ssbio: A Python Framework for Structural Systems Biology

Nathan Mih[a], Elizabeth Brunk[b], Ke Chen[b], Edward Catoiu[b], Anand Sastry[b], Erol Kavvas[b], Jonathan M. Monk[b], Zhen Zhang[b], Bernhard O. Palsson[b]

* Correspondence should be addressed to: B.O.P. (palsson@ucsd.edu)

[a] Bioinformatics and Systems Biology Graduate Program, University of California, San Diego, CA 92093

[b] Department of Bioengineering, University of California, San Diego, CA 92093

Supplementary Information

Attribute access

In the context of a COBRApy `Model` loaded with *ssbio* (`my_model` in the example below), the protein and its main attributes can be accessed simply by:

```
from ssbio.pipeline.gempro import GEMPRO
my_model = GEMPRO(gem_name='si_demo', gem_file_path='/path/to/gem.xml',
                  gem_file_type='sbml')
my_gene_id = 'geneA'
my_protein = my_model.genes.get_by_id(my_gene_id).protein
my_protein.sequences  # Contains a list of stored amino acid sequences
my_protein.structures  # Contains a list of stored 3D structures
my_protein.representative_sequence  # Single representative amino acid sequence
my_protein.representative_structure  # Single representative 3D structure
```

Package organization

ssbio is organized into the following submodules for defined purposes:

1. **`ssbio.databases`**, modules that heavily depend on the Bioservices package (Cokelaer et al. 2013) and custom code to enable pulling information from web services such as UniProt, KEGG, and the PDB, and to directly convert that information into sequence and structure objects to load into a protein.
2. **`ssbio.core`**, modules which represent objects that make up some of the contents of a cell systems model (genes, proteins and protein complexes).
3. **`ssbio.protein.sequence`**, modules which allow a user to execute and parse sequence-based utilities such as sequence alignment algorithms or structural feature predictors.
4. **`ssbio.protein.structure`**, modules that mirror the sequence module but instead work with structural information to calculate properties, and also to streamline the generation of homology models as well as to prepare structures for molecular modeling tools such as docking or molecular dynamics.
5. **`ssbio.pipeline.gempro`**, a pipeline that simplifies the execution of these tools per protein while placing them into the context of a genome-scale model.

6. **`ssbio.viz`**, modules which focus on the 3D visualization of protein structures and network representations of systems models.

A table of currently available functionalities for protein sequences and structures can be found in Supplementary Tables S1 and S2.

Cached files

Files such as sequences, structures, alignment files, and property calculation outputs can optionally be cached on a user's disk to minimize calls to web services, limit recalculations, and provide direct inputs to common sequence and structure algorithms which often require local copies of the data. For a GEM-PRO project, files are organized in the following fashion once a root directory and project name are set:

```
<ROOT_DIR>
└── <PROJECT_NAME>
      ├── data  # General directory for pipeline outputs
      ├── model  # SBML and GEM-PRO models are stored in this directory
      └── genes  # Per gene information
            └── <gene_id1>  # Specific gene directory
                  └── <protein_id1>  # Protein directory
                        ├── sequences  # Protein sequence files, alignments, etc.
                        └── structures  # Protein structure files, calculations, etc.
```

Supplementary Table S1. A summary of currently available functionalities for a **protein sequence** contained in *ssbio*. External software or web servers are listed if the functionality is dependent on it. If there are software listed as an alternate, then that means that there exists either another third-party program or a Python-based method that carries out the same function. For example, the Biopython pairwise2 function carries out a pairwise sequence alignment, but can also be run using the EMBOSS needle package. This table can also be found in the documentation at: http://ssbio.readthedocs.io/en/latest/sequence.html

Analysis type	**Function**	**Description**	**Internal ssbio class or function**	**External software**	**Web server**	**Alternate external software**
Sequence-based predictions	Secondary structure and solvent accessibilities	Predictions of secondary structure and relative solvent accessibilities per residue	scratch module	SCRATCH (Cheng et al. 2005)		
	Thermostability	Free energy of unfolding (ΔG), adapted from Oobatake (Oobatake & Ooi 1993) and Dill (Dill et al. 2011)	thermostability module			
	Transmembrane domains	Prediction of transmembrane domains from sequence	tmhmm module	TMHMM (Krogh et al. 2001)		
	Aggregation propensity	Consensus method to predict the aggregation propensity of proteins, specifically the number of aggregation-prone segments on an unfolded protein sequence	aggregation propensity module		AMYLPRED 2 (Tsolis et al. 2013)	
Sequence-based calculations	Various sequence properties	Basic properties of the sequence, such as percent of polar, non-polar, hydrophobic or hydrophilic residues.	Biopython ProteinAnalysis (Cock et al. 2009) sequence residues module			EMBOSS pepstats (Rice et al. 2000)
	Sequence alignment	Basic functions to run pairwise or multiple sequence alignments	Biopython pairwise2 (Cock et al. 2009) alignment module			EMBOSS needle (Rice et al. 2000)

Supplementary Table S2. A summary of currently available functionalities for a **protein structure** contained in *ssbio*. Residue-level properties from structures are linked to the correct residue numbering schemes in mapped sequences, or vice-versa. As in Supplementary Table S1, external software, web servers and alternates are provided in the rightmost columns. This table can also be found in the documentation at: http://ssbio.readthedocs.io/en/latest/structure.html

Analysis type	**Function**	**Description**	**Internal ssbio class or function**	**External software**	**Web server**	**Alternate external software**

Sequence-based prediction	Homology modeling	Preparation scripts and parsers for executing homology modeling algorithms	`itasserprep module`
			
			`itasserprop module`
				I-TASSER (Roy et al. 2010)		

Structure-based prediction	Transmembrane orientation	Prediction of transmembrane domains and orientation in a membrane	`opm module`		OPM (Lomize et al. 2012)	
	Kinetic folding rate	Prediction of protein folding rates from amino acid sequence	`kinetic folding rate module`		FOLD-RATE (Gromiha et al. 2006)	

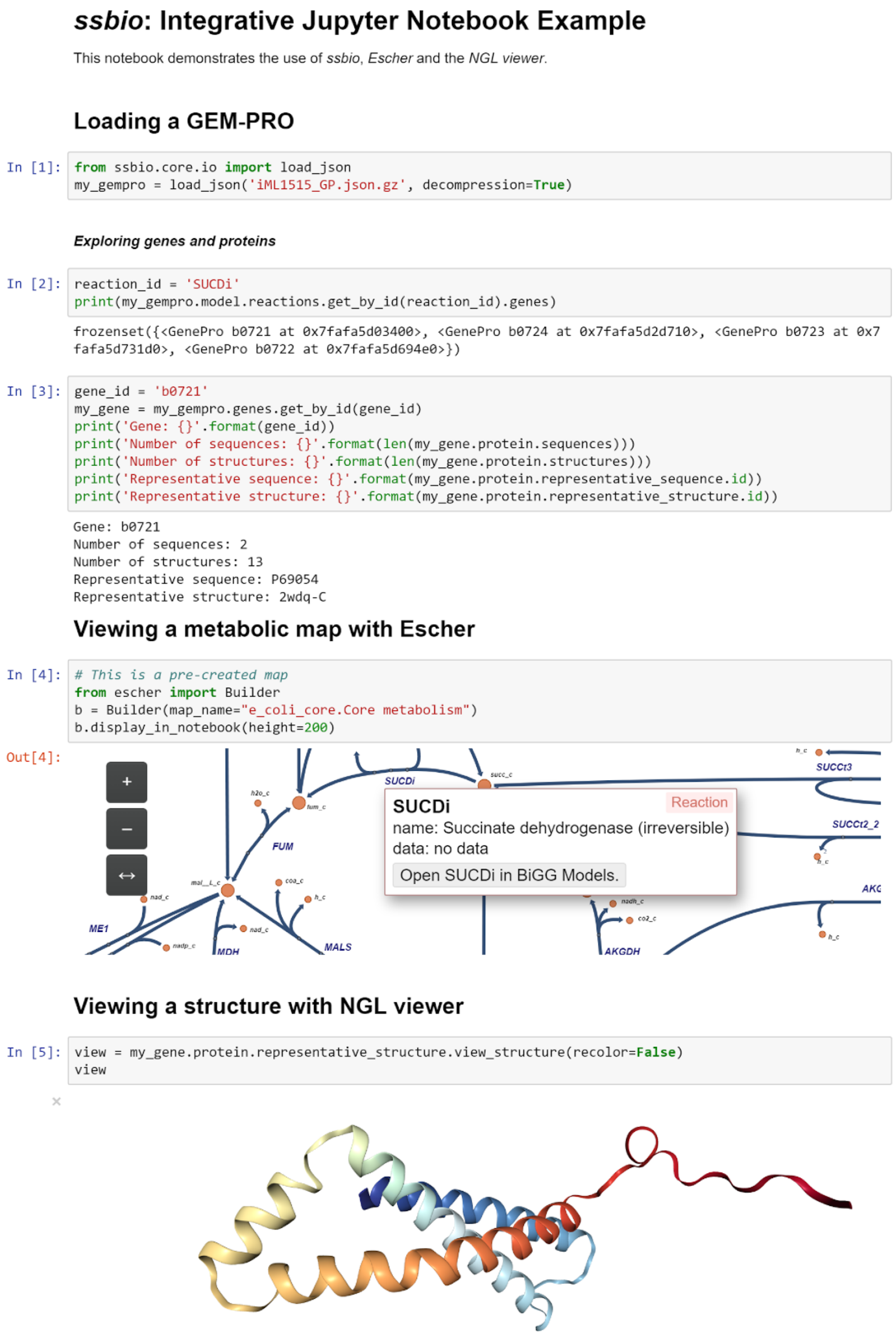
Structure-based calculation	Secondary structure	Calculations of secondary structure	Biopython DSSP (Hamelryck & Manderick 2003)
			
