PROJECT: CLIMATOLOGY

In collaboration with the LSCE

Conducted by:
Pasha Alidadi
Linn Habberstad
Teddy Tonin

INTRODUCTION

→ Analyseries: a time series analysis and comparison software employed by the LSCE and others for paleoclimatological studies analyzing ice cores to interpret climate signals

→Limitations: Developed in the 1990s, incompatible with recent macOS or Windows-based computers

→ Response: Students at CentraleSupélec updating the software to maintain the same functionalities but enabling it to be open source and programmed in Python

OUTLINE

→ Background

→ Methodology

→ Implementation and results

→ Discussion

→ Conclusion

BACKGROUND

→ Ice cores: Provide insights into historical climate changes through measurement of physical and chemical properties such as various chemical compounds

→Data: Each measurement in the data links to a time scale or depth scale, which allows the analysis of the chronology of sedimentation. (The Metadata file provides a better understanding.)

→ Interpretation: Researchers interpret the data extracted through comparison with reference data to conclude the sedimentation rates and the definition of the geological specificities of the specific location

Depth and Depth (m) EDC	Depth at which ice core sample was obtained
δ18Oforams-b, δ18Oforams-b.1	Values for the oxygen isotope found in a sample
δ13Cforams-b, δ13Cforams-b.1	Values for the carbon isotope found in a sample
Time (ka)	The time in thousands of years (ka) before present
Benthic δ18O (per mil)	Values of oxygen isotope ratio at the bottom of the ocean
Standard error (per mil)	An indication of the precision of the measurements
Ice age (a 1950) AICC/AICC.1	Age of the ice at a particular depth
Gas age (a 1950) AICC	Age of the gases trapped within the ice core
EDC CH4 [ppbv]	Concentration of methane
CO2 composite Bereiter	Concentration of carbon dioxide
nssCa800ka	Non-sea-salt calcium concentrations
EDC deuterium excess (deut cor)	Deuterium excess data
Depth/ice age m EDC	Ratio between the depth and the age of the ice

BACKGROUND

	depth ODP849	d18Oforams-b	d13Cforams-b	depth ODP846	d18Oforams-b.1	d13Cforams-b.1	Time (ka)	Benthic d18O (per mil)	Standard error (per mil)
0	7.0	3.66	0.21	12.0	3.380	0.14	0	2.588	0.031
1	17.0	3.49	0.08	23.0	3.460	0.01	1	2.588	0.037
2	28.0	3.31	0.19	33.0	3.765	-0.08	2	2.539	0.031
3	45.0	4.17	-0.15	43.0	4.140	-0.17	3	2.646	0.027
4	55.0	4.69	-0.27	53.0	4.470	-0.21	4	2.655	0.033
...	...	...	...	...	...	...	...	...	...
2111	NaN	NaN	NaN	NaN	NaN	NaN	5305	2.154	0.037
2112	NaN	NaN	NaN	NaN	NaN	NaN	5310	2.150	0.093
2113	NaN	NaN	NaN	NaN	NaN	NaN	5315	2.202	0.073
2114	NaN	NaN	NaN	NaN	NaN	NaN	5320	2.265	0.089
2115	NaN	NaN	NaN	NaN	NaN	NaN	5325	2.051	0.051

→Data: Each measurement in the data links to a time scale or depth scale, which allows the analysis of the chronology of sedimentation

BACKGROUND

→ Ice cores: Provide insights into historical climate changes through measurement of physical and chemical properties such as various chemical compounds

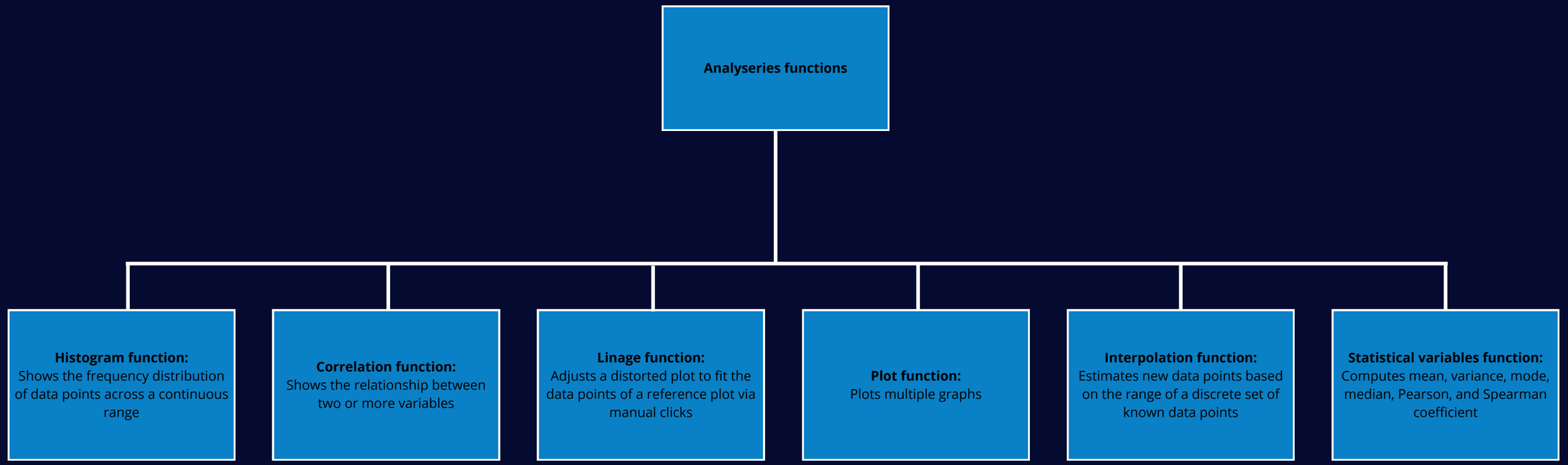
→Data: Each measurement in the data links to a time scale or depth scale, which allows the analysis of the chronology of sedimentation. (The Metadata file provides a better understanding.)

→ Interpretation: Researchers interpret the data extracted through comparison with reference data to conclude the sedimentation rates and the definition of the geological specificities of the specific location

THE STARTING POINT

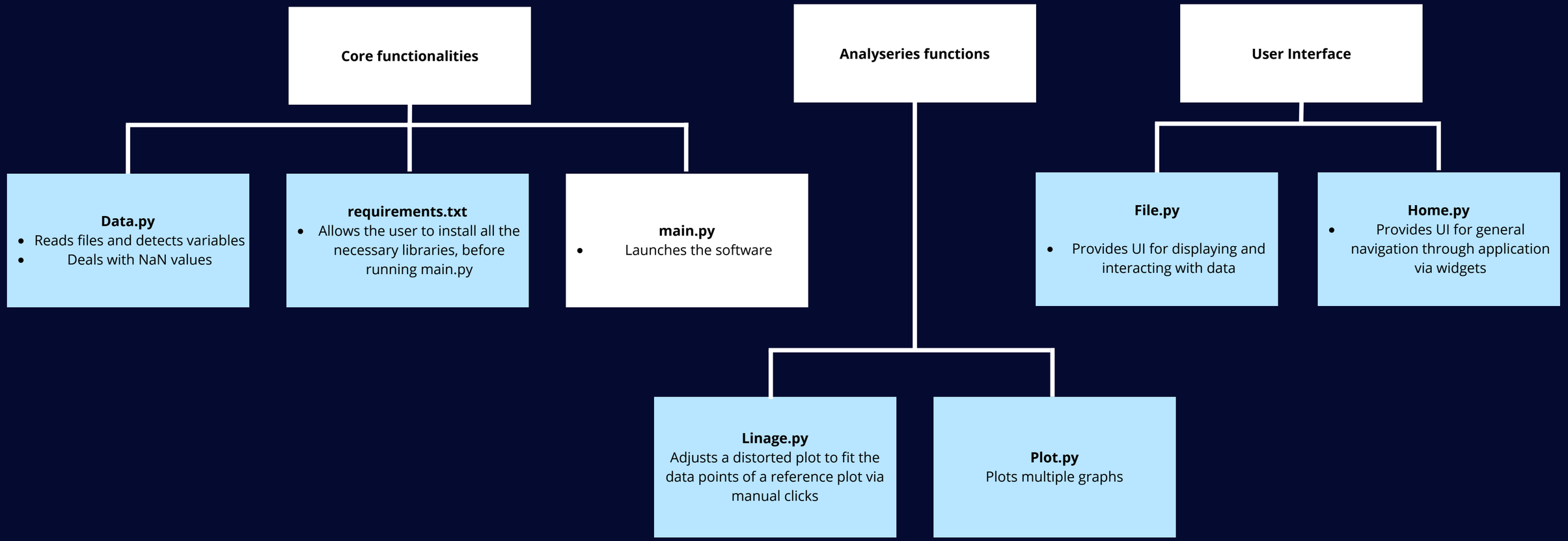
→ Original version: Carried essential functionalities (see below)

→ It was coded in C++ and therefore incompatible with modern computers.



THE STARTING POINT

→ Spring version



METHODOLOGY

- Evaluation: Conducted thorough review and user feedback on the spring version to identify practical challenges.

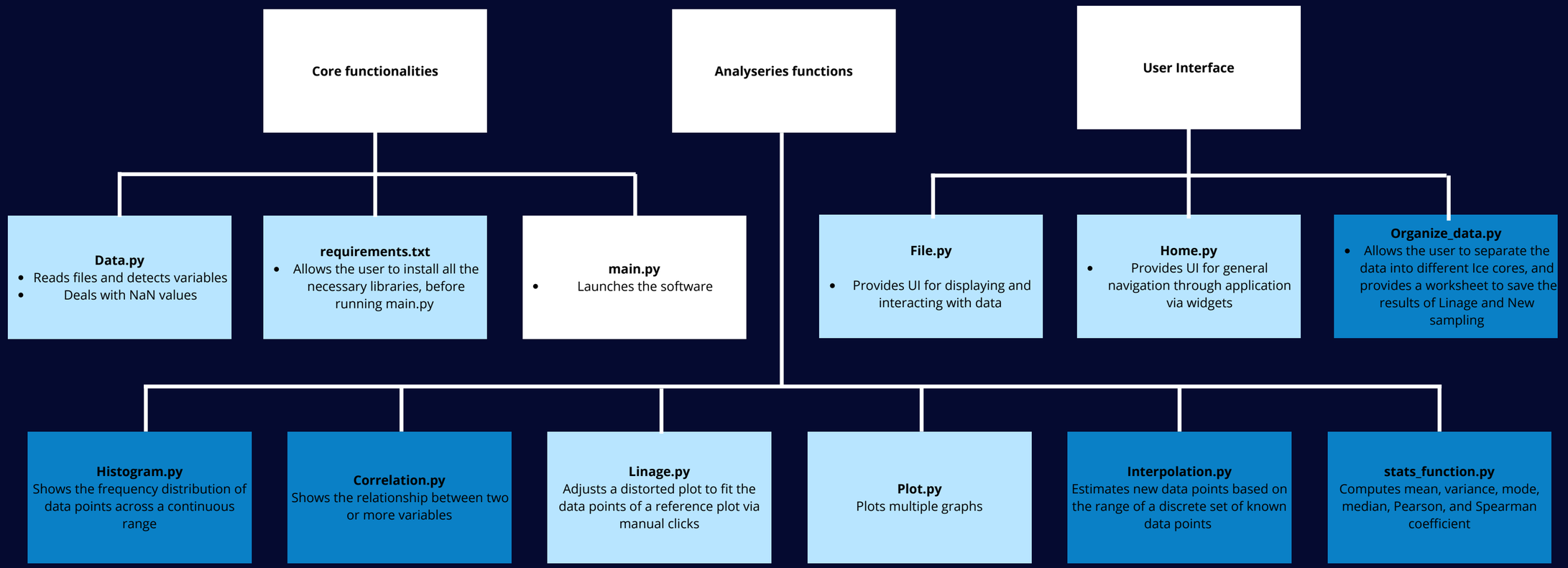
- Identification: Issues in data representation, file processing, code efficiency, and functional limitations in both original and Python versions could be identified.

