
Functional Requirements (FR)

- **FR1:** Load external datasets into memory using intuitive syntax (CSV initially, JSON/Excel in future).
- **FR2:** Provide data transformation capabilities including selecting, sorting, sampling, and filtering datasets based on user-defined conditions.
- **FR3:** Support joining multiple datasets on common column(s).
- **FR4:** Allow grouping and aggregating data with functions such as sum, average, min, and max.
- **FR5:** Handle missing data through removal (`dropna`) and filling (`fillna`) strategies.
- **FR6:** Provide concise syntax for applying custom or predefined mathematical and statistical functions to dataset columns.
- **FR7:** Generate comprehensive descriptive statistics, summaries, and metadata of datasets.
- **FR8:** Detect and report outliers within datasets using statistical methods (e.g., IQR).
- **FR9:** Offer normalization methods (e.g., Z-score) and rolling window calculations on datasets.
- **FR10:** Perform basic statistical hypothesis testing (e.g., t-tests, correlations).
- **FR11:** Provide comprehensive visualization capabilities including scatter plots, box plots, heatmaps, pair plots, time series plots, and pie charts.
- **FR12:** Allow exporting transformed datasets (CSV) and generated visualizations (PNG, JPG) to disk.
- **FR13:** Provide quick access to detailed dataset information (e.g., schema, memory usage, data types).
- **FR14:** Clearly define and implement a formal grammar specification using parser-generation tools (Lark).
- **FR15:** Perform comprehensive static semantic and type checking to catch errors and report meaningful, domain-specific feedback.

- **FR16:** Compile Noeta scripts into an intermediate representation (IR) before translating them to Python for optimized and maintainable execution.

- **FR17:** Provide interactive querying capabilities through Jupyter notebook kernel integration, with full support for interactive user inputs and cell-based execution.

- **FR18:** Provide an intuitive and robust CLI interface for direct script compilation and execution.

Non-Functional Requirements (NFR)

- **NFR1:** Ensure the compiled Python code runs efficiently, leveraging optimized libraries such as Pandas, NumPy, and Matplotlib.

- **NFR2:** Minimize compilation overhead to provide near-instantaneous feedback and maintain interactivity.

- **NFR3:** Design the DSL syntax to be intuitive, readable, and beginner-friendly, minimizing the learning curve for new users.

- **NFR4:** Provide meaningful and actionable error messages in DSL terminology, clearly distinguishing Noeta-level errors from Python-level exceptions.

- **NFR5:** Ensure the software maintains stability, reliably handling syntax errors, runtime errors, and user-input mistakes gracefully.

- **NFR6:** Structure the architecture to ensure maintainability and straightforward extensibility, clearly separating parsing, semantic analysis, IR translation, and code generation phases.

- **NFR7:** Ensure compatibility across all major operating systems (Windows, Linux, macOS) and popular Python versions (3.8+).

- **NFR8:** Guarantee compatibility and stable integration with standard Jupyter notebook interfaces (JupyterLab, Classic Notebook, VS Code Notebook).

- **NFR9:** Provide comprehensive and clear documentation, including getting-started guides, detailed tutorials, and API reference material.

- **NFR10:** Maintain openness to community collaboration by hosting the project on accessible open-source platforms (GitHub), providing clear contribution guidelines.
