cytominer_database Documentation

Release 0.0

Claire McQuin,Allen Goodman,Shantanu Singh

Dec 19, 2019

Table of Contents:

cytominer-database provides command-line tools for organizing measurements extracted from images.

Software tools such as CellProfiler can extract hundreds of measurements from millions of cells in a typical high-throughput imaging experiment. The measurements are stored across thousands of CSV files.

cytominer-database helps you organize these data into a single database backend, such as SQLite.

CHAPTER 1

Why cytominer-database?

While tools like CellProfiler can store measurements directly in databases, it is usually infeasible to create a centralized database in which to store these measurements. A more scalable approach is to create a set of CSVs per "batch" of images, and then later merge these CSVs.

cytominer-database ingest reads these CSVs, checks for errors, then ingests them into a database backend, including SQLite, MySQL, PostgresSQL, and several other backends supported by odo.

```
cytominer-database ingest source_directory sqlite:///backend.sqlite -c ingest_config.
↪ini
```

will ingest the CSV files nested under source_directory into a SQLite backend

1.1 Configuration

```
[filenames]
image = image.csv
object = object.csv
experiment = Experiment.csv
```

1.2 Reference

1.2.1 ingest

A mechanism to ingest CSV files into a database.

In morphological profiling experiments, a CellProfiler pipeline is often run in parallel across multiple images and produces a set of CSV files. For example, imaging a 384-well plate, with 9 sites per well, produces 384 * 9 images; a CellProfiler process may be run on each image, resulting in a 384*9 output directories (each directory typically contains one CSV file per compartment (e.g. Cells.csv, Cytoplasm.csv, Nuclei.csv) and one CSV file for per-image measurements (e.g. Image.csv).

`cytominer_database.ingest.seed` can be used to read all these CSV files into a database backend. SQLite is the recommended engine, but ingest will likely also work with PostgreSQL and MySQL.

`cytominer_database.ingest.seed` assumes a directory structure like shown below:

plate_a/
    set_1/
        file_1.csv
        file_2.csv
        ...
        file_n.csv
    set_2/
        file_1.csv
        file_2.csv
        ...
        file_n.csv
    ...
    set_m/
        file_1.csv
        file_2.csv
        ...
        file_n.csv

Example:

```
import cytominer_database.ingest

cytominer_database.ingest.seed(source, target, config)
```

`cytominer_database.ingest.`**`checksum`**(*pathname*, *buffer_size=65536*)
Generate a 32-bit unique identifier for a file.

Parameters

- **`pathname`** – input file
- **`buffer_size`** – buffer size

`cytominer_database.ingest.`**`into`**(*input*, *output*, *name*, *identifier*, *skip_table_prefix=False*)
Ingest a CSV file into a table in a database.

Parameters

- **`input`** – Input CSV file.
- **`output`** – Connection string for the database.
- **`name`** – Table in database into which the CSV file will be ingested
- **`identifier`** – Unique identