datanator documentation

Release 0.2

Karr Lab

May 25, 2020

Contents

1 Contents **3**

1.1 Overview . . . 3

1.1.1 Motivation . . . 3

1.1.2 Methodology . . . 4

1.1.3 Supported data types and data sources . . . 4

1.1.4 Advantages of `datanator` versus manual data aggregation . . . 5

1.2 Installation . . . 5

1.2.1 Install dependencies . . . 5

1.2.2 Install `datanator` . . . 6

1.2.3 Run `datanator` . . . 6

1.3 Tutorial - Getting Familiar With datanator's Command Line Interface . . . 6

1.3.1 Generate Template Doc . . . 6

1.3.2 Collecting Kinetic Data . . . 7

1.3.3 Entering Reactions Into datanator . . . 7

1.3.4 Set Maximum Proximity Limit . . . 9

1.4 About . . . 9

1.4.1 License . . . 9

1.4.2 Development team . . . 10

1.4.3 Acknowledgements . . . 10

1.4.4 Questions and comments . . . 10

1.5 References . . . 10

`datanator` is a software tool for finding experimental data for building and calibrating dynamical models of cellular biochemistry such as metabolite, RNA, and protein abundances; protein complex compositions; transcription factor binding motifs; and kinetic parameters. `datanator` is particularly useful for building large models, such as whole-cell models, that require large amounts of data to constrain large numbers of parameters. `datanator` was motivated by the need for large amounts of data to constrain whole-cell models and the fact that this data is hard to utilize because it is scattered across numerous siloed repositories.

`datanator` currently supports the following data types and data sources:

- Metabolite concentrations: ECMDB and YMBD
- RNA abundance: ArrayExpress
- Protein abundance: PaxDb
- Protein complex composition: CORUM
- Transcription factor binding motifs: JASPAR
- Reaction kinetics: SABIO-RK
- Taxonomy: NCBI Taxonomy

`datanator` (1) downloads these repositories; (2) normalizes their data to a common ontology and units; (3) stores their data to a local SQLite database; and (4) provides a Python API for (a) finding relevant data to model a specific organism and environmental condition from similar species, reactions, genotypes (taxon, variant), and environments (temperature, pH, media), and (b) reducing multiple relevant observations to a single consensus recommended parameter value, and (c) exporting these consensus recommendations and their provenance to an Excel workbook. To make `datanator` easier to use, we plan to develop user-friendly command line and web-based interfaces for finding data for SBML-encoded models.

`datanator` is under active development and is not yet ready for end users. Please check back soon for updates.

This website contains detailed documentation of the `datanator` source code. Going forward, this website will also contain detailed instructions and tutorials on how to use `datanator`.

CHAPTER 1

Contents

1.1 Overview

`datanator` is a software tool for finding experimental data for building and calibrating dynamical models of cellular biochemistry such as metabolite, RNA, and protein abundances; protein complex compositions; transcription factor binding motifs; and kinetic parameters. `datanator` is particularly useful for building large models such as whole-cell models that require large amounts of data to constrain large numbers of parameters.

1.1.1 Motivation

Large models such as whole-cell models are needed to help researchers, bioengineers, and physicians predict how genotype and the environment determine phenotype. In particular, large models that represent the function of each individual gene are needed to help bioengineers rationally design microorganisms to perform specific functions such as producing drugs and neutralizing pathogens and to help physicians interpret personal genomes and design personalized therapies tailored to each patient's unique genome.

Despite their potential, large models are hard to build, in part, because it is hard to aggregate the large amount of data needed to build large models because this data is scattered across numerous siloed repositories. Without data aggregation tools such as `datanator`, this data must be manually aggregated:

1. Modelers must identify appropriate data sources for their model.
2. Modelers must identify the relevant subset of this data for their model.
3. Modelers must merge inconsistently annotated data from multiple sources.
4. Modelers must reduce multiple experimental observations to individual consensus parameter value recommendations.
5. Modelers must record the provenance of each recommendation so their model is comprehensible and reproducible.

This requires extensive time and effort, is unscalable, introduces substantial selection bias into the data underlying models, and is irreproducible.

To address these problems, we have developed `datanator` to systemize and accelerate the aggregation of data for biomodels.

