect hemoglobin variants accurately and with a high level of sensitivity. HPLC testing can be costly and may not be available in all areas, particularly in places that have limited resources. Therefore, it is necessary to consider other methods to screen for thalassemia [1].

A potentially viable replacement to HPLC-based screening is the use of erythrocyte indices from complete blood count (CBC) examination. The erythrocyte indices provide information about the size and shape of red blood cells, which can identify individuals who may be carriers of thalassemia. The CBC test is widely available and relatively inexpensive, making it a cost-effective alternative to HPLC-based screening [2].

Several studies have shown the efficiency of CBC-based testing as a method for screening for thalassemia. Weatherall and Clegg published one of the earliest studies on CBC-based screening for thalassemia in 1975. They found that using the mean corpuscular volume (MCV) from a CBC test had an acute sensitivity and specificity for identifying individuals with the beta-thalassemia trait [3]. Since then, there have been many other studies published on CBC-based screening for thalassemia [4,5].

Based on our previous study [6], we propose an alternative approach to exclude normal populations from chromatography-based thalassemia screening programs using decision tree method and multiple logistic regression with erythrocyte indices, such as MCV, mean corpuscular hemoglobin (MCH), and red blood cell distribution width (RDW), to identify normal individuals and exclude them from further testing. The study found that using erythrocyte indices or CBC parameters as a screening tool had an acute sensitivity and specificity for excluding normal populations with very high fidelity. The approach could be a cost-effective and efficient way to improve thalassemia screening programs, particularly in resource-limited settings [7].

To verify this strategy, the present study intends to validate the index with a large quantity of thalassemia samples and create two Python applications that can quickly remove HPLC samples and normal samples that are not needed with no false negatives. This may help streamline thalassemia screening programs and make them more accurate, especially in places where HPLC testing may not be accessible or cost-effective.

METHODS

Sample preparation

We obtained two sets of thalassemia reports, thal-1 and thal-2 (Supplementary data), from the School of Tropical Medicine in Kolkata with a total sample size of 386.

New screening process

A Python program was developed utilizing version 3.9 on a Windows operating system to perform analysis on CBC data and distinguish between normal blood samples and those not indicative of thalassemia. This separation allows for the identification of samples that can be excluded from further confirmation testing through HPLC.
Individual processing graphical user interface (GUI)
For individual processing, the program analyzed each sample separately. CBC data were input into the program, and the program generated an output box indicating whether the sample was normal or abnormal.

Batch processing GUI
For batch processing, the program was modified to accept multiple samples as input. The program generated a report for each sample in the batch, indicating whether it was normal or if a recommendation for an HPLC test was necessary.

Ethical considerations
The research conducted in this study received approval from the ethics committee of Panskura Banamali College. Before the collection of blood samples, all participants willingly provided informed consent.

RESULTS
From the two datasets, two and five samples were identified and indicated, respectively, from the thal-1 and thal-2 sample sets.

Batch file processing GUI
The program you provided is a Python script that uses the tkinter library to create a GUI that allows users to select and process data from a CSV file. Here is a line-by-line explanation of the script:

```python
import tkinter as tk
import pandas as pd
from tkinter import filedialog

window = tk.Tk()
window.title('Filter Data')
Window.geometry('800x600')

This creates the main window of the GUI using tkinter. The window is given a title and dimensions.

Filepath_label = tk.Label(window, text='File Path')
Filepath_label.pack()
Filepath_entry = tk.Entry(window)
Filepath_entry.pack()
Browse_button = tk.Button(window, text='Browse', command=browse_file)
Browse_button.pack()

This creates the input area for the file path. A label and entry field are created using tk. Label() and tk. Entry(), respectively. A button is also created using tk. Button() that calls the browse_file() function when clicked. These widgets are then packed into the GUI window using pack().

Process_button = tk.Button(window, text='Process Data', command=process_data)
Process_button.pack()

This creates a button for processing the data using tk. Button(). It calls the process_data() function when clicked and is also packed into the GUI window.

Output_label = tk.Label(window, text='*May exclude from HPLC test')
Output_label.pack()
Output_text = tk.Text(window)
Output_text.pack()

This creates the output area for displaying the processed.

Individual processing GUI
This Python program is a GUI that allows users to input values for four parameters related to a blood test and calculates a result based on those inputs. Here is a breakdown of the program line by line:

```python
import tkinter as tk

This imports tkinter module and renames it tk. Tkinter is a standard Python library for creating GUI applications.

```python
import filedialog

This creates the output area for displaying the processed.

Example_filtered = example[example['MCV']<78]
filtered['cut_off'] = 3.667 - 0.01*Hb + 0.001*HCT - 0.004*MCV + 0.064*MCH
output = return_filtered

This defines a function filter_data() that takes a file path as input, reads the CSV file using pandas, filters the data based on some criteria, and calculates a new column based on a formula. The filtered data are then returned.

```python
def process_data(filepath):
    data = pd.read_csv(filepath)
    example_filtered = example[example['MCV']<78]
    example_filtered['cut_off'] = 3.667 - 0.01*example['Hb'] + 0.001*example['HCT'] - 0.004 * example['MCV'] + 0.064*example['MCH']
    example_filtered['cut_off'] > 4.96
    return example_filtered

This defines a function process_data() that is called when the user clicks the "Process Data" button. It gets the file path from the entry field, calls filter_data() to filter and process the data, deletes any existing text in the output area, and inserts the processed data as a string.
output += "\n\n" + disclaimer
return output

The provided function accepts four input parameters (Hb, HCT, MCV, and MCH) and performs calculations based on these values. The calculation process includes the use of conditional statements and equations. At the end of the calculation, a disclaimer and an asterisk are appended to the output. The resulting output is then returned by the function.

```python
Def on_click():
    Hb = float(hb_entry.get())
    HCT = float(hct_entry.get())
    MCV = float(mcv_entry.get())
    MCH = float(mch_entry.get())
    result = calculate_result(Hb, HCT, MCV, MCH)
    Result_text.config(state=tk.NORMAL)
    Result_text.delete("1.0", tk.END)
    Result_text.insert(tk.END, result)
    Result_text.config(state=tk.DISABLED)
```

This function is called when the user clicks the "Calculate" button. It reads the values of the four input fields (hb_entry, hct_entry, mcv_entry, and mch_entry), converts them to floats, passes them as parameters to the calculate_result() function, and then, displays the resulting text in the output field (result_text). The output field is first set to normal state, then cleared, then populated with the calculated result, and finally set back to disabled state.

```python
window = tk.Tk()
window.title("Thal_Screen_CBC")
# Set the window size
Window.geometry("640x360")
```

This creates the main GUI window and sets its title to "Thal_Screen_CBC". The window size was also set to 640x360 pixels.

```python
# Create input field for Hb, HCT, MCV, and MCH
Hb_label = tk.Label(window, text="Hb (g/dL)\n")
Hb_label.grid(row=0, column=0, padx=5, pady=5)
Hb_entry = tk.Entry(window)
Hb_entry.grid(row=0, column=1, padx=5, pady=5)
```

This section creates a label and an entry box for the hemoglobin (Hb) value. Hb_label is a label widget, and Hb_entry is an entry widget. The grid() method is used to place the widgets in a grid format on the window. row=0, column=0 specifies the position of the label widget, and row=0, column=1 specifies the position of the entry widget. pads and pay add padding around the widgets to improve their appearance.

```python
Hct_label = tk.Label(window, text="HCT (%):")
Hct_label.grid(row=1, column=0, padx=5, pady=5)
Hct_entry = tk.Entry(window)
Hct_entry.grid(row=1, column=1, padx=5, pady=5)
```

This section creates a label and an entry box for the hematocrit (HCT) value. Hct_label is a label widget and Hct_entry is an entry widget. The grid() method is used to place the widgets in a grid format on the window.

```python
Mcv_label = tk.Label(window, text="MCV (fL)\n")
Mcv_label.grid(row=2, column=0, padx=5, pady=5)
Mcv_entry = tk.Entry(window)
Mcv_entry.grid(row=2, column=1, padx=5, pady=5)
```

This section creates a label and an entry box for the mean corpuscular volume (MCV) value. Mcv_label is a label widget and Mcv_entry is an entry widget. The grid() method is used to place the widgets in a grid format on the window.

```python
Mch_label = tk.Label(window, text="MCH (pg)\n")
Mch_label.grid(row=3, column=0, padx=5, pady=5)
```