- Procedure: All actions generally involved regular code review, refactoring, refining sessions, and post-implementation feedback for validation.

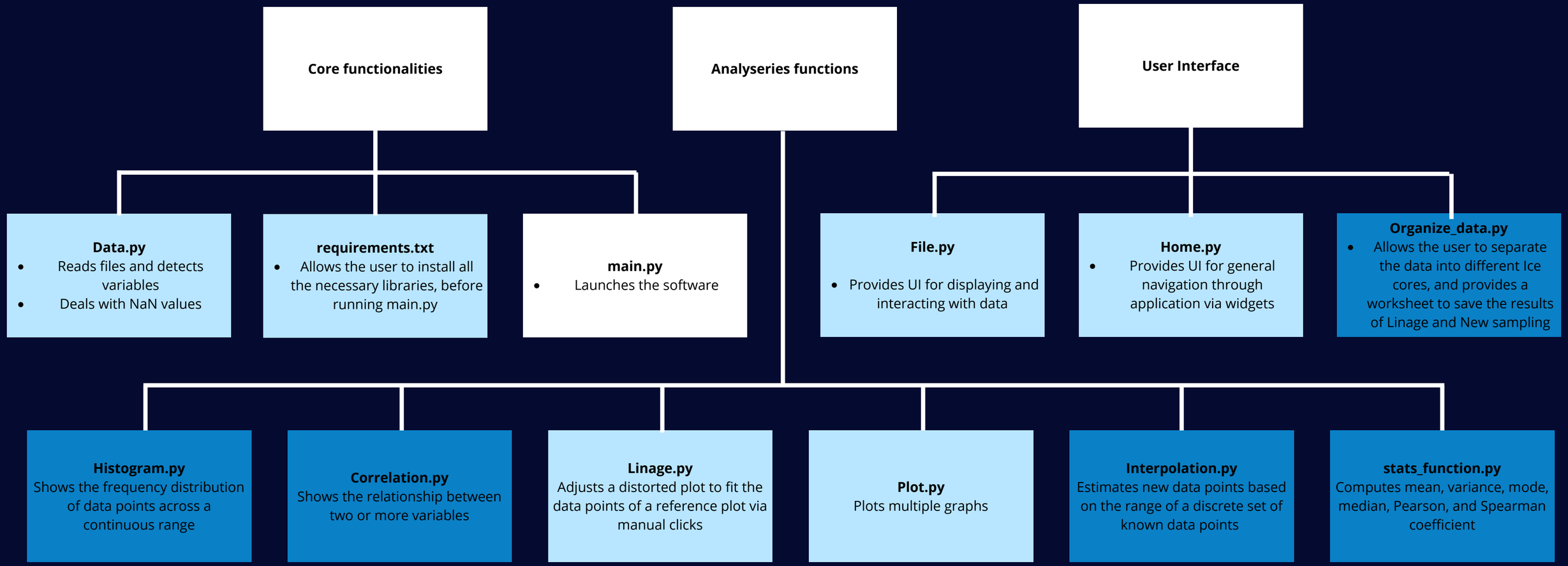
    - Address flaws by refining problematic code.
    - Add missing functionalities with a focus on underlying mathematical logic.
    - Enhance code conciseness, efficiency, and scalability.
    - Introduce a Multiseries function for simultaneous recalibration of multiple datasets.

METHODOLOGY

1. Core functionalities: Include file management and software launching
2. User Interface: Deals with how the user interacts with the software
3. Analyseries functionalities: The functionalities from the initial version.



IMPLEMENTATION AND RESULTS



USER INTERFACE

Problems:

→ Current menu is inconsistent with the Mac version

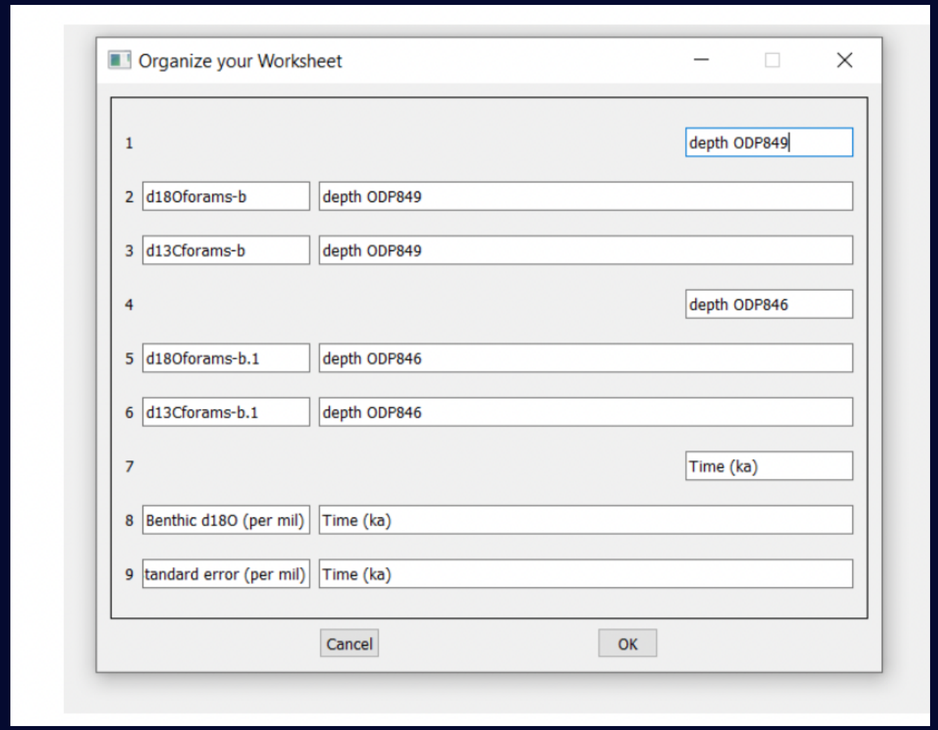
→ Unable to save the results of Linage, Interpolation or the Stats functions

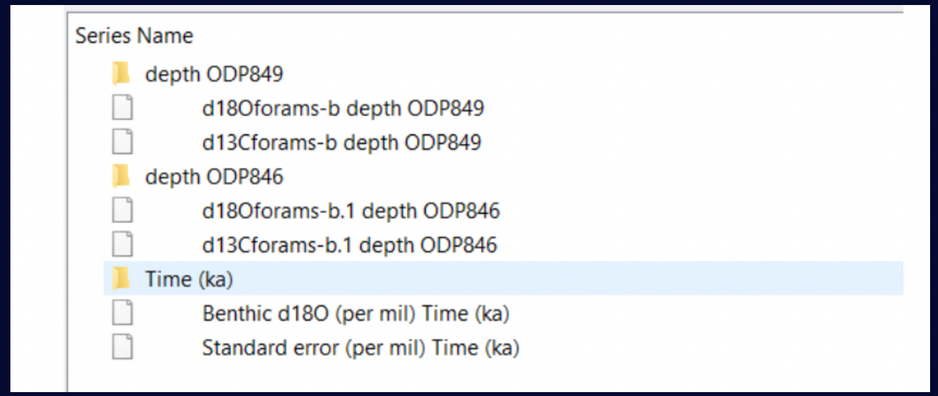
→ User has to select data on each menu

Implementation:

→ **Worksheet feature**, a window that offers data management and saving capabilities, as seen in old application

```python
def is_monotonic_increasing(array):
    return np.all(np.diff(array) >= 0)
```





USER INTERFACE

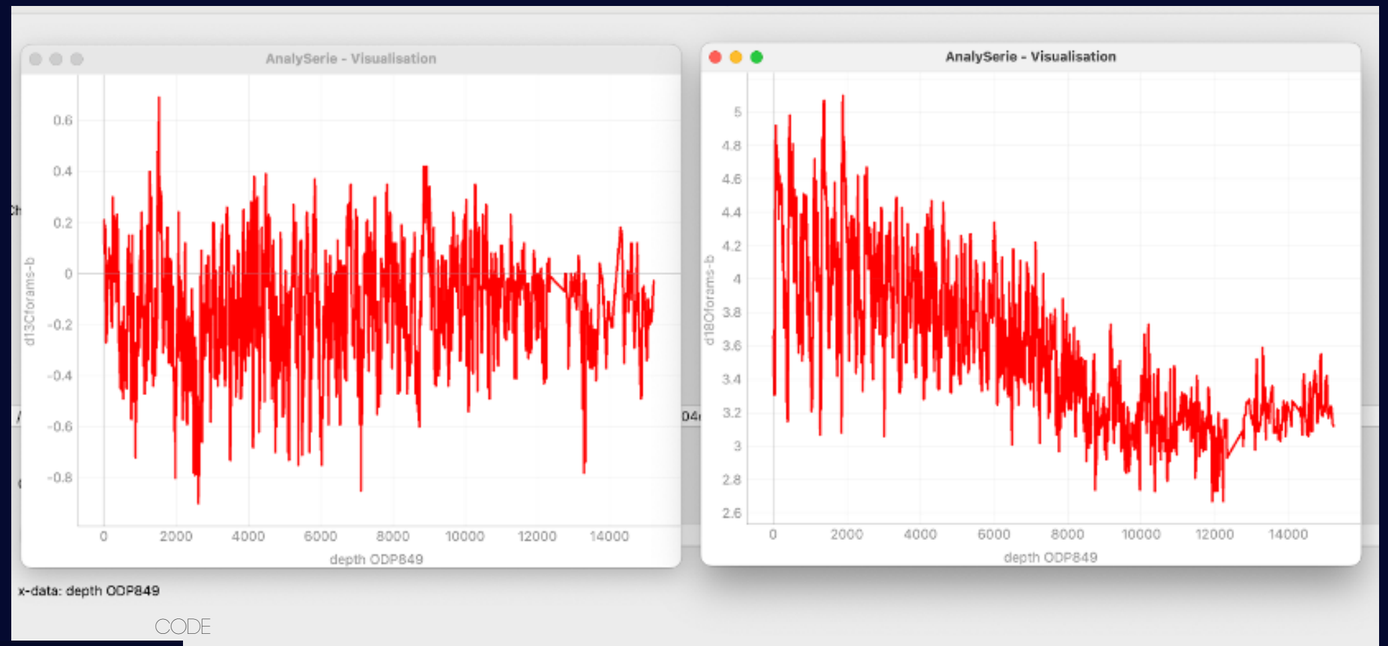
Problems:

→ Plot.py would not allow for multiple windows to be shown in old application

Implementation:

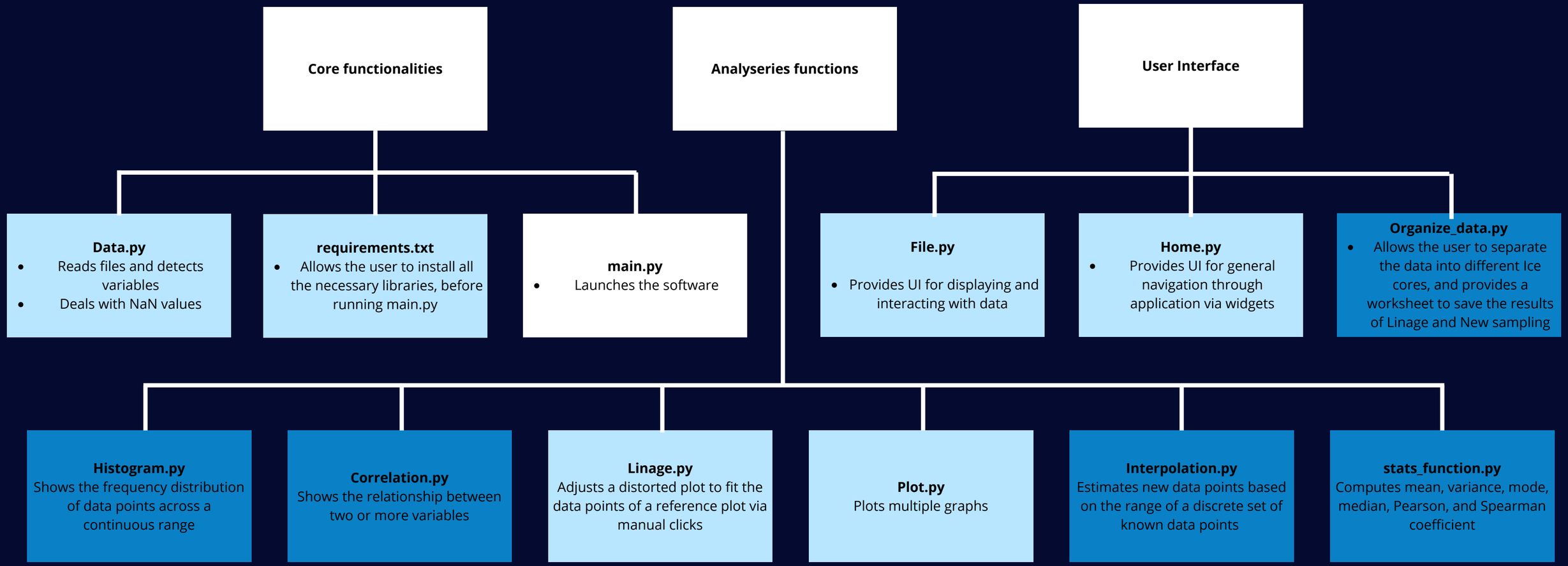
```python
def update_graph(self, x, y, abs, ord):
    # Clear the current graph
    self.graphWidget.clear()
    # Update the graph with new data
    pen = pg.mkPen(color=(255, 0, 0), width=5)
    self.graphWidget.setLabel('left', ord)
    self.graphWidget.setLabel('bottom', abs)
    self.graphWidget.plot(x, y, pen=pen)

def closeEvent(self, event):
    self.isClosed = True  # Update the flag when the window is closed
    super(Fenetre_graphe, self).closeEvent(event)
```

After



CODE

IMPLEMENTATION AND RESULTS



DATA REPRESENTATION AND FILE PROCESSING

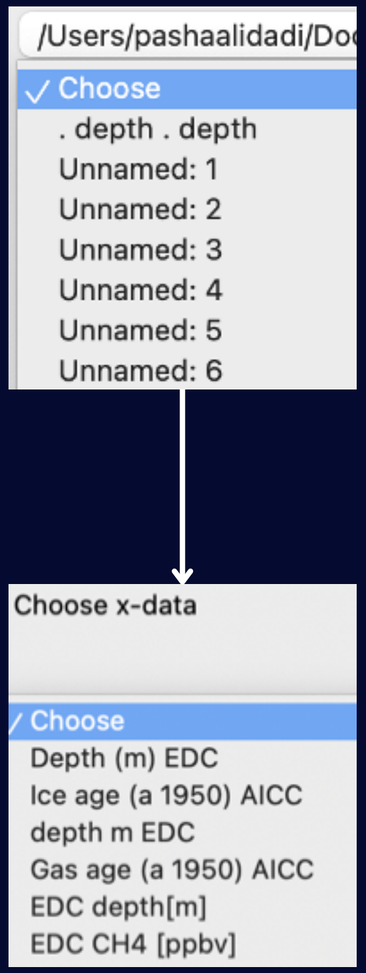
Problems:

→ failing to detect variable names correctly from .txt files

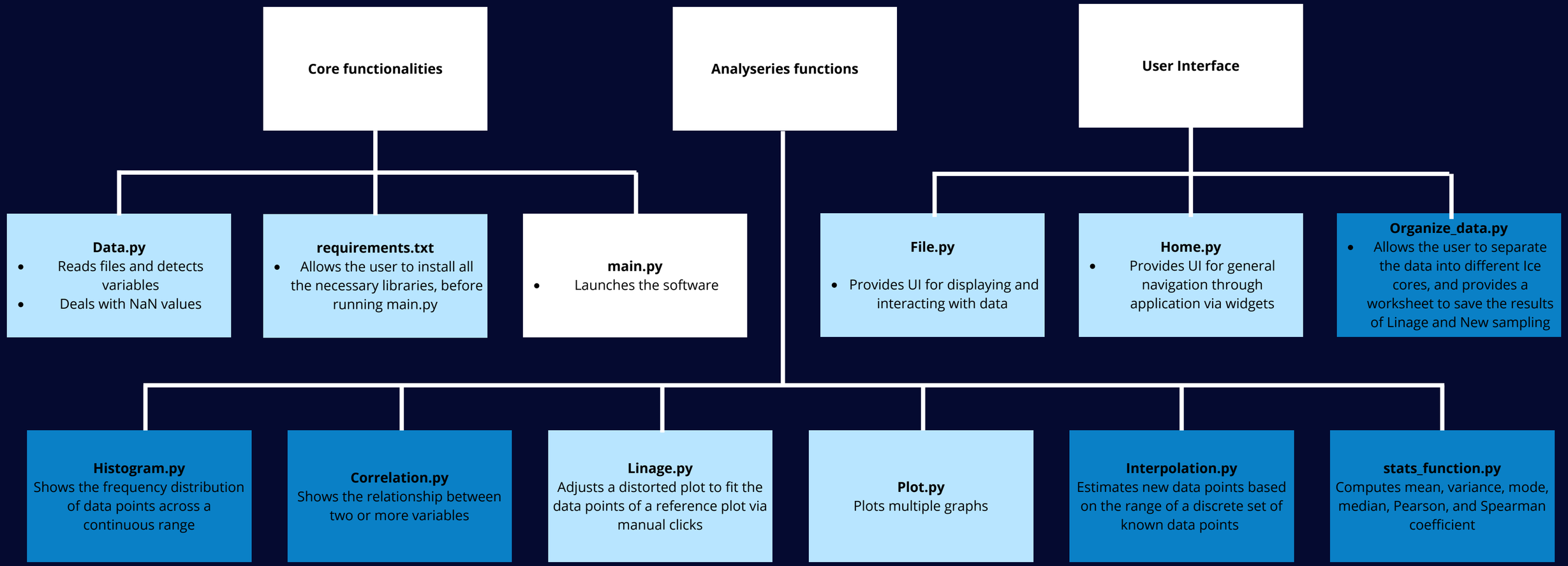
→ fails to recognize .xlxs files

Implementation:

```python
def read_data(filepath):
    if filepath.endswith('.xlsx'):
        # Excel file
        dtype_dict = {col_name: str for col_name in pd.read_excel(filepath, nrows=detect_header(filepath))}
        df = pd.read_excel(filepath, header=detect_header(filepath), dtype=dtype_dict)

        # Convert the DataFrame to a .txt file, because the program does not work with xlsx files
        txt_filepath = filepath.rsplit('.', 1)[0] + '.txt'  # Change the file extension to .txt
        df.to_csv(txt_filepath, sep='\t', index=False, header=True)  # Save as a tab-delimited .txt file

        print(f"Converted Excel file to {txt_filepath}")
        df = df.dropna(axis=1, how='all')
        return df  # Return the DataFrame for further processing
```



IMPLEMENTATION AND RESULTS



CODE OPTIMIZATION

Problems:

→ untidy and inefficient code with redundant loops

Implementation (one of many):

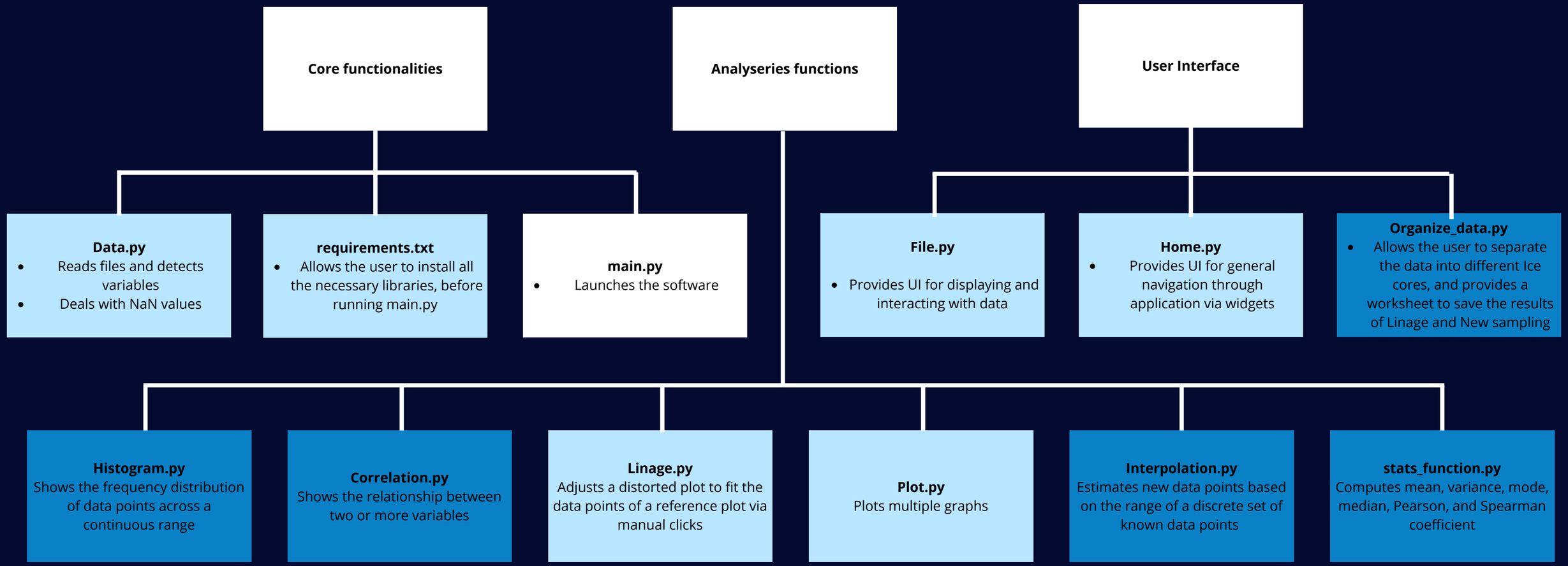
Before

```
def push_choice_y(self, index):
    self.y = index
    y1 = math.inf
    i = 0
    while (y1 == math.inf) and i < len(list(self.df.columns)):
        if list(self.df.columns)[i] == index:
            y1 = list(self.df.columns)[i]
        i += 1
    self.y_values = list(self.df[y1])
    self.labelY.setText("y-data: {}".format(index))
    if self.x is not None:
        self.graph()
```

After

```
def push_choice_y(self, index):
    if index in self.df.columns:
        self.y = index
        self.y_values = list(self.df[index])
        self.labelY.setText(f"y-data: {index}")
        if self.x is not None:
            self.graph()
```

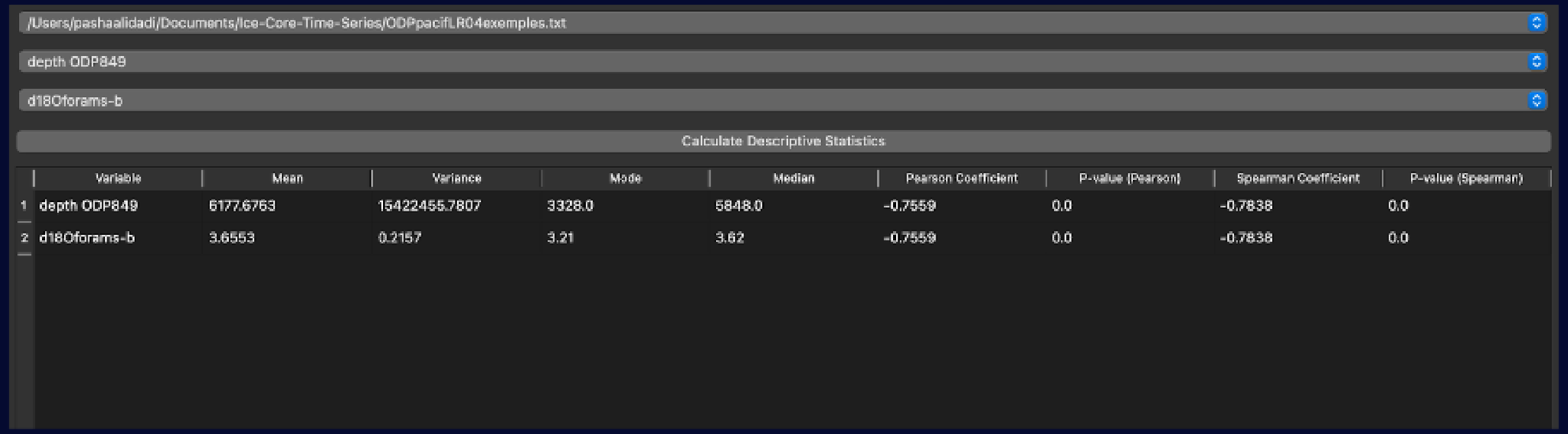
IMPLEMENTATION AND RESULTS



STATS FUNCTIONS

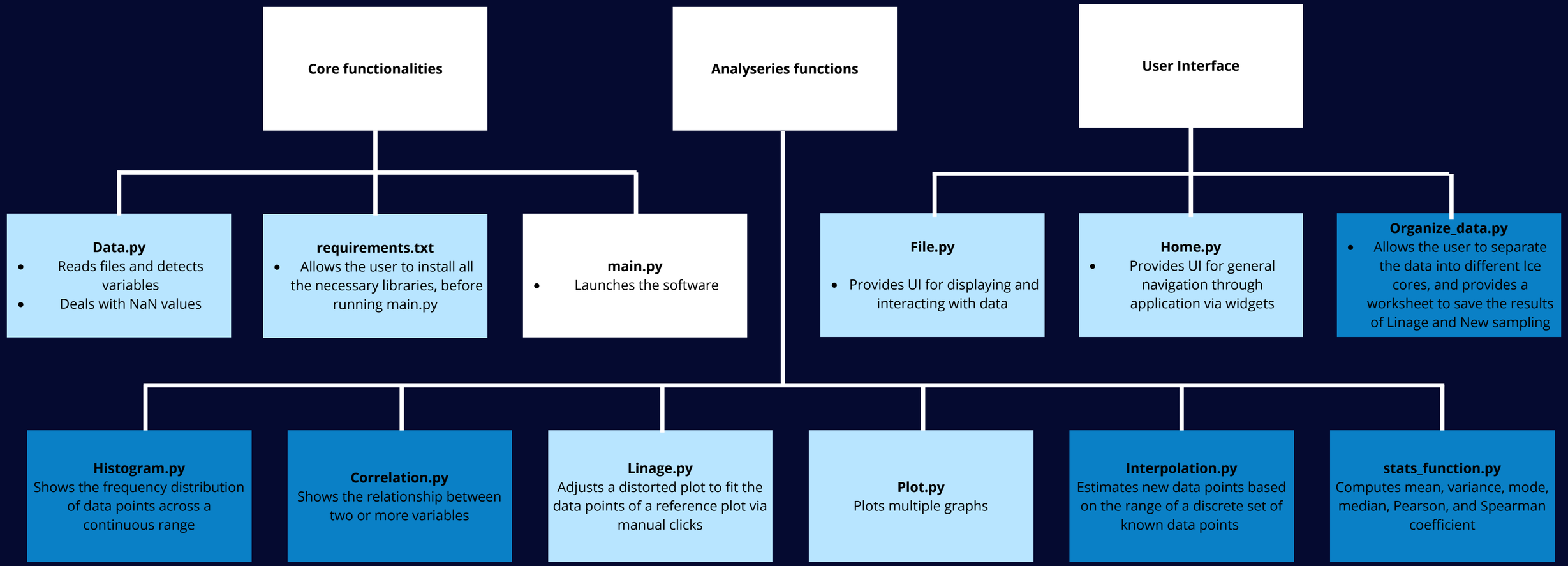
Calculates the mean, median, variance, Pearson's correlation coefficient, Spearman's correlation coefficient, and the probability of zero rank of each variable inside of a dataset
→ descriptive statistics

Output:



	Variable	Mean	Variance	Mode	Median	Pearson Coefficient	P-value (Pearson)	Spearman Coefficient	P-value (Spearman)
1	depth ODP849	6177.6763	15422455.7807	3328.0	5848.0	-0.7559	0.0	-0.7838	0.0
2	d18Oforams-b	3.6553	0.2157	3.21	3.62	-0.7559	0.0	-0.7838	0.0

IMPLEMENTATION AND RESULTS



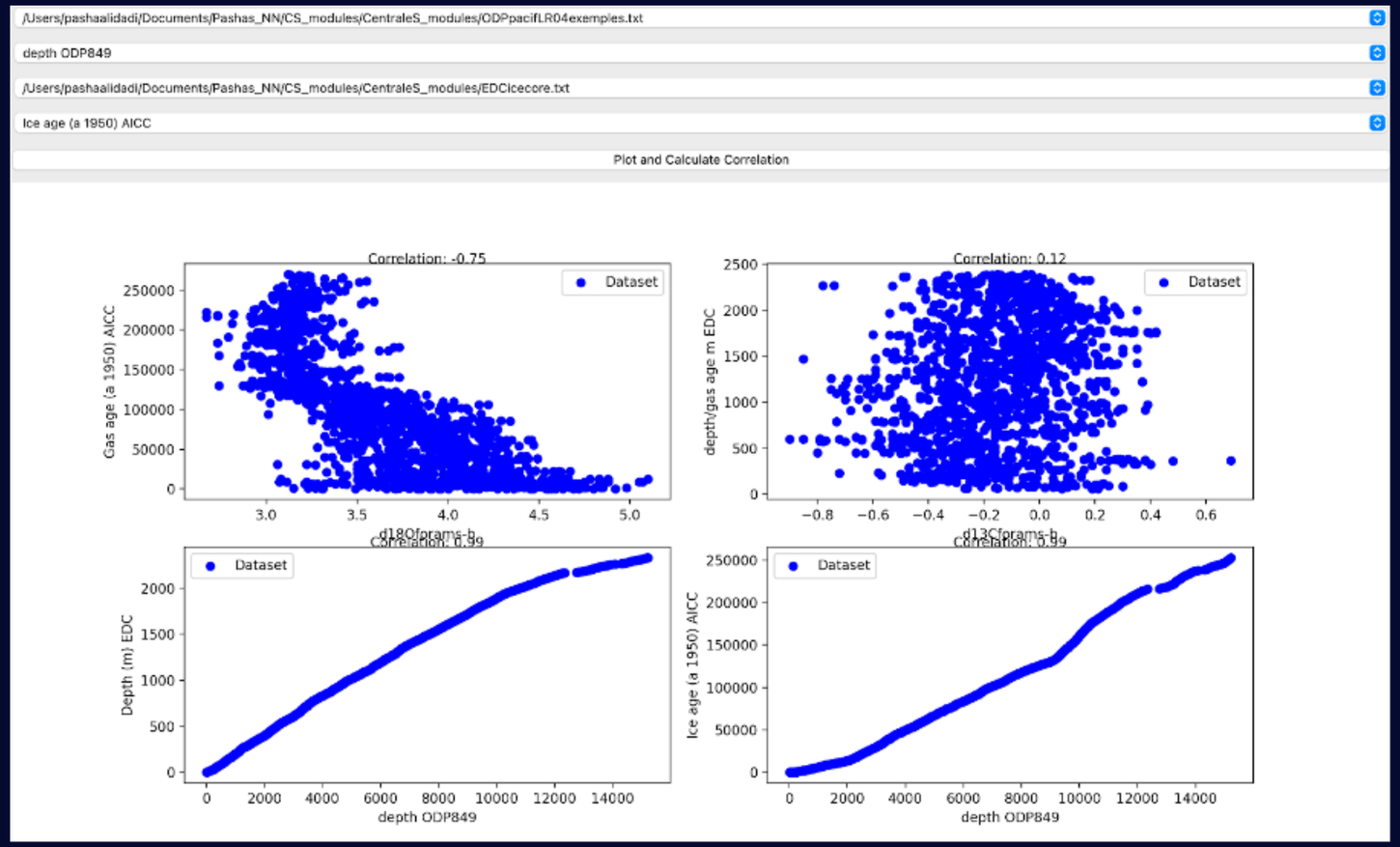
CORRELATION

Is a measure of an association or relationship between variables

→ between [-1, 1] while 1 means there is a positive correlation between the two variables and -1 vice versa

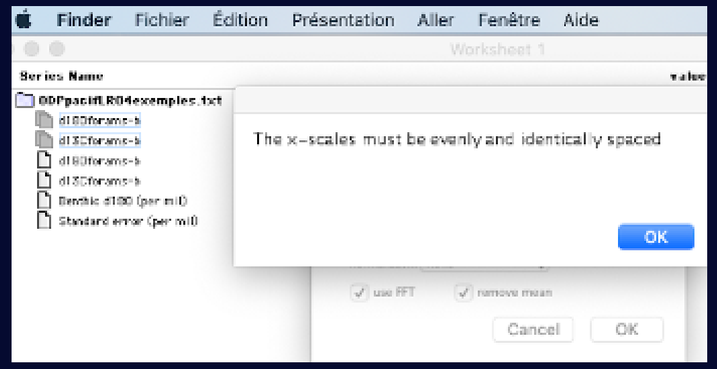
→ can display up to four different plots at the same time of different file sources

Output:



CORRELATION

Problem: Old application had difficulties with different lengths of variables

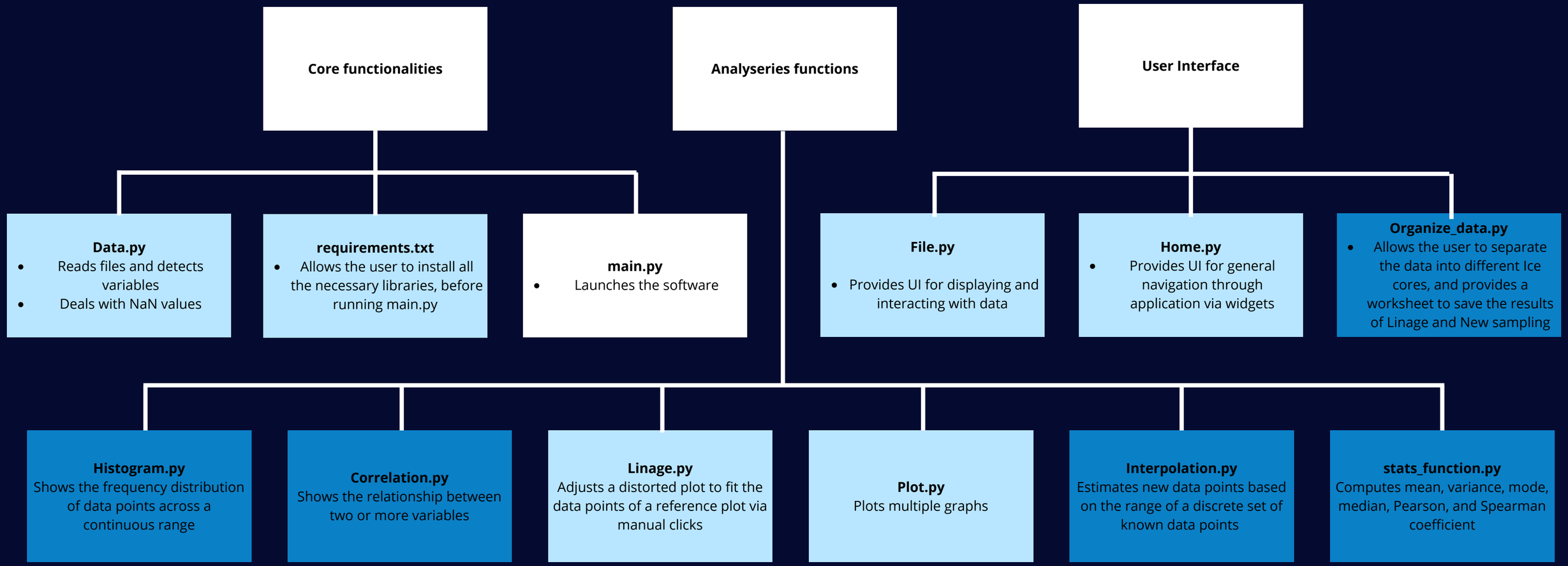


Implementation:

```python
def get_data_ready_for_linage(data_array1, col1, data_array2, col2):
    '''This function handles data from different sources and aligns them by
length after removing NaN values'''

    x_ref = data_array1[col1].dropna().to_numpy()
    y_ref = data_array2[col2].dropna().to_numpy()

    # Truncate the longer array to match the length of the shorter one
    min_length = min(len(x_ref), len(y_ref))
    x_ref = x_ref[:min_length]
    y_ref = y_ref[:min_length]

    return x_ref, y_ref
```

Same principle applied to the Histogram function as well
→ adds flexibility for comparing different archives with different lengths

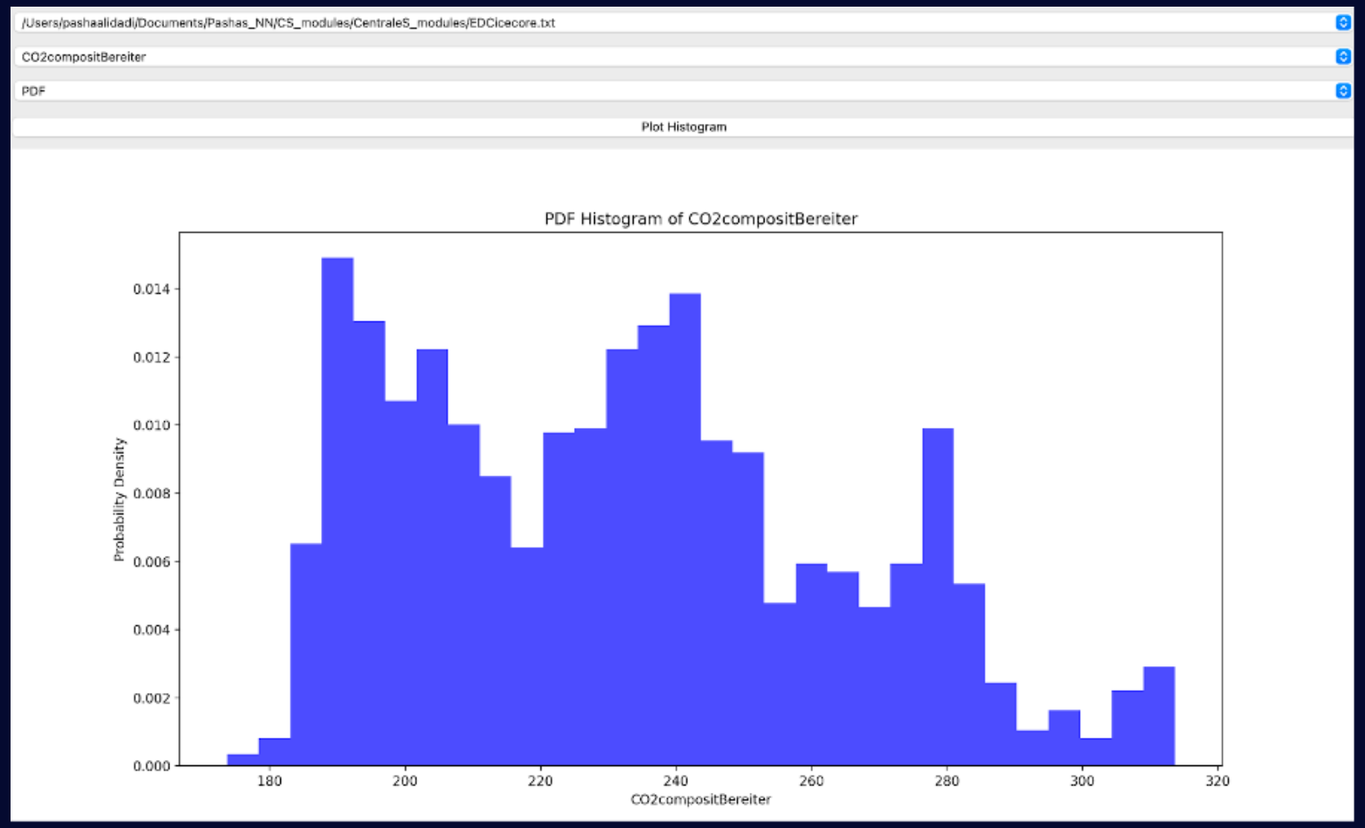
IMPLEMENTATION AND RESULTS



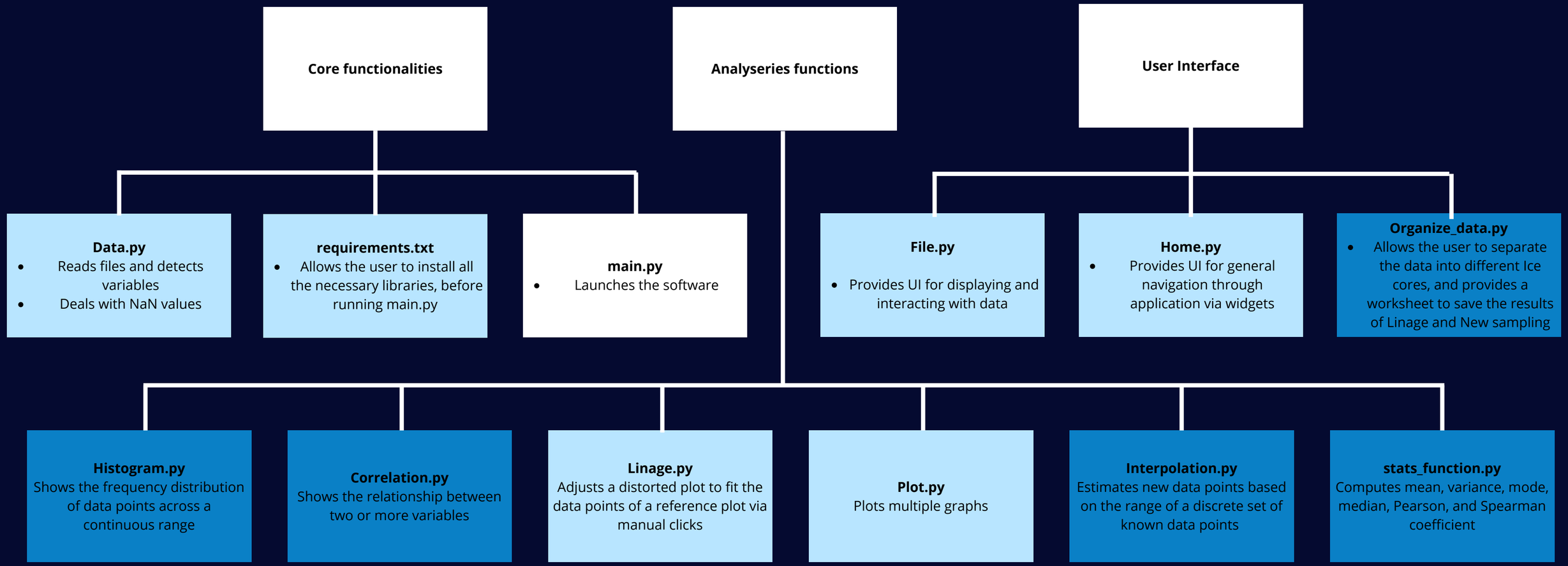
HISTOGRAM

Depicts rectangles whose area is proportional to the frequency of a variable and whose width is equal to the class interval
→ user can select between cumulative probability, probability density, and specific values

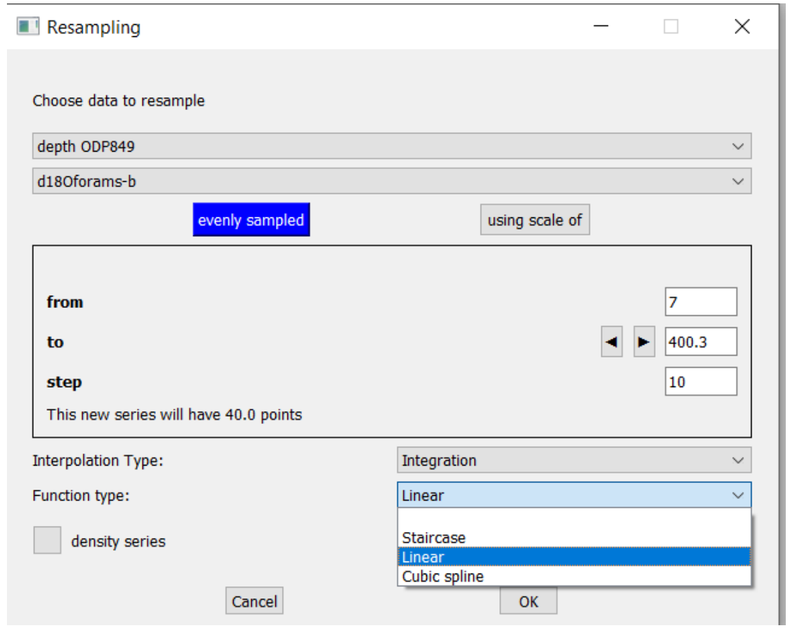
Output:

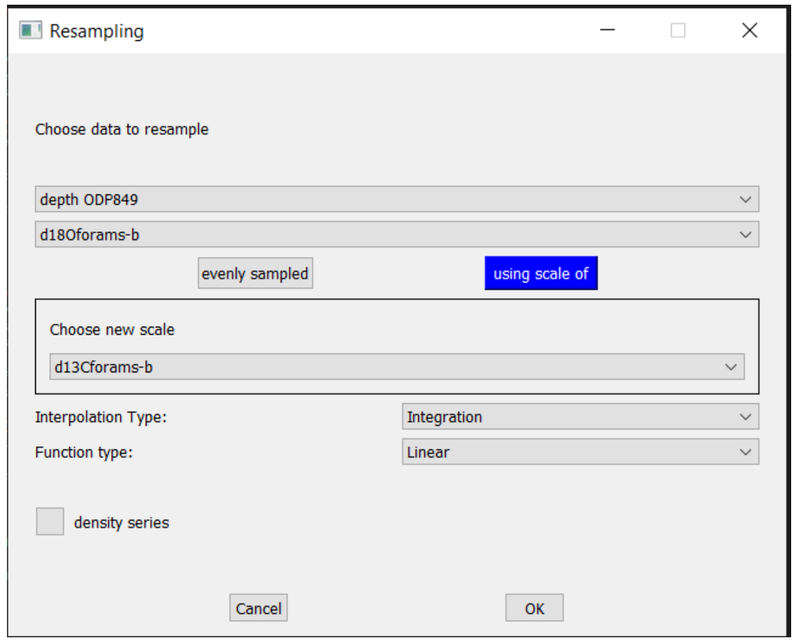


IMPLEMENTATION AND RESULTS

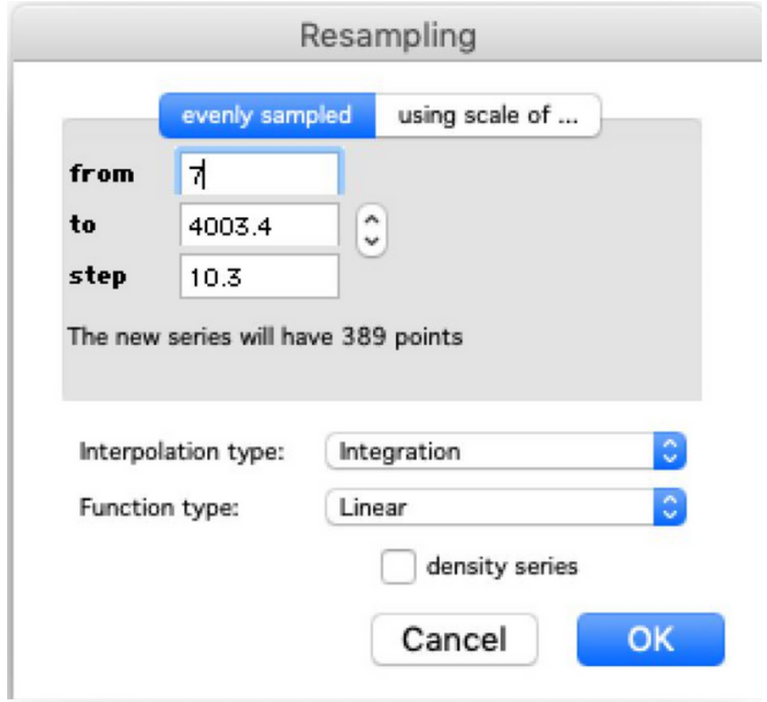


INTERPOLATION

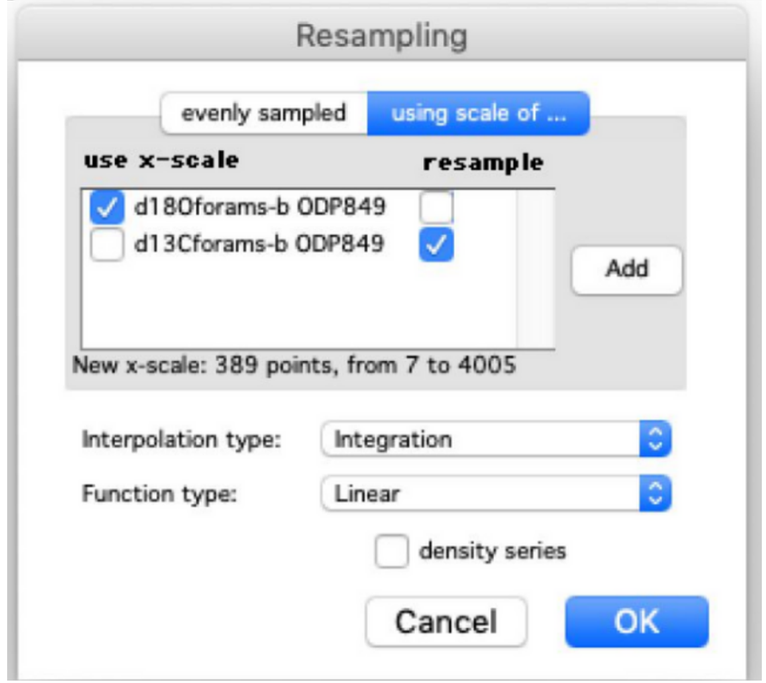




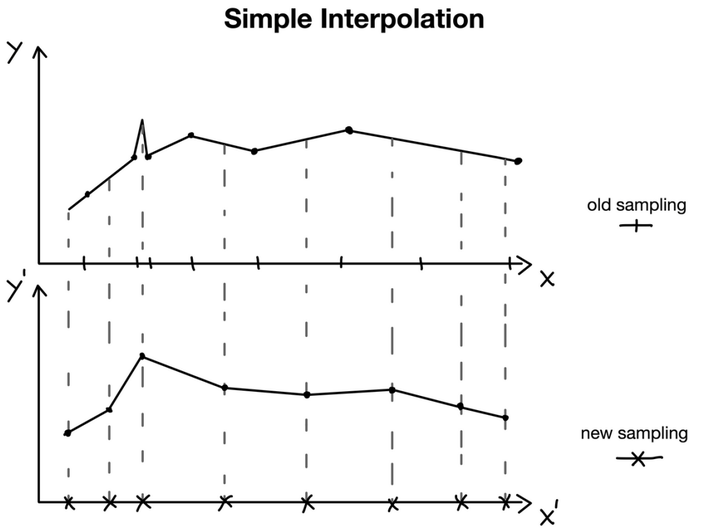
Interpolation on Mac→

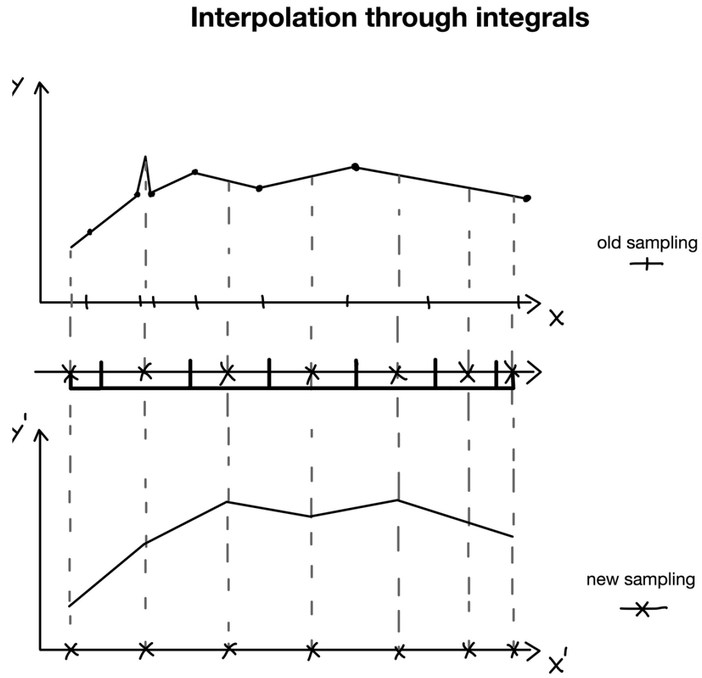


←Interpolation on the updated version of Analyseries



INTERPOLATION BY INTEGRATION

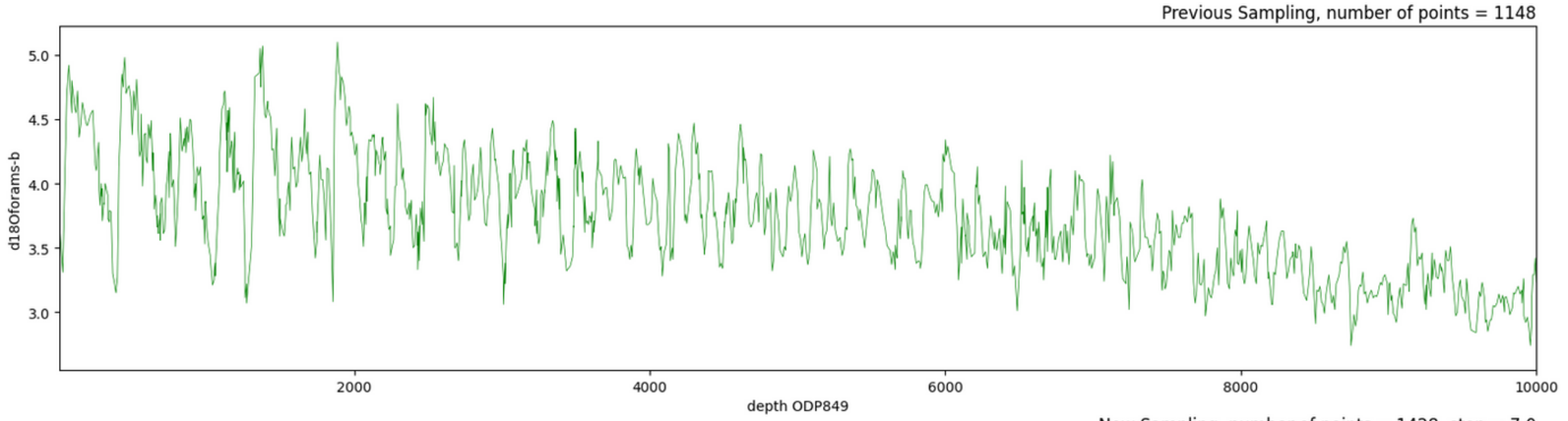


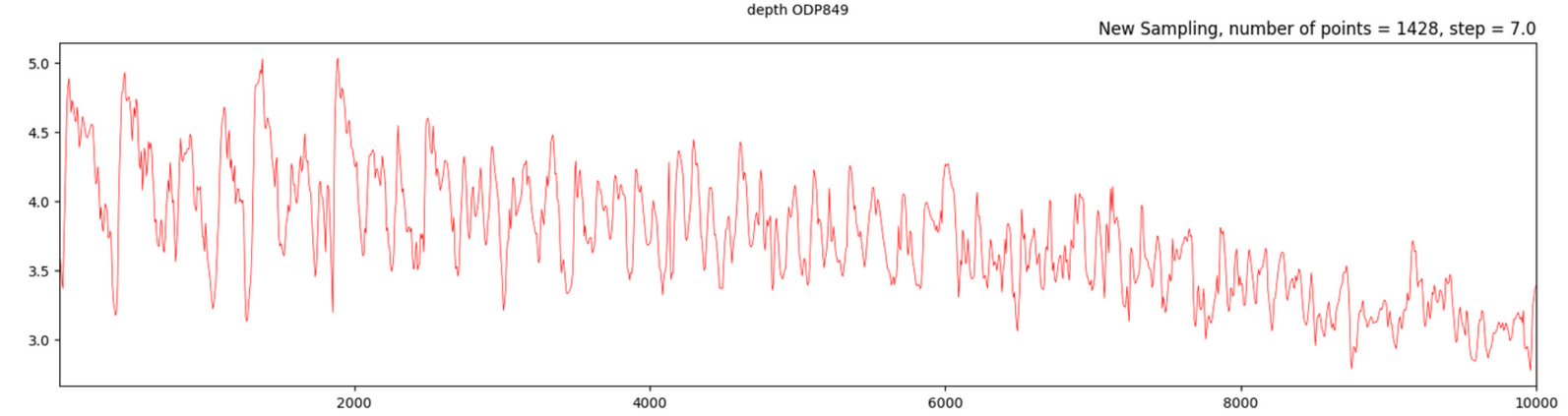


Linear interpolation:

→ estimate a value y between two known data points (x0,y0) and (x1,y1)

However, assumes a straight line connecting the two known data points, which might not be feasible for noisy data as it is the case with environmental data sometimes



