

# BioNeuralNet

licenseMIT

pypipackage or version not found

pythonpackage or version not found

**BioNeuralNet** is designed to integrate omics data with neural network embeddings, facilitating advanced data analysis in bioinformatics. Leveraging Python, BioNeuralNet constructs and analyzes complex biological networks, ensuring comprehensive and scalable data processing.

## Table of Contents

1. Overview

2. Features

3. Installation

- For Users: Python Package Installation
- For Users: R Dependencies
- For Developers: Development Environment Setup

4. Quick Start Guide

5. Pipeline Components

6. Acknowledgements

7. Documentation

8. Testing

9. Contributing

10. License

11. Contact

## Overview

BioNeuralNet empowers researchers to seamlessly integrate various omics datasets using advanced neural network embeddings. By combining the strengths of Python for data manipulation and analysis, BioNeuralNet provides a comprehensive toolkit for bioinformatics applications.



## Features

- Graph Construction:** Utilize SmCCNet and WGCNA algorithms to build complex biological networks.
- Clustering:** Perform clustering using methods like PageRank and Hierarchical Clustering.
- Network Embedding:** Generate high-dimensional embeddings using Graph Neural Networks (GNNs) and Node2Vec.
- Subject Representation:** Enhance omics data representations for downstream analyses.
- Comprehensive Testing:** Ensure reliability with a robust testing suite and continuous integration.
- Developer-Friendly:** Streamlined setup for contributors with pre-commit hooks and development tools.

# Installation

## For Users: Python Package Installation

Most users can install **BioNeuralNet** directly via [pip](#), which includes all necessary Python dependencies.

### 1. Ensure Python 3.7 or Higher is Installed

Verify your Python version:

```
python3 --version
```

### 2. Install BioNeuralNet via Pip

```
pip install bioneuralsnet
```

*This command installs the latest stable release from PyPI, including all base dependencies.*

**Note:** If you require CUDA-enabled functionalities (for GPU acceleration), ensure that you have the appropriate CUDA version installed on your system.

## For Users: R Dependencies

**BioNeuralNet** integrates R scripts for graph construction using SmCCNet and WGCNA. While Python users can install the package via [pip](#), R dependencies need separate installation. Most users do not require R functionalities unless they intend to utilize these specific graph construction methods.

### 1. Manual R Installation

If you prefer manual installation or are on an unsupported operating system, follow these steps:

#### a. Install R

- **Download R:**

Visit the [CRAN R Project](#) and download the appropriate installer for your operating system.

- **Install R:**

Follow the installation instructions provided on the CRAN website for your specific OS.

#### b. Install Required R Packages

Open R or RStudio and execute the following commands to install necessary packages:

```
install.packages(c("dplyr", "SmCCNet", "WGCNA"))
```

**Notes:**

- **System Dependencies:** Some R packages might require additional system dependencies. Ensure you have the necessary build tools installed (e.g., `gcc`, `make`).
- **Permissions:** You may need administrative privileges to install certain packages or dependencies.

**For Developers: Development Environment Setup**

Researchers and developers intending to contribute to **BioNeuralNet** should set up a development environment. This involves creating a virtual environment, installing dependencies, and setting up pre-commit hooks to maintain code quality.

**1. Clone the Repository**

```
git clone https://github.com/UCD-BDLab/BioNeuralNet.git
cd BioNeuralNet
```

**2. Run the Setup Script**

The `setup.sh` script automates the setup process for developers.

```
./setup.sh
```

**What `setup.sh` Does:**

- **Creates and Activates a Virtual Environment:** Ensures that dependencies are isolated.
- **Installs Base and Development Dependencies:** Sets up the environment with necessary packages.
- **Installs Pre-Commit Hooks:** Automates code quality checks before commits.
- **Initializes the Project:** Sets up directories and configuration files.

**Note:** Ensure you have execution permissions for `setup.sh`. If not, make it executable:

```
chmod +x setup.sh
```

**3. Verify Pre-Commit Hooks**

After running `setup.sh`, pre-commit hooks should be installed automatically. To confirm, you can run:

```
pre-commit run --all-files
```

*This will execute all configured pre-commit hooks on the entire codebase.*

## 4. Install R Dependencies (If Needed)

If your development work involves R scripts, ensure that R and its required packages are installed as per the [R Dependencies](#) section.

---

## Quick Start Guide

Begin using **BioNeuralNet** by following these streamlined steps:

### 1. Prepare Input Data

- **Omics Data:** Place your omics CSV files (e.g., `proteomics_data.csv`, `metabolomics_data.csv`) in the `input/` directory.
- **Phenotype Data:** Place `phenotype_data.csv` in the `input/` directory.
- **Clinical Data:** Place `clinical_data.csv` in the `input/` directory.

### 2. Combine Omics Data

If you have multiple omics datasets, combine them using the `combine_omics_data` utility.

```
from bioneuralsnet.utils.data_utils import combine_omics_data

omics_file_paths = [
    './input/proteomics_data.csv',
    './input/metabolomics_data.csv'
]

combined_omics_file = './input/omics_data.csv'

combine_omics_data(omics_file_paths, combined_omics_file)
```

### 3. Run the Example Usage Script

Execute the example usage script or create your own pipeline script to perform data integration and analysis.

```
python examples/usage_examples.py
```

*This script demonstrates how to use various components of BioNeuralNet with direct parameter passing.*

---

## Pipeline Components

BioNeuralNet's pipeline consists of several interconnected components:

### 1. Graph Construction

- **SmCCNet**: Builds graphs based on higher-order correlations.
- **WGCNA**: Constructs weighted correlation networks and detects modules.

## 2. Clustering

- **PageRankClustering**: Clusters nodes based on personalized PageRank.
- **HierarchicalClustering**: Performs agglomerative hierarchical clustering.

## 3. Network Embedding Generation

- **GNNEmbedding**: Generates embeddings using Graph Neural Networks.
- **Node2VecEmbedding**: Generates embeddings using the Node2Vec algorithm.

## 4. Subject Representation

- **SubjectRepresentationEmbedding**: Integrates node embeddings into omics data to enhance subject representations.



## 5. Integrated Tasks

- **Disease Prediction using Multi-Omics Networks**: Leverages the power of Graph Neural Networks (GNNs) to capture intricate relationships between biological entities and extract valuable knowledge from this network structure.



## 6. Utility Functions

- Includes tools for data manipulation, path validation, and more.

---

# Acknowledgements

BioNeuralNet utilizes several external packages and libraries that are integral to its functionality. We extend our gratitude to the developers and contributors of these projects:

- **SmCCNet**

- *Description*: An R package for Sparse Multiple Canonical Correlation Network.
- *Repository*: [SmCCNet on CRAN](#)

- **WGCNA**

- *Description*: Weighted Correlation Network Analysis for R.
- *Repository*: [WGCNA on CRAN](#)

- **Node2Vec**

- *Description*: Scalable Feature Learning for Networks.
- *Repository*: [Node2Vec GitHub](#)

- **Other Libraries**

- **dplyr**: A grammar of data manipulation for R. [dplyr on CRAN](#)
- **PyTorch**: An open source machine learning library based on the Torch library. [PyTorch Official Site](#)
- **PyTorch Geometric**: Extension library for PyTorch to handle geometric data. [PyTorch Geometric GitHub](#)
- **Pytest**: A framework that makes building simple and scalable tests easy. [Pytest Official Site](#)
- **Pre-commit**: A framework for managing and maintaining multi-language pre-commit hooks. [Pre-commit GitHub](#)
- **Other Dependencies**: Refer to [requirements.txt](#) for a complete list of Python dependencies.

*Thank you to all the open-source communities that make projects like BioNeuralNet possible.*

---

## Documentation

Comprehensive documentation is available to help you navigate and utilize all features of **BioNeuralNet**.

- **Main Documentation**: Located in the [docs/](#) directory, providing detailed guides and usage instructions.
- **API Reference**: Detailed API documentation is available [here](#). *(Replace with your actual documentation URL)*
- **Testing Documentation**: Refer to [tests/README.md](#) for information on running and writing tests.
- **Additional Guides**: Explore other [README.md](#) files within subdirectories for specific component details.

*Note*: The **README.md** provides an overview and essential instructions, while the **API Reference** offers in-depth technical details about the package's classes, functions, and methods.

---

## Testing

Ensuring the reliability of **BioNeuralNet** is paramount. Automated tests run on every commit and pull request via GitHub Actions, and pre-commit hooks enforce local testing before code is committed.

### Overview

- **Testing Framework**: Utilizes [pytest](#) for writing and running tests.
- **Continuous Integration**: GitHub Actions runs tests on multiple Python versions, checks code quality, and reports coverage.
- **Local Enforcement**: Pre-commit hooks automate tests and code quality checks before commits.

### Running Tests Locally

Detailed instructions are available in [tests/README.md](#), but here's a quick overview:

#### 1. Ensure Development Dependencies are Installed

```
pip install -r requirements-dev.txt
```

---

## 2. Run All Tests

```
pytest
```

## 3. View Coverage Reports

```
pytest --cov=bioneuronet --cov-report=html tests/
```

Open [htmlcov/index.html](#) in your browser to view the coverage details.

## Continuous Integration

Every commit and pull request triggers GitHub Actions workflows that:

- Install dependencies.
- Lint and format code.
- Run the test suite.
- Upload coverage reports to Codecov.

Ensure that all tests pass in the CI pipeline before merging changes.

---

## Contributing

Contributions are welcome! To ensure a smooth collaboration process, please adhere to the following guidelines:

### 1. Fork the Repository

```
git clone https://github.com/UCD-BDLab/BioNeuralNet.git
cd BioNeuralNet
```

### 2. Create a Feature Branch

```
git checkout -b feature/your-feature-name
```

### 3. Install Development Dependencies

```
./setup.sh
```

### 4. Make Your Changes

- Write clean, readable code following PEP 8 standards.
- Add or update tests as necessary.
- Update documentation if your changes affect usage or functionality.

## 5. Run Tests Locally

```
pytest
```

## 6. Commit Your Changes

```
git add .  
git commit -m "Add feature XYZ"
```

*Pre-commit hooks will run automatically, ensuring code quality and passing tests before the commit is finalized.*

## 7. Push to Your Fork

```
git push origin feature/your-feature-name
```

## 8. Open a Pull Request

Navigate to the original repository and open a pull request detailing your changes.

## Pre-Commit Hooks

To maintain code quality, pre-commit hooks are enforced. After setting up your development environment, ensure that pre-commit hooks are installed:

```
pre-commit install
```

*These hooks will automatically run tests, format code, and perform linting before each commit.*

## Guidelines for Writing Tests

Refer to [tests/README.md](#) for comprehensive guidelines on writing effective and consistent tests.

---

## License

This project is licensed under the [MIT License](#).

---

## Contact



For questions, support, or contributions, please open an issue on [GitHub](#) or contact the maintainers directly.

---

## Final Notes

- **Primary Installation Method:** Most users can seamlessly install **BioNeuralNet** using `pip install bioneuralnet`, which handles all necessary Python dependencies.
- **R Dependencies:** R scripts required for SmCCNet and WGCNA are not installable via `pip`. Users needing these functionalities should follow the R installation instructions provided [above](#).
- **Usage with Direct Parameter Passing:** All components in BioNeuralNet can be used by directly passing parameters to the classes, without the need for configuration files. Refer to the example usage scripts in the `examples/` directory.
- **Developer Setup:** Developers contributing to the project should use the `setup.sh` script to establish their development environment, ensuring consistency and adherence to project standards.
- **Documentation and Testing:** Comprehensive documentation and a robust testing suite ensure that both users and contributors have the resources needed to effectively utilize and enhance **BioNeuralNet**.

By maintaining a clear and organized `README.md`, we aim to facilitate ease of use for end-users and streamline the development process for contributors. This structured approach enhances the overall quality and maintainability of the **BioNeuralNet** project.

---