1.1.2 Methodology

1. `datanator` downloads data from the data repositories listed below.
2. `datanator` parses this data, normalizes this data to a common schema and common identifiers, and converts this data to common units.
3. `datanator` stores the normalized data and its provenance to a local SQLite database.
4. `datanator` helps researchers find relevant data to model a specific organism and environmental condition from similar species, reactions, genotypes, and environments according to the following filters:
    - Chemical similarity: `datanator` helps researchers identify data observed for (a) molecularly-similar species as determined by the Tanimoto similarity of their molecular fingerprints and (b) chemically-similar reactions that involve similar reaction centers and reaction mechanisms.
    - Taxonomic similarity: `datanator` helps researchers identify data observed for taxonomically-close taxa and similar genetic variants.
    - Environmental similarity: `datanator` helps researchers identify data observed for similar temperatures, pHs, and growth media.
5. `datanator` reduces multiple relevant observations to a single consensus recommended parameter value.
6. `datanator` records the provenance of the data underlying each consensus recommendation.
7. `datanator` exports the consensus recommendations and their provenance to Excel workbooks.

Currently, `datanator` provides researchers a Python API to programmatically aggregate data for models. To make `datanator` easier to use, we plan to develop user-friendly command line and web-based interfaces which will help users find data for SBML-encoded models, review this data, and generate consensus recommendations for parameter values.

1.1.3 Supported data types and data sources

`datanator` currently supports the following data type and data sources. We are also actively integrating additional data types and data sources into `datanator`.

- Databases
    - Metabolite concentrations: ECMDB and YMBD
    - RNA abundance: ArrayExpress
    - Protein abundance: PaxDb
    - Protein complex composition: CORUM
    - Transcription factor binding motifs: JASPAR
    - Reaction kinetics: SABIO-RK
    - Taxonomy: NCBI Taxonomy
- Prediction tools
    - Metabolite properties (charge, pK_a, protonation): Open Babel
    - EC number: E-zyme

1.1.4 Advantages of `datanator` versus manual data aggregation

- `datanator` accelerates data aggregation by merging data from multiple repositories into a single location.
- `datanator` reduces selection bias in the data used to build models by systemizing how researchers find data for models.
- `datanator` helps researchers find more data for models by helping researchers find data from chemically-similar species and reactions, taxonomically-similar organisms, and similar environmental conditions based on their temperature, pH, and media.
- `datanator` increases the comprehensibility and reproducibility of models by automatically tracking the provenance of each recommended parameter value.

1.2 Installation

The following instructions describe how to install `datanator` onto Ubuntu Linux 16.04. Datanator only supports Python 3.

1.2.1 Install dependencies

First, please install the following dependencies:

- Git
- Open Babel
- Python 3
- Pip

The following shell commands can be used to install these dependencies onto Ubuntu Linux 16.04:

```
apt-get install \
    git \
    libcairo2-dev \
    libeigen3-dev \
    libxml2-dev \
    python \
    python-pip \
    swig \
    zlib1g-dev

cd /tmp
wget https://sourceforge.net/projects/openbabel/files/openbabel/2.4.1/openbabel-2.4.1.
↪tar.gz/download -O /tmp/openbabel-2.4.1.tar.gz
tar -xvvf /tmp/openbabel-2.4.1.tar.gz
cd openbabel-2.4.1
mkdir build
cd build
cmake ..
make
make install
ldconfig
```

1.2.2 Install datanator

Second, please run the following shell commands to clone and install datanator from GitHub:

```
git clone git@github.com:KarrLab/datanator.git
python3 -m pip install -e datanator
```

Because datanator is under active development, we recommend regularly pulling the latest revision of datanator from GitHub.

1.2.3 Run datanator

The API for datanator can be run with a test and production server.

In order to run the test server, run the following command:

```
python3 manage.py runserver
```

NOTE: You will need to have the correct configuration in the datanator/__init__.py file. Configurations can be found in datanator/config.py include:

- LocalDevelopmentConfig - Local server for database
- CircleTestingConfig - CircleCI server for database
- BuildConfig - Docker Compose/UCONN HPC server for database
- ProductionConfig - AWS RDS server for database (PRIVATE)

In order to run the production server, run the following command:

```
gunicorn -w 4 -b localhost:5000 --timeout 120 manage:app
```

This command will create a gunicorn production server with 4 workers at the localhost:5000 address with a timeout of 2 min

Contact Saahith for any questions regarding installation and running the server

1.3 Tutorial - Getting Familiar With datanator's Command Line Interface

1.3.1 Generate Template Doc

datanator can be used run directly from the command line.

Let's do an example

Create a new directory

In that directory run the following command:

```
$ datanator generate-template
```

This will output a template document to your directory. This template is an example of an excel sheet that can be entered into datanator

1.3.2 Collecting Kinetic Data

In order to collect kinetic information on the reaction in the template, we will use "find-kinetics"

There are three required arguments necesssary to run the data collection method "find-kinetics"

1. A path to an excel sheet with the reactions you would like to search (we will discuss the necessary formatting later)
2. A name for the results file (the results are saved as an excel file, so the name should end in '.xlsx')
3. The name of the species you are searching for

The first argument is a path to an excel sheet, we will use the template document we just generated labelled InputTemplate.xlsx

The second argument is a name for the output file. We will use *Results.xlsx*, but you can name it anything.

The third argument is the name of the species we are searching for. We will use 'homo sapiens'

Run "get-kinetics":

```
$ datanator get-kinetics InputTemplate.xlsx Results.xlsx 'homo sapiens'
```

$ datanator get-taxonomic-lineage 'Escherichia coli'

1.3.3 Entering Reactions Into datanator

Open up the template document InputTemplate.xlsx

This document has two worksheets: Reactions, and Metabolites.

Reactions Worksheet

Reaction ID	Stoichimetry
A Reacion Name #1 (ATP + UMP <==> UDP + ADP)	ATP + UMP <==> UDP + ADP
A Reaction Name #2 (GMP + ATP <==> ADP + GDP)	GMP + ATP ==> ADP + GDP

Look at the Reactions worksheet. There are two columns. The first is an identifier for the reaction. You can name this whatever you like as long as it is unique. A good suggestion is just to use the reaction string (in the template document, we did not follow this practice in order to illustrate that the name can be anything). The second column is a the stoichiometric string.

Each metabolite used in the stoichiometric string needs to be structurally defined in the "Metabolites" worksheet. Open up the "Metabolites" worksheet

Compound ID	Structure
ATP	NC1=C2N=CN(C3OC(COP([O-])(=O)OP([O-])(=O)OP([O-])([O-])=O)C(O)C3O)C2=NC=N1
UMP	OC1C(O)C(OC1COP([O-])([O-])=O)N1C=CC(=O)NC1=O
UDP	OC1C(O)C(OC1COP([O-])(=O)OP([O-])([O-])=O)N1C=CC(=O)NC1=O
AMP	NC1=C2N=CN(C3OC(COP([O-])([O-])=O)C(O)C3O)C2=NC=N1
GMP	NC1=NC2=C(N=CN2C2OC(COP([O-])([O-])=O)C(O)C2O)C(=O)N1
GDP	NC1=NC2=C(N=CN2C2OC(COP([O-])(=O)OP([O-])([O-])=O)C(O)C2O)C(=O)N1

The compound ID corresponds to the ID of the compound in the stoichiometric string. The structure is either an Inchi or a SMILES string (SMILES in this example).

Let's try adding a reaction:

Lets say we wanted to add the reaction:

```
Adenosine 3',5'-bisphosphate + H2O ==> phosphate + AMP
```

The first step is to give each metabolite in the stoichiometric string and ID without spaces. So let's change it to:

```
A-3-5-bisphosphate + H2O ==> phosphate + AMP
```

The second step is to add this reaction string to the second column in "Reactions" worksheet. In the first column, give the reaction some distinct name.

Reaction ID	Stoichimetry
A Reacion Name #1 (ATP + UMP <==> UDP + ADP)	ATP + UMP <==> UDP + ADP
A Reaction Name #2 (GMP + ATP <==> ADP + GDP)	GMP + ATP ==> ADP + GDP
A distinct name of your choosing	A-3-5-bisphosphate + H2O ==> phosphate + AMP

The third step is to structurally define each metabolite in the reaction string. We already defined structurally defined AMP (we used it in the previous reactions), so we will have to structurally define A-3-5-bisphosphate, H2O, and phosphate.

