Technical Report: Biofilter System Status Update

Summary

Over the past two weeks, we temporarily paused the development of unit tests to focus on the Issues Tests, addressing several GitHub issues which are detailed further in this report. Significant progress was also made in transitioning from Setuptools to Poetry, implementing tox package for multi-environment testing, and integrating Sphinx for project documentation. Additionally, we launched the first version of the project's technical documentation, published on a new branch in the Biofilter repository, and successfully implemented the first GitHub Actions workflow for the automated deployment of this documentation.

Key Actions Taken

Tests for Reported Issues

Biofilter Group Annotation #15

The primary goal of this test was to validate the consistency of gene annotations across Biofilter versions 2.4.2 and 2.4.3. A discrepancy was observed in version 2.4.2 where reprocessing a subset of genes previously excluded from annotations yielded new results, raising concerns about the system's consistency.

Findings:

- GRCh37 vs. GRCh38 Build Compatibility: Many genes missing from the first execution were due to a mismatch in genome builds. These genes belonged to GRCh37, while the current LOKI database supports GRCh38.
- Duplicate ENSEMBL Codes: Some genes in the second execution were processed due to duplicate ENSEMBL codes pointing to the same biopolymer_id. The Biofilter system, when the --allow-duplicate-output argument is not activated, only considers the first ENSEMBL gene linked to a biopolymer_id and ignores subsequent duplicates.
- Annotation Inconsistencies: The behavior in version 2.4.3 aligns with the expected logic, demonstrating that the current implementation does not have systemic issues. However, certain ambiguities in the data (e.g., unassigned source_id values) and database structure (e.g., inconsistent relationships between GRCh37 and GRCh38 genes) may require clarification or improvement in future versions.

Prepared by Andre Rico, App Dev
Date: December 04, 2024

- A script (manual_query.py) was developed to simulate queries directly in the LOKI database for troubleshooting and validation.

Proposed Actions:

1. Enhance logging mechanisms in Biofilter to explicitly list missing and ambiguous genes during execution.
2. Evaluate the feasibility of supporting GRCh37 Ensembl codes in LOKI or providing clear documentation on unsupported builds.
3. Address ambiguities in the source_id field within the group_biopolymer table to improve clarity for debugging.

Team Feedback:

- As shared in the GitHub Issue, the system is working as expected. However, these findings highlight areas for discussion in the next system meeting. Pending team approval, the issue can be closed once all concerns are addressed.
- One point we recently discussed is the possibility that the Genes that are pointing to the same biopolymer_id are variations (they do not have the .x suffix and are different Ensembl codes). I would need help analyzing this.

GitHub Issue: https://github.com/RitchieLab/biofilter/issues/15

GitHub Tests: https://github.com/RitchieLab/biofilter/tree/development/tests/issues/b15_biofilter_group_annotation

*Bu**ild** 37 LOKI #16*

This issue investigates a discrepancy in the compatibility of the LOKI database with genome assemblies GRCh37 (b37) and GRCh38 (b38). While the Biofilter system supports both assemblies, the GRCh37 version of the LOKI database was last updated in 2014, whereas the GRCh38 version was updated in 2022. This creates challenges for users relying on the outdated GRCh37 build, potentially impacting the accuracy of their analyses.

Findings:

- Version Mismatch: Users intending to provide positions from GRCh37 encounter difficulties as the current LOKI database is built on GRCh38.
- Dynamic Assembly Mapping: A test scenario was implemented to address this issue. The test demonstrated that Biofilter detects discrepancies between the user-provided assembly version (--ucsc-build-version, e.g., "19" for GRCh37) and the LOKI database's build version. When a mismatch is identified, Biofilter employs a conversion table (e.g., LiftOver) to map input positions from one assembly to another (GRCh37 to GRCh38).

- Output Consistency: The output retains the original input positions as labels (e.g., GRCh37/hg19), while the results are aligned with the GRCh38 assembly used by LOKI.

Team Feedback:

- Unified Query Handling: Ensure all queries are processed using coordinates aligned with the LOKI database's assembly while maintaining the original input positions as labels.
- As we plan future versions of Biofilter and LOKI, we need to decide whether to maintain the current single-build structure (supporting only GRCh38) or introduce multi-build support to include data from other assemblies like GRCh37. Supporting multiple builds would enhance flexibility for users but increase system complexity and maintenance demands. Alternatively, sticking to a single-build approach simplifies development and ensures consistency, with users handling conversions externally. This decision requires input from the team to align with user needs and resource availability.

