Priyanka Talur

BIOEN 537

Component Specifications

Overview

SBVisualizer is a Python-based library designed to create and manage visualization, supporting SBML (Systems Biology Markup Language), specifically, the library SBMLDiagrams as well as tellurium. Users are able to generate detailed drawings of biological and chemical reaction networks, with a focus on illustrating reaction fluxes in a network.

Components

1. Tellurium Model Parser
    a. Can read tellurium models
    b. Can load them into SBMLDiagrams
        i. **loadIntoSBML()** to read and load into SBMLDiagrams
2. Visualizer
    a. Visualizer class that takes in a tellurium model – includes all functions listed in Components
    b. Generates species to species models with arrows in between representing reactions
    c. Flux visualization through arrow thickness and color
        i. **get_fluxes()** computes fluxes of all reactions in the network, no inputs, outputs a dictionary mapping reaction name to flux
        ii. **get_line_thickness()** scales the thickness of the reaction arrows based on the reaction fluxes (from **get_fluxes()**) scaled to a set minimum and maximum, no inputs, outputs a dictionary mapping reaction name to thickness
        iii. **set_line_thickness()** sets the line thicknesses in SBMLDiagrams, inputs required are the **get_line_thickness()** dictionary, no outputs
        iv. **get_color_gradient()** computes the RGB tuple values based on flux values, no inputs, outputs a dictionary mapping reaction names to RGB values
        v. **set_colors()** sets the colors in SBMLDiagrams
3. File Output
    a. Outputs final diagram to a .png or .pdf file
        i. **draw()** draws the reaction diagrams using SBMLDiagrams and outputs it into a file, user can scale to their liking

Dependencies

- Python Libraries:

    - SBMLDiagrams
    - tellurium
- System Requirements:
    - Python 3.8 to 3.11
    - Compatible with Linux, macOS, and Windows.

Interactions to Accomplish Use Cases

The user first creates a Visualizer object with their tellurium model as the input.

1. Setting Line Thickness
    a. The user can write their Antimony model, load it into tellurium, and use **loadIntoSBML()** to load their model into SBMLDiagrams. The user can then get reaction fluxes with **get_fluxes()**, and use that to determine line thicknesses for the network with **get_line_thickness()**. The user can then use this to set the line thicknesses in SBMLDiagrams with **set_line_thickness()**. Finally, the user can see these changes using **draw(outputFile = 'outputFileName.pdf')**
2. Setting Colors
    a. The user can use **loadIntoSBML()** to once again load their tellurium model into SBMLDiagrams. The user can then get the color gradient using **get_color_gradient()** which uses **get_fluxes()** to determine color scaling based on the fluxes of each reaction. The user can then set the colors in SBMLDiagrams with **set_colors()** and then view the final output using **draw(outputFile = 'outputFileName.pdf')**.
3. Although the respective color and thickness functions work separately, users can still combine both attributes to create a diagram that visualizes flux with both color and line thickness.

Project Timeline

1. Work on functional and component specifications, decide which components are necessary for project to work and how to structure them. Decide if using sbmlnetowrk or SBMLDiagrams. Familiarize and write a tester script with both that accomplishes what SBVisualizer aims to do.
2. Start working on Visualizer class code and approach to determine how to scale colors and thicknesses. Milestone: first draft of code
3. Write a script to test this code and functions to get the flux and print out line thickness values.

4. Write tests for all the functions that the user uses. Milestone: finish tests and make sure they all work.
5. Write script to test the SBVisualizer package with various tellurium models and see the diagrams it outputs. Milestone: makes a reasonable diagram with correct colors and thicknesses
6. Modify the code and approach to determining thickness and color scheme as needed.
7. Finalize documentation, README file, write __init__.py and setup.py, and upload package onto PyPi.

Future Work

I plan to expand this Visualization tool to complex networks. One issue with SBMLDiagrams is that the line thickness is scaled so much that the border of the arrow head becomes too thick and hides the color underneath. SBMLDiagrams is not supported anymore, so there is no way to change this. However, I had issues sbmlnetwork and seeing the color changes in VSCode, and due to the limited time I had, I couldn't properly figure out how to configure Spyder. My next steps are to use sbmlnetwork with Spyder to get a better visualization of the diagram.