This section creates a label and an entry box for the mean corpuscular hemoglobin (MCH) value. Mch_label is a label widget, and Mch_entry is an entry widget. The grid() method is used to place the widgets in a grid format on the window.

```python
Calculate_button = tk.Button(window, text="Calculate", command=on_click)
Calculate_button.grid(row=4, column=0, columnspan=2, padx=5, pady=5)
```

This line creates a button widget with the label "Calculate" and assigns it to the variable calculate_button. The command argument specifies the function that will be called when the button is clicked, which is onclick.

```python
Result_label = tk.Label(window, text="*Result:\n")
Result_label.grid(row=5, column=0, pady=5)
Result_text = tk.Text(window, width=20, height=2, state=tk.DISABLED)
Result_text.grid(row=5, column=1, padx=5, pady=5)
```

This line creates a label widget with the text "*Result:" and assigns it to the variable Result_label. The text widget is initialized with a blank string, and padding is added. This line creates a text widget with a width of 20 characters and height of 2 lines. The state argument sets the text widget to be initially disabled so that the user cannot type into it. This line places the resulting text widget on the GUI window using the grid geometry manager. The widget is initialized with a frame object and grid manager. The label is placed in row 5 and column 1 and has some padding.

```python
result_label = tk.Label(window, text=disclaimer, wraplength=600, justify=tk.LEFT)
result_label.grid(row=6, column=0, columnspan=2, padx=5, pady=5)
```

This line creates a string variable called disclaimer that contains a disclaimer message for the program. This line creates a label widget with the text from the disclaimer variable and assigns it to the variable result_label. The wraplength argument specifies the maximum width of the label before the text wraps to the next line and the justify argument sets the text alignment to the left. This line places the disclaimer_label widget on the GUI window using the grid geometry manager. The label is placed in row 6 and spans 2 columns. Padding is added to the widget.

```python
window.mainloop()
```

This line starts the event loop for the GUI window, which waits for user input and responds accordingly. It runs until the user closes the window or the program exits.

**DISCUSSION**

Moreover, the development of the Python program is a cost-effective solution for improving screening programs, especially in resource-limited settings. HPLC is commonly used to screen for thalassemia, but it is expensive and not available everywhere. In contrast, the CBC method is readily available and affordable, making it a more accessible option for screening programs. Using CBC data and the Python program to analyze it, the screening process can be made even more affordable and accessible. This approach can benefit many communities that do not have access to HPLC-based screening programs.
Another advantage of the Python program is that it can be easily adapted to incorporate additional parameters or algorithms; for example, the program can be modified to include other blood parameters, such as reticulocyte count or hemoglobin electrophoresis, to further improve the accuracy of blood disorder diagnosis. The program can also be updated with more advanced algorithms such as machine learning to enhance the accuracy and efficiency of the screening process further.

Ethical considerations were carefully considered during the study. The local ethics committee of Panskura Banamali College approved the study, and informed consent was obtained from all participants before collecting the blood samples. The anonymity of the participants was also ensured by anonymizing the dataset to protect their privacy.

**CONCLUSION**

The new screening process based on the CBC method and the Python program built for individual and batch processing is a valuable tool for blood disorder diagnosis. The program’s accuracy, flexibility, and affordability make it a promising solution for improving screening programs, particularly in resource-limited settings. The program’s ability to incorporate additional parameters or algorithms also means that it has the potential for further improvement and refinement. With further validation studies and modifications, the program could become an essential tool in improving health-care outcomes for many communities.

**ACKNOWLEDGMENT**

The authors are thankful to Calcutta School of Tropical Medicine, 108, Chittaranjan Ave, Calcutta Medical College, College Square, Kolkata, West Bengal 700073 for providing the necessary facilities for HPLC test for both validation test sample sets.

**CONFLICTS OF INTERESTS**

The authors declare no conflict of interest regarding the publication of this manuscript.

**AUTHORS CONTRIBUTION**

The study conception and design involved contributions from both authors. Abhisek Samanta played a role in study design, data collection, and manuscript preparation. Nandan Bhattacharyya contributed to the study design, supervised the project, and prepared the final manuscript. Both authors thoroughly reviewed and approved the final manuscript.

**AUTHORS FUNDING**

This research received no specific funding from any external agency, commercial entity, or institution. The study was conducted as part of the authors’ academic and research activities.

**REFERENCES**