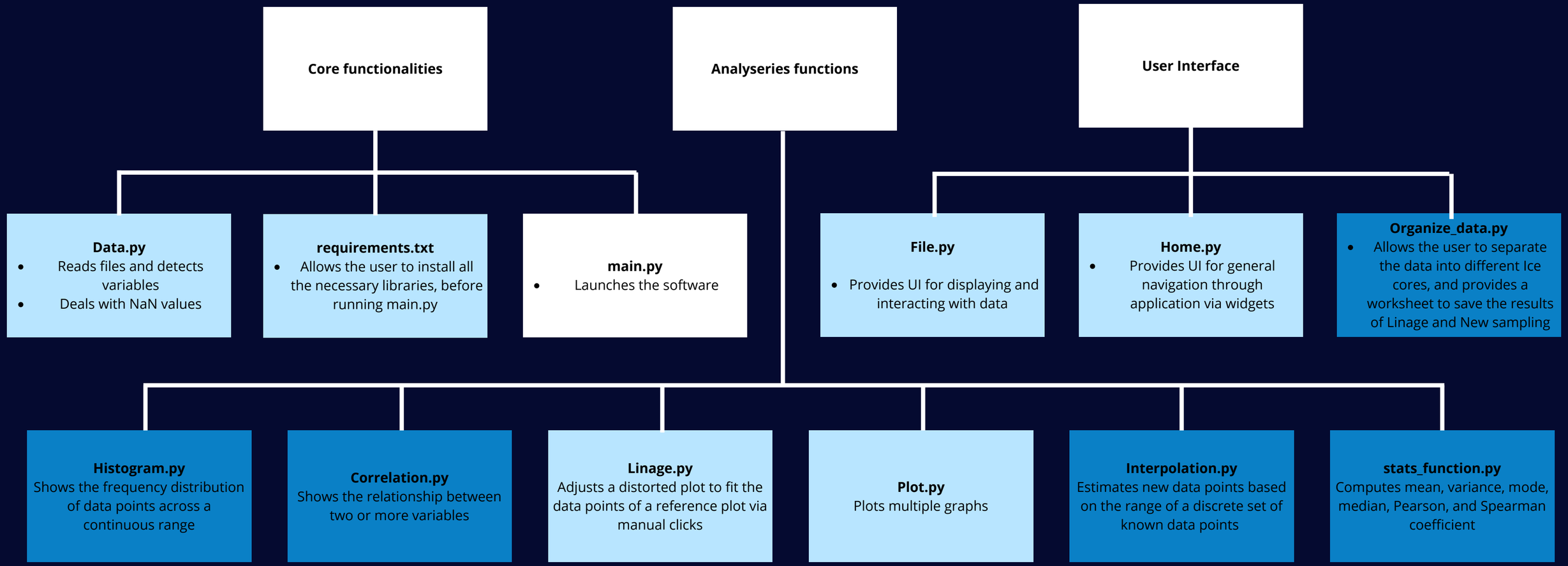


Interpolation via integrals:

→ helps to average out the noise by dividing the function into segments, which are then used to compute the new sampling points

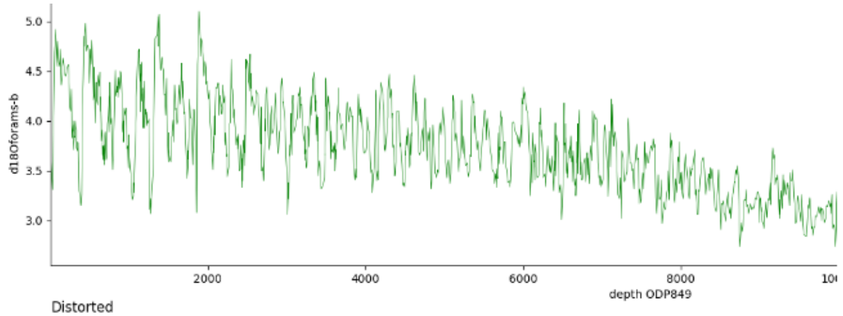
→ can be used in junction with staircase (“step”) and cubic spline interpolation

IMPLEMENTATION AND RESULTS

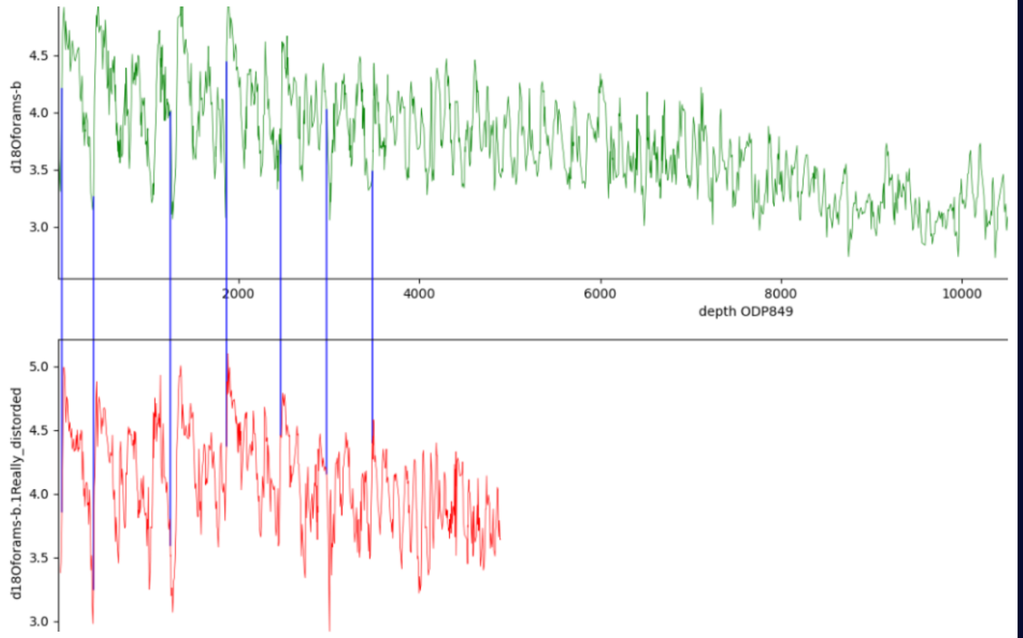


LINAGE

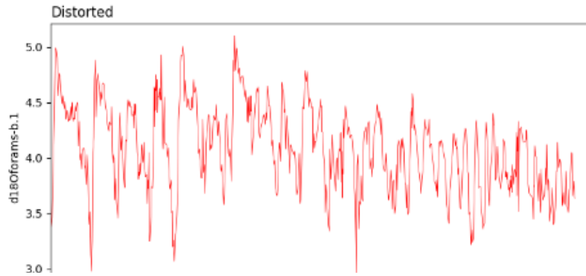
(x_a, y_a)



$(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{ip}, y_{ip})$



(x_b, y_b)



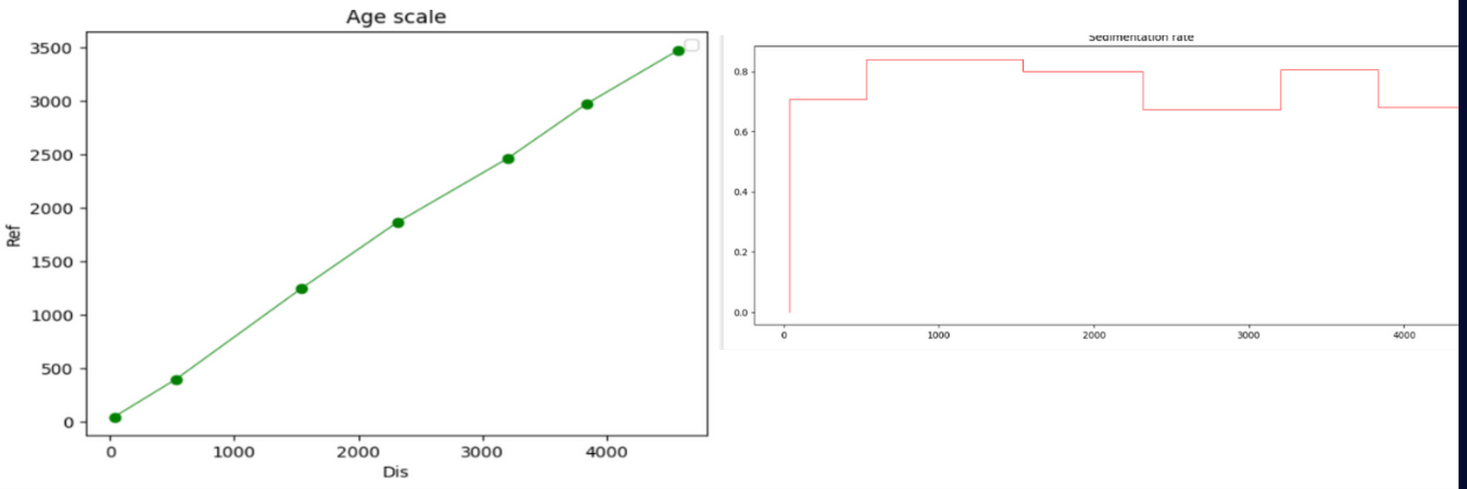
$(x_{j1}, y_{j1}), (x_{j2}, y_{j2}), \ldots, (x_{jp}, y_{jp})$