The structural information is here:

```
A-3-5-bisphosphate - NC1=C2N=CN(C3OC(COP([O-])([O-])=O)C(OP([O-])([O-])=O)C3O)C2=NC=N1

H2O - O

phosphate - OP([O-])([O-])=O
```

Now we need to add this information to the "Metabolites" worksheet.

Open up the "Metabolites" worksheet. Add the name of the compound used in the stoichiometric string (ex: phosphate) to the first column, and add the structure in the second.

Compound ID	Structure
ATP	NC1=C2N=CN(C3OC(COP([O-])(=O)OP([O-])(=O)OP([O-])([O-])=O)C(O)C3O)C2=NC=N1
UMP	OC1C(O)C(OC1COP([O-])([O-])=O)N1C=CC(=O)NC1=O
UDP	OC1C(O)C(OC1COP([O-])(=O)OP([O-])([O-])=O)N1C=CC(=O)NC1=O
AMP	NC1=C2N=CN(C3OC(COP([O-])([O-])=O)C(O)C3O)C2=NC=N1
GMP	NC1=NC2=C(N=CN2C2OC(COP([O-])([O-])=O)C(O)C2O)C(=O)N1
GDP	NC1=NC2=C(N=CN2C2OC(COP([O-])(=O)OP([O-])([O-])=O)C(O)C2O)C(=O)N1
A-3-5-bisphosphate	NC1=C2N=CN(C3OC(COP([O-])([O-])=O)C(OP([O-])([O-])=O)C3O)C2=NC=N1
H2O	O
phosphate	OP([O-])([O-])=O

Once again, run "get-kinetics":

```
$ datanator get-kinetics InputTemplate.xlsx Results.xlsx 'Escherichia coli'
```

You should see results for the new reaction you inputted.

1.3.4 Set Maximum Proximity Limit

The ideal kinetic data is information about the reaction being studied, collected from an expirement done in the species you are studying. However, often you will have to rely on kinetic data from different species. At a certain taxonomic distance, you might decide it's better to collect data from similar reactions taken from expirements in more closely related organisms.

There are two dimensions of granularity - reaction variation and species variation - and the user can decide which data is preferred.

The user can do this by setting the "proximLimit"

Let's try an example:

First, you want to get taxonomic infomation about the organism you are stuyding. Run:

```
$ datanator get-taxonomic-lineage 'Escherichia coli'
```

You should see:

```
1: Escherichia coli
2: Escherichia
3: Enterobacteriaceae
4: Enterobacterales
5: Gammaproteobacteria
6: Proteobacteria
7: Bacteria
8: cellular organisms
9: root
```

This is the number of nodes as you start from your organism, and climb up to the top of the taxonomic tree. Each number corresponds to a node.

So let's say you are studying Escherichia coli. Maybe you think that anything outside the phylum protobacteria is too distantly related to be useful. In that case, you will want to run the "get-kinetics" argument, with an optional argument –proximit-limit. The number given after –proxim-limit is the highest node that will be considered useful. Since we have chosen Protobacteria, that number is 6

So, run:

```
datanator get-kinetics InputTemplate.xlsx Results.xlsx 'Escherichia coli' --proxim-
↪limit 6
```

1.4 About

1.4.1 License

The software is released under the MIT license:

```
The MIT License (MIT)

Copyright (c) 2017-2018 Karr Lab

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
```

(continues on next page)

(continued from previous page)

1.4.2 Development team

This package was developed by the following researchers in the Karr Lab at the Icahn School of Medicine at Mount Sinai in New York, USA:

- Yosef Roth
- Saahith Pochiraju
- Balazs Szigeti
- Jonathan Karr

1.4.3 Acknowledgements

This work was supported by a National Institute of Health MIRA award [grant number 1 R35 GM 119771-01]; a National Science Foundation INSPIRE award [grant number 1649014]; and the National Science Foundation / ERASynBio [grant numbers 1548123, 335672].

1.4.4 Questions and comments

Please contact the Karr Lab with any questions or comments.

1.5 References