			`dssp module`
			
			`stride module`
				DSSP (Kabsch & Sander 1983),		STRIDE (Frishman & Argos 1995)

	Solvent accessibilities	Calculations of per-residue absolute and relative solvent accessibilities	Biopython DSSP (Hamelryck & Manderick 2003)
			
			`dssp module`
			
			`freesasa module`
				DSSP (Kabsch & Sander 1983),		FreeSASA (Mitternacht 2016)

	Residue depths	Calculations of residue depths	Biopython ResidueDepth (Hamelryck & Manderick 2003) `msms module`	MSMS (Sanner et al. 1996)		
	Structural similarity	Pairwise calculations of 3D structural similarity	`fatcat module`	FATCAT (Ye & Godzik 2003)		
	Quality	Custom functions to allow ranking of structures by percent identity to a defined sequence, structure resolution, and other structure quality metrics	`set representative structure function`			
	Various structure properties	Basic properties of the structure, such as distance measurements between residues or number of disulfide bridges	`structure residues module`	Biopython Struct (Hamelryck & Manderick 2003)		

Structure-based function	Structure cleaning, mutating	Custom functions to allow for the preparation of structure files for molecular modeling, with options to remove hydrogens/waters/heteroatoms, select specific chains, or mutate specific residues.	Biopython Select (Hamelryck & Manderick 2003)
			
			`cleanpdb module`
			
			`mutatepdb module`
						AmberTools (Duke et al. 2016)

Supplementary Figure S1. A screenshot from a Jupyter notebook tutorial demonstrating the use *ssbio* to load a GEM-PRO model, Escher (King et al. 2015) to build or view existing metabolic maps, and the NGL viewer (Rose & Hildebrand 2015) to view protein structures.



ssbio: Integrative Jupyter Notebook Example

This notebook demonstrates the use of *ssbio*, *Escher* and the *NGL viewer*.

Loading a GEM-PRO

```
In [1]: from ssbio.core.io import load_json
        my_gempro = load_json('iML1515_GP.json.gz', decompression=True)
```

Exploring genes and proteins

```
In [2]: reaction_id = 'SUCDi'
        print(my_gempro.model.reactions.get_by_id(reaction_id).genes)

frozenset({<GenePro b0721 at 0x7fafa5d03400>, <GenePro b0724 at 0x7fafa5d2d710>, <GenePro b0723 at 0x7
fafa5d731d0>, <GenePro b0722 at 0x7fafa5d694e0>})
```

```
In [3]: gene_id = 'b0721'
        my_gene = my_gempro.genes.get_by_id(gene_id)
        print('Gene: {}'.format(gene_id))
        print('Number of sequences: {}'.format(len(my_gene.protein.sequences)))
        print('Number of structures: {}'.format(len(my_gene.protein.structures)))
        print('Representative sequence: {}'.format(my_gene.protein.representative_sequence.id))
        print('Representative structure: {}'.format(my_gene.protein.representative_structure.id))

Gene: b0721
Number of sequences: 2
Number of structures: 13
Representative sequence: P69054
Representative structure: 2wdq-C
```

Viewing a metabolic map with Escher

```
In [4]: # This is a pre-created map
        from escher import Builder
        b = Builder(map_name="e_coli_core.Core metabolism")
        b.display_in_notebook(height=200)

Out[4]:
```

SUCDi Reaction
name: Succinate dehydrogenase (irreversible)
data: no data
Open SUCDi in BiGG Models.