$$\forall\, k \in \{1, 2, \ldots, p\},\ f(x_{i_k}) = x_{j_k}$$

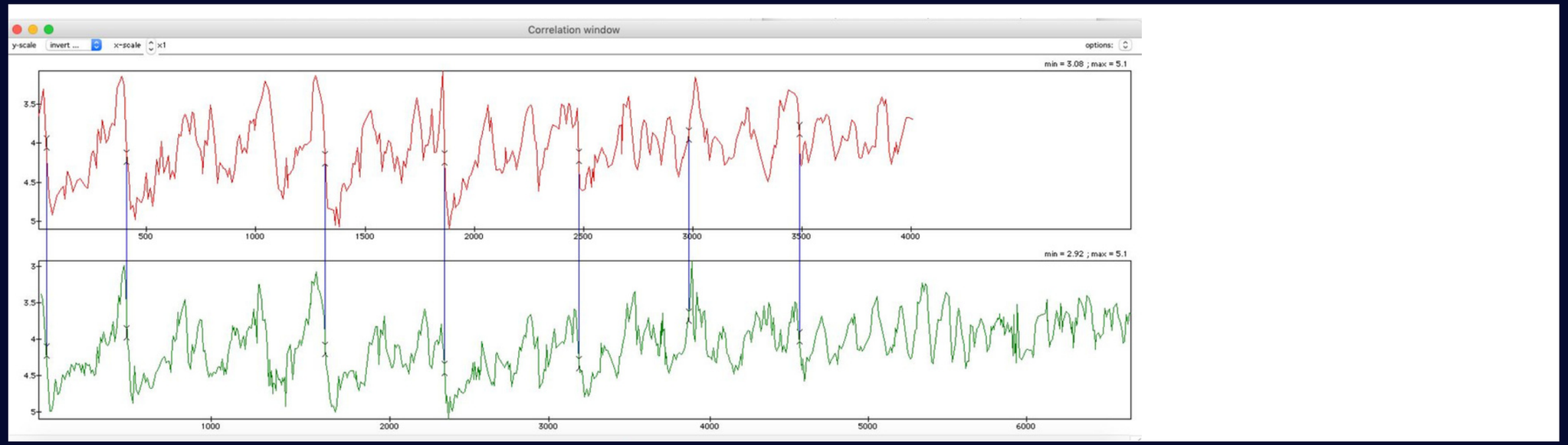
$$\forall\, x \text{ and } k \in \{1, 2, \ldots, p-1\}$$

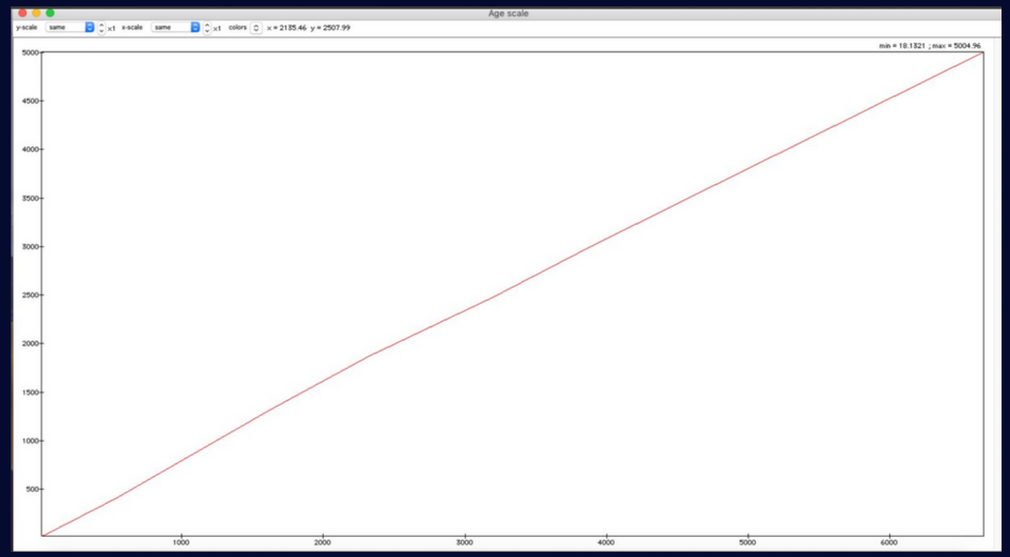
$$\text{such that } x \in [x_{j_k}, x_{j_{k+1}}]$$

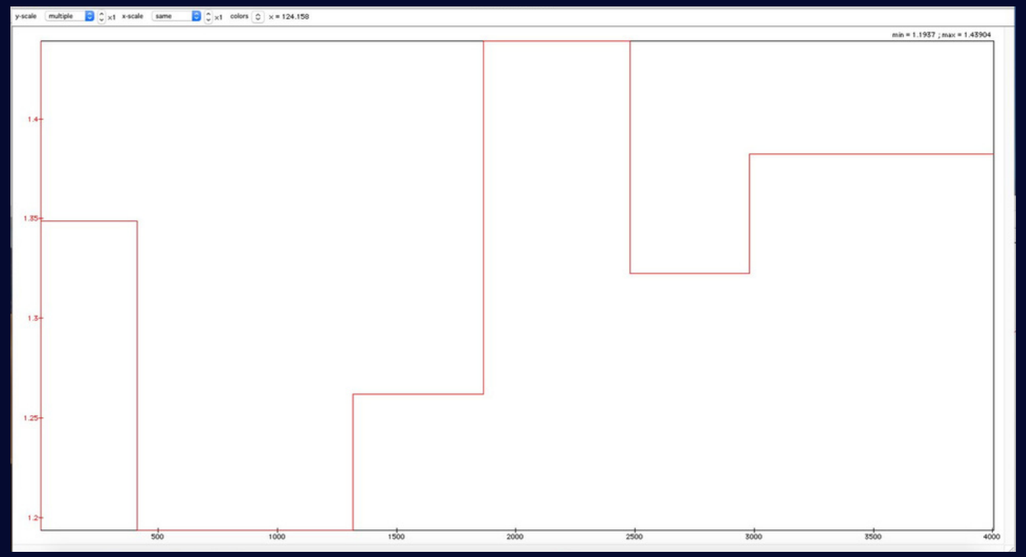
$$f(x) = \frac{x_{i_{k+1}} - x_{i_k}}{x_{j_{k+1}} - x_{j_k}}$$



LINAGE: COMPARISON WITH MAC VERSION

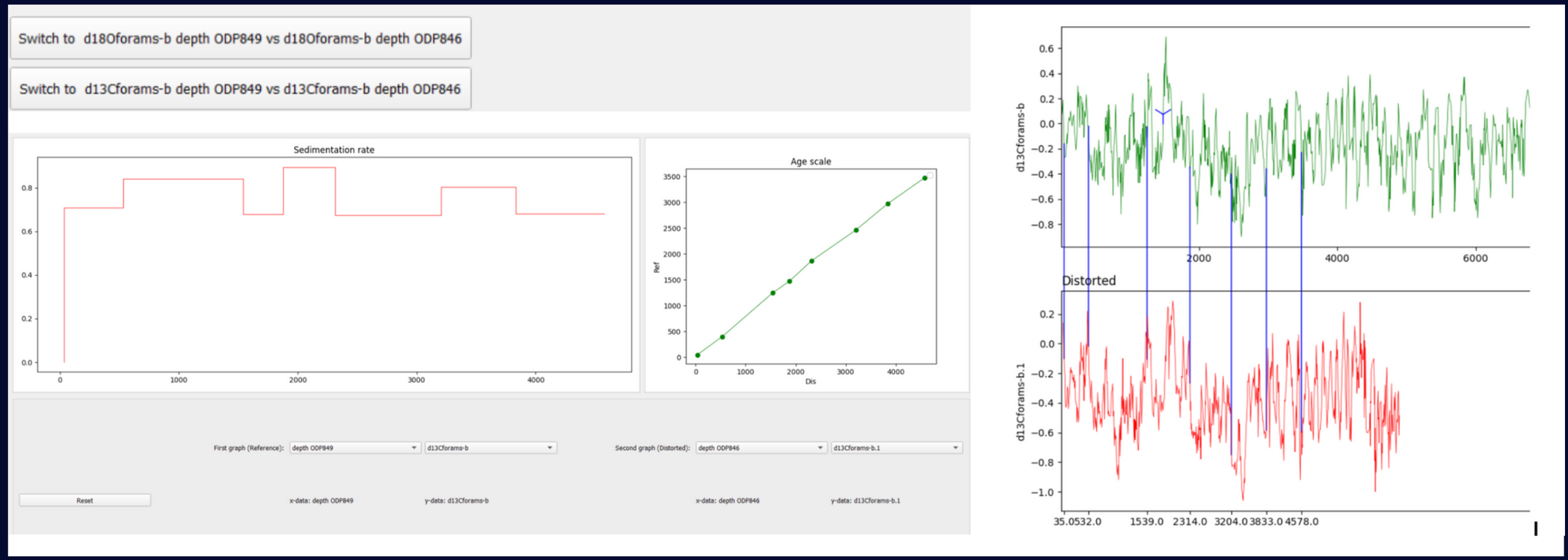






MULTISERIES

- Creation of a single interpolation window
- Creation of buttons to switch between data



DISCUSSION: RESULTS

→ Main objectives:

1. **To align the software with the LSCE's specific needs**
2. **To further modernize the software to enable it to become an open-source, focusing on user-friendliness, efficiency, and the breadth of its features and flexibility**

Aligning software with LSCE's specific needs

- Support for Multiple File Formats ("**.xlsx**" files)
- Addition of the remaining functions into Python)

Modernizing Software for Open-Source

User-Friendliness:

- New "Worksheet" feature
- Intuitive application menu
- Elimination of inadequate data representation
- Correlation function upgrades (multiple file handling and displays up to four plots simultaneously)
- Automatic adjustment in correlation and histogram functions

Efficiency:

- Code optimization
- Support for "**.xlsx**" files
- Automatic adjustment in correlation and histogram functions

The breadth of Features and Flexibility

- Multi-window plotting
- Multiseries feature
- Flexibility in file formats

DISCUSSION:CHALLENGES

- Translating complex mathematical functions into a new programming language (Python)
- Ensuring the software's portability across various operating systems for open-source development
- Addressing the initial lack of comprehensive metadata for ice core
- Developing a new Multiseries feature without cluttering the user interface
- Project Management Challenges: technical expertise gap, balancing learning with client needs, ensuring user acceptance, and effectively leveraging the diverse skills within the team

CONCLUSION: IMPACT OF THE PROJECT AND NEXT STEPS

- **Modernized software:** enhanced user-friendliness, flexibility, and efficiency through advancements in data handling and user interface as well as the introduction of advanced analytical features.

- **Future development:** the contributions have provided the groundwork for future open-source development.

- **Value for the team:** Enhanced technical skills and problem-solving abilities as well as the understanding of user-centric design.

Next Steps:

- **Recalage Automatic:** Automatic Detection of landmarks using wavelet transforms
- **Wavelet transform**: Additional tool for ice cores times series analysis (Application to filtering and denoising, in the pywt library)
- **User interface improvements:** The worksheet does not allow data selection before applying the math functions
- **Code improvements:** PyQT tools are not intuitive, and code is not always easy to follow.
- U**ser Guide update:** To facilitate future development.


Thank you
FOR YOUR ATTENTION!