GitHub Issue: https://github.com/RitchieLab/LOKI/issues/16

GitHub Tests: https://github.com/RitchieLab/biofilter/tree/development/tests/issues/l16_build_37_loki

Fix biofilter usage of NCBI symbol vs official gene name #11

While annotating a list of 270 genes using Biofilter, six genes were not recognized and failed to return results. Further investigation revealed that these six genes are synonymous with official gene names in the NCBI database. Below is the list of input genes and their corresponding official names:

- Input Gene: MKL2 → Official Gene Name: MRTFB
- Input Gene: EBI2 → Official Gene Name: GPR183
- Input Gene: LOC729974 → Official Gene Name: RFPL4AL1
- Input Gene: PSFL → Official Gene Name: APH1B
- Input Gene: FBXL10 → Official Gene Name: KDM2B
- Input Gene: C2ORF7 → Official Gene Name: PRADC1

It was observed that when Biofilter was rerun using the official gene names, the system returned appropriate annotations for these genes. However, using the synonymous names as input failed to produce results.

Analysis

1. Presence in the LOKI Database: All six genes, except *LOC729974*, are present in the LOKI database. When searching for *LOC729974* in NCBI, it returns *RFPL4AL1* as a

result. However, no reference explicitly confirms *LOC729974* as a synonym for *RFPL4AL1*. Further investigation is required using NCBI APIs to validate this linkage.

2. Handling Synonymous Names: Biofilter currently relies on the Symbol field from the NCBI Gene API to match input names. However, the NCBI API also provides a Synonyms field that lists alternate names for the same gene. Expanding Biofilter to check both the Symbol and Synonyms fields could allow it to resolve synonymous names and return annotations for these cases.
3. Potential Input Conflict: For the five recognized genes, the use of official names in the input might override their alternate names, avoiding duplication in the output. This behavior could explain why some synonymous names fail to produce results.

Next Steps:

To better analyze this issue and validate the proposed solutions, it is necessary to obtain:

- The full list of 270 input genes.
- The exact arguments provided in the Biofilter command during execution.

```
SELECT
  bp.biopolymer_id,
  bp.type_id,
  bpn.namespace_id,
  bp.label AS official_name,
  bpn.name AS alternative_name,
  bp.description
FROM
  biopolymer AS bp
INNER JOIN
  biopolymer_name AS bpn
  ON bp.biopolymer_id = bpn.biopolymer_id
WHERE
  bpn.name IN ('MKL2', 'EBI2', 'LOC729974', 'PSFL', 'FBXL10', 'C2orf7' );
```

	biopolymer_id	type_id	namespace_id	official_name	alternative_name	description
1	2643275	2	2	PRADC1	C2orf7	protease associated domain containing 1
2	2628876	2	2	GPR183	EBI2	G protein-coupled receptor 183
3	2643457	2	2	KDM2B	FBXL10	lysine demethylase 2B
4	2640995	2	2	MRTFB	MKL2	myocardin related transcription factor B
5	2642948	2	2	APH1B	PSFL	aph-1 homolog B, gamma-secretase subunit

Poetry Dependency Management

Overview

The transition from Setuptools to Poetry for dependency management has been successfully implemented in the Biofilter project. This change enhances control over dependencies, improves the project's structure, and simplifies the process of managing development environments.

System Dependency Management:

- All system dependencies, such as libraries required for running the Biofilter, have been clearly defined in the [tool.poetry.dependencies] section of the pyproject.toml file.
- This ensures that the correct versions of each dependency are installed, avoiding compatibility issues.

Development Dependencies:

- A separate development group ([tool.poetry.group.dev.dependencies]) has been configured to include tools like linters (e.g., Black), testing frameworks (e.g., Pytest, Coverage), and documentation generators (e.g., Sphinx).
- Developers working on the project can install these additional tools by running:

  $ poetry install --with dev

Python Version Control:

- The pyproject.toml file explicitly defines the supported Python versions using the python field. For example:

  python = "^3.10 || ^3.11 || ^3.12"

- This ensures the project is not installed or executed in unsupported Python environments, reducing errors due to version mismatches.

Global Accessibility:

- Entry points for the Biofilter application have been defined in the [tool.poetry.scripts] section, allowing the tool to be executed globally once installed. Example:

  [tool.poetry.scripts]

  biofilter = "biofilter_modules.biofilter:main"

  loki-build = "loki_modules.loki_build:main"

- After installing the Biofilter package, users can run commands like:

  $ biofilter --help

  $ loki-build --help

Benefits of the Transition

Improved Dependency Management:

- Dependencies are now centralized in a single file (pyproject.toml), providing a clear overview of the project's requirements.
- Locking exact dependency versions via poetry.lock ensures consistent environments across systems.

Streamlined Development Setup:

- Developers can easily set up a fully configured environment, including all development tools, using a single command:

  $ poetry install --with dev

Enhanced Python Compatibility:

- By specifying supported Python versions, the project avoids installation in untested environments, ensuring stability and reliability.

Simplified Packaging and Distribution:

- Building and publishing the Biofilter package is now easier with Poetry's built-in commands:

  $ poetry build

  $ poetry publish

Conclusion

The adoption of Poetry has significantly improved the management and distribution of the Biofilter project. By providing better control over dependencies, environment consistency, and Python versioning, this change lays a solid foundation for future development and collaboration.

Tox Tool

Overview

Tox has been integrated into the Biofilter project as a development dependency to streamline testing across multiple Python versions. It provides an automated and isolated environment for running tests, ensuring compatibility and consistency in various Python setups.

Key Details

Development Dependency:

Tox has been included in the dev group of dependencies in *pyproject.toml* and is installed alongside other development tools:

$ poetry install --with dev

Configuration:

The *tox.ini* file in the root directory specifies the Python versions to test and the commands to execute:

```
[tox]
envlist = py310, py311, py312
[testenv]
description = Run tests with pytest for {envname}
deps = poetry
```

```
commands =
  poetry install --no-root
  pytest tests/
```

The *envlist* defines the Python versions (py310, py311, py312) to run tests against, ensuring the project is tested for compatibility across these environments.

Execution:

- Run tests across all Python versions specified in tox.ini using:

  $ poetry run tox

To test a specific version:

$ poetry run tox -e py311

Python Interpreter Paths:

If Tox cannot locate the required Python versions, update the tox.ini file with specific paths for each version:

```
[testenv:py310]
basepython = /path/to/python3.10
```

Benefits

- Multi-Version Compatibility: Ensures that the Biofilter project is functional across different Python versions.
- Automated Testing: Simplifies the process of running tests with isolated environments for each Python version.
- Integration with CI/CD: Can be configured to run automatically during GitHub Actions workflows for consistent validation.

Conclusion

By integrating Tox, the Biofilter project ensures robust and consistent testing across supported Python versions, improving the reliability of the codebase and facilitating a smoother development process.

SPHINX Documentation Management

Overview

Sphinx has been implemented in the Biofilter project to create and maintain technical documentation for both users and developers. It provides a flexible and extensible system to generate HTML, PDF, and other formats for project documentation.

Key Details

Dual Documentation Setup:

- The project includes two distinct documentation directories:
- docs/: Contains user-focused documentation, hosted on Read the Docs.
- docs-dev/: Contains developer-focused technical documentation, deployed to GitHub Pages.

Sphinx Configuration

Both directories are independently configured with their respective conf.py files tailored to their audience:

- docs/conf.py: Focuses on features and usage of Biofilter for end-users.
- docs-dev/conf.py: Documents developer tools, workflows, and technical details like Tox, Poetry, and CI/CD integration.

Deployment:

- User Docs: Deployed via Read the Docs to provide end-users with up-to-date guidance.
- Developer Docs: Automatically deployed to GitHub Pages using GitHub Actions.

Benefits

- Separation of Concerns: Clear distinction between user and developer documentation ensures targeted and effective communication.
- Automation: GitHub Actions ensure developer documentation is deployed with every update to the development branch.
- Customization: Developers can extend Sphinx with themes and plugins, such as sphinx_rtd_theme and autodoc, for professional and maintainable documentation.

Conclusion

The integration of Sphinx improves project transparency and accessibility by providing well-structured and up-to-date documentation for both end-users and contributors. This dual documentation approach ensures all stakeholders have the information they need to effectively use and contribute to the Biofilter project.

Project Documentation

Overview

The newly established technical documentation serves as a comprehensive resource for onboarding new developers and guiding advanced users. Its primary goal is to ensure that all tools, configurations, and workflows are well-documented, promoting alignment with best development practices across the Biofilter and LOKI systems.

Biofilter Developer Docs: https://ritchielab.github.io/biofilter/

Key Highlights

Comprehensive Resource:

- The documentation covers every aspect of the Biofilter and LOKI systems, including tools, workflows, and configurations necessary for effective development and maintenance.
- It acts as a central hub for information, reducing the learning curve for newcomers and providing clarity on advanced system features.

Targeted Audience:

- New Developers: Step-by-step instructions and guidelines to set up the development environment, understand the codebase, and start contributing.
- Advanced Users: Detailed insights into the inner workings of the systems, enabling advanced customization and troubleshooting.

Key Inclusions:

- Development Tools:
  - Usage of Poetry for dependency and environment management.
  - Setup and execution of Tox for multi-version testing.
  - Integration of Black for consistent code formatting.
  - Measurement of test coverage with Coverage.py.
  - Automation of tests and deployments via GitHub Actions.
- Configurations:
  - Documentation on branching strategies for seamless collaboration.
  - Parameterization and best practices for both Biofilter and LOKI.
- Workflows:
  - How to use tools like Sphinx for generating and maintaining user and developer documentation.
  - Guidance on linking code updates to GitHub Issues for traceability.

Purpose:

- To align the Biofilter and LOKI systems with modern development standards.
- To foster collaboration and maintainability by providing clear, actionable documentation.

Benefits

- Consistency Across Tools: Ensures all contributors are aligned with the same workflows, reducing potential discrepancies.
- Ease of Onboarding: New developers can quickly familiarize themselves with the project, leading to faster contributions.
- Improved Maintenance: Advanced users can utilize detailed documentation to debug, optimize, and extend system functionality.
- Future-Proofing: Comprehensive documentation ensures long-term maintainability and scalability of both Biofilter and LOKI.

Conclusion

The technical documentation is designed to be the first point of contact for anyone looking to work on or with the Biofilter and LOKI systems. By detailing all essential tools, configurations, and workflows, it establishes a solid foundation for sustainable and high-quality development aligned with best practices. This initiative strengthens the project's reliability and accessibility for all stakeholders.

New gh-pages Branch and Github Actions Integrations

Overview

As part of the recent improvements to the Biofilter project, we have implemented a dedicated gh-pages branch to host the technical documentation for developers. This branch works in tandem with GitHub Actions for automated deployment, ensuring the documentation remains up-to-date with every relevant change in the codebase.

gh-pages Branch

The gh-pages branch is a specialized branch in the repository designed to:

- Serve as the **host for the technical documentation**, making it easily accessible through GitHub Pages.
- Automate the deployment of updates whenever documentation files in the docs-dev directory are modified.

Key Characteristics:

- **Read-Only**: Developers should not make manual changes to this branch. It is entirely managed by GitHub Actions workflows.
- **Generated Content**: The branch is recreated automatically during deployment, containing only the built documentation files.
- **Hosted Pages**: The branch provides a web-friendly view of the documentation, accessible directly through GitHub Pages.

GitHub Actions for Deployment

GitHub Actions has been configured to automate the build and deployment of the documentation to the gh-pages branch. This workflow ensures the technical documentation reflects the latest changes without requiring manual intervention.

Key Features:

- Triggering Events:
    - The workflow is triggered on every push to the development branch or changes in the docs-dev directory.
- Build Process:
    - The Sphinx documentation files in docs-dev are compiled into HTML format.
    - Any errors in the build process are flagged in the CI pipeline to maintain documentation integrity.
- Automated Deployment:

    - Once the documentation is successfully built, it is deployed to the gh-pages branch.
    - The branch is completely overwritten to ensure a clean and consistent deployment.
- Workflow File:
    - The docs-dev.yml file in the .github/workflows/ directory defines the deployment pipeline.

Benefits of this Approach

- **Automation**: Reduces manual effort by automating the entire build and deployment process.
- **Consistency**: Ensures that the hosted documentation always reflects the latest code and configurations.
- **Accessibility**: Provides an easy-to-navigate, hosted version of the technical documentation, improving accessibility for developers.

Conclusion

The introduction of the gh-pages branch and GitHub Actions integration represents a significant step toward maintaining high-quality, up-to-date documentation. By automating deployment, we ensure that developers have reliable and immediate access to the most recent information, enabling smoother collaboration and development workflows.

Next Steps

Finalize Version 2.4.4

- Complete unit and functional tests to validate stability.
- Resolve remaining GitHub issues reported by the team.
- Prepare for release using the LOKI database on the LPC environment.