Viewing a structure with NGL viewer

```
In [5]: view = my_gene.protein.representative_structure.view_structure(recolor=False)
        view
```

References

Cheng, J., Randall, A. Z., Sweredoski, M. J., & Baldi, P. (2005). ‘SCRATCH: a protein structure and structural feature prediction server’, *Nucleic acids research*, 33/Web Server issue: W72–6. DOI: 10.1093/nar/gki396

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., et al. (2009). ‘Biopython: freely available Python tools for computational molecular biology and bioinformatics’, *Bioinformatics* , 25/11: 1422–3. Oxford Univ Press.

Cokelaer, T., Pultz, D., Harder, L. M., Serra-Musach, J., & Saez-Rodriguez, J. (2013). ‘BioServices: a common Python package to access biological Web Services programmatically.’, *Bioinformatics* , 29/24: 3241–2. DOI: 10.1093/bioinformatics/btt547

Dill, K. A., Ghosh, K., & Schmit, J. D. (2011). ‘Physical limits of cells and proteomes’, *Proceedings of the National Academy of Sciences of the United States of America*, 108/44: 17876–82. DOI: 10.1073/pnas.1114477108

Duke, R. E., Giese, T. J., Gohlke, H., Goetz, A. W., Homeyer, N., Izadi, S., Janowski, P., et al. (2016). ‘AmberTools 16’. University of California, San Francisco.

Frishman, D., & Argos, P. (1995). ‘Knowledge-based protein secondary structure assignment’, *Proteins*, 23/4: 566–79. DOI: 10.1002/prot.340230412

Gromiha, M. M., Thangakani, A. M., & Selvaraj, S. (2006). ‘FOLD-RATE: prediction of protein folding rates from amino acid sequence’, *Nucleic acids research*, 34/Web Server issue: W70–4. DOI: 10.1093/nar/gkl043

Hamelryck, T., & Manderick, B. (2003). ‘PDB file parser and structure class implemented in Python’, *Bioinformatics* , 19/17: 2308–10.

Kabsch, W., & Sander, C. (1983). ‘DSSP: definition of secondary structure of proteins given a set of 3D coordinates’, *Biopolymers*, 22: 2577–637.

King, Z. A., Dräger, A., Ebrahim, A., Sonnenschein, N., Lewis, N. E., & Palsson, B. O. (2015). ‘Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways’, *PLoS computational biology*, 11/8: e1004321. DOI: 10.1371/journal.pcbi.1004321

Krogh, A., Larsson, B., von Heijne, G., & Sonnhammer, E. L. (2001). ‘Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes’, *Journal of molecular biology*, 305/3: 567–80. DOI: 10.1006/jmbi.2000.4315

Lomize, M. A., Pogozheva, I. D., Joo, H., Mosberg, H. I., & Lomize, A. L. (2012). ‘OPM database and PPM web server: resources for positioning of proteins in membranes’, *Nucleic acids research*, 40/Database issue: D370–6. DOI: 10.1093/nar/gkr703

Mitternacht, S. (2016). ‘FreeSASA: An open source C library for solvent accessible surface area calculations’, *F1000Research*, 5: 189. DOI: 10.12688/f1000research.7931.1

Oobatake, M., & Ooi, T. (1993). ‘Hydration and heat stability effects on protein unfolding’, *Progress in biophysics and molecular biology*, 59/3: 237–84.

Rice, P., Longden, I., & Bleasby, A. (2000). ‘EMBOSS: the European Molecular Biology Open Software Suite’, *Trends in genetics: TIG*, 16/6: 276–7. DOI: 10.1016/S0168-9525(00)02024-2

Rose, A. S., & Hildebrand, P. W. (2015). ‘NGL Viewer: a web application for molecular visualization’, *Nucleic acids research*, 43/W1: W576–9. DOI: 10.1093/nar/gkv402

Roy, A., Kucukural, A., & Zhang, Y. (2010). ‘I-TASSER: a unified platform for automated protein structure and function prediction’, *Nature protocols*, 5/4: 725–38. DOI: 10.1038/nprot.2010.5

Sanner, M. F., Olson, A. J., & Spehner, J.-C. (1996). ‘Reduced surface: an efficient way to compute molecular surfaces’, *Biopolymers*, 38/3: 305–20. Wiley Online Library.

Tsolis, A. C., Papandreou, N. C., Iconomidou, V. A., & Hamodrakas, S. J. (2013). ‘A consensus method for the prediction of “aggregation-prone” peptides in globular proteins’, *PloS one*, 8/1: e54175. DOI: 10.1371/journal.pone.0054175

Ye, Y., & Godzik, A. (2003). ‘Flexible structure alignment by chaining aligned fragment pairs allowing twists’, *Bioinformatics* , 19 Suppl 2: ii246–55.