RNAseq Analysis Agent

A comprehensive AI-powered agent for analyzing RNAseq data with interactive plotting capabilities. This agent can query SQLite databases containing RNAseq analysis results and generate various types of visualizations to help interpret the data.

Features

🔍 Database Operations

- Connect to SQLite databases containing RNAseq data
- Execute SQL queries with safety checks
- Retrieve database schema information
- Handle multiple table types (differential expression, pathway enrichment, etc.)

📊 Interactive Plotting

- **Volcano Plots**: Visualize differential expression results
- **MA Plots**: Show relationship between expression level and fold change
- **Pathway Enrichment Plots**: Display enriched biological pathways
- **Histograms**: Show distribution of values
- **Scatter Plots**: Explore relationships between variables
- **Box Plots**: Display data distributions and outliers
- **Heatmaps**: Visualize correlation matrices
- **Bar Plots**: Show categorical data comparisons

🤖 AI-Powered Analysis

- Natural language query processing
- Automatic plot type selection based on data
- Biological context interpretation
- Statistical significance assessment

Installation

Prerequisites

- Python 3.11+
- Required packages (install via pip):

```
pip install pandas numpy matplotlib seaborn plotly langchain langchain-
mistralai
```

Files Structure

```
 ├── agent.py            # Main agent implementation
├── test_agent.py       # Test script with sample data
├── README.md           # This documentation
└── plots/             # Directory for generated plots
```

Quick Start

1. Basic Usage

```
from agent import RNAseqAgent

# Initialize the agent
agent = RNAseqAgent(
    db_path="your_rnaseq.db",
    mistral_api_key="your_mistral_api_key"
)

# Ask questions about your data
response = agent.ask("What are the top upregulated genes?")
print(response)

# Clean up
agent.close()
```

2. Running Tests

```
python test_agent.py
```

This will:

- Create a sample database with RNAseq data
- Test all plotting functionalities
- Generate example visualizations
- Validate the complete workflow

Database Schema

The agent expects SQLite databases with tables following this naming convention: `{sample_subset}_{comparison}_{analysis_type}_{gene_set}`

Example Tables

Differential Expression Table

```
CREATE TABLE NS_flattening_yes_vs_no_deseq2 (
    gene_id TEXT PRIMARY KEY,
    gene_name TEXT,
    basemean REAL,
    log2fc REAL,
    lfcse REAL,
    stat REAL,
    pvalue REAL,
    padj REAL
);
```

Pathway Enrichment Table

```
CREATE TABLE NS_flattening_yes_vs_no_ora_hallmark (
    pathway TEXT PRIMARY KEY,
    description TEXT,
    enrichment REAL,
    pvalue REAL,
    padj REAL,
    genes_in_pathway INTEGER,
    genes_found INTEGER
);
```

Usage Examples

Example 1: Differential Expression Analysis

```
# Query for significantly upregulated genes
response = agent.ask("""
Show me the top 10 upregulated genes with padj < 0.05 and log2fc > 1.
Also create a volcano plot to visualize the results.
""")
```

Example 2: Pathway Analysis

```
# Analyze enriched pathways
response = agent.ask("""
What are the most significantly enriched pathways in the hallmark gene set?
Create a pathway enrichment plot showing the top 15 pathways.
""")
```

Example 3: Data Exploration

```
# Explore data distribution
response = agent.ask("""
Show me the distribution of log2 fold changes and create a histogram.
Also show me the correlation between different statistical measures.
""")
```

Plot Types and Usage

1. Volcano Plot

- **Purpose**: Visualize differential expression results
- **Usage**: `Create_Plot volcano`
- **Features**:
    - Color-coded significance levels
    - Adjustable thresholds
    - Interactive hover information

2. MA Plot

- **Purpose**: Show relationship between expression level and fold change
- **Usage**: `Create_Plot ma`
- **Features**:
    - Log-scale transformation
    - Reference line at y=0

3. Pathway Enrichment Plot

- **Purpose**: Display enriched biological pathways
- **Usage**: `Create_Plot pathway_enrichment`
- **Features**:

    - Horizontal bar chart
    - Color-coded by significance
    - Customizable number of pathways

4. Histogram

- **Purpose**: Show distribution of values
- **Usage**: `Create_Plot histogram|column=column_name`
- **Features**:
    - Customizable bin count
    - Automatic column selection

5. Scatter Plot

- **Purpose**: Explore relationships between variables
- **Usage**: `Create_Plot scatter|x_column=col1|y_column=col2`
- **Features**:
    - Automatic column selection
    - Interactive zoom and pan

6. Box Plot

- **Purpose**: Display data distributions and outliers
- **Usage**: `Create_Plot boxplot|column=column_name`
- **Features**:
    - Quartile visualization
    - Outlier detection

7. Heatmap

- **Purpose**: Visualize correlation matrices
- **Usage**: `Create_Plot heatmap`
- **Features**:
    - Correlation coefficient display
    - Color-coded intensity

API Reference

RNAseqAgent Class

`__init__(db_path, mistral_api_key)`

Initialize the agent with database path and API key.

`ask(question)`

Process a natural language question and return analysis results.

`close()`

Clean up database connections and resources.

RNAseqDatabase Class

`connect()`

Establish database connection.

`execute_query(query)`

Execute SQL query with safety checks.

`get_table_info()`

Retrieve database schema information.

RNAseqPlotter Class

`store_query_data(data, query_info)`

Store query results for plotting.

`create_plot(plot_type, **kwargs)`

Generate interactive plots from stored data.

Configuration

Environment Variables

- `MISTRAL_API_KEY`: Your Mistral AI API key

Plot Settings

- Output directory: `plots/` (configurable)
- Plot format: HTML (interactive Plotly plots)
- Default theme: `plotly_white`

Troubleshooting

Common Issues

1. **Database Connection Failed**
   - Check if the database file exists
   - Verify file permissions
   - Ensure SQLite3 is available
2. **API Key Issues**
   - Verify your Mistral API key is valid
   - Check API rate limits
   - Ensure internet connectivity
3. **Plot Generation Errors**
   - Ensure data is loaded before plotting
   - Check column names match expected format
   - Verify numeric data types for calculations

Debug Mode

Enable verbose logging:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

Contributing

1. Fork the repository
2. Create a feature branch
3. Add tests for new functionality
4. Ensure all tests pass
5. Submit a pull request

License

This project is licensed under the MIT License.

Changelog

Version 1.0.0

- Initial release

- Complete plotting functionality
- Database operations
- AI-powered query processing
- Comprehensive test suite

Support

For issues and questions:

1. Check the troubleshooting section
2. Review the test examples
3. Create an issue with detailed error information

Note: This agent requires a valid Mistral AI API key for natural language processing. The plotting functionality works independently and can be used without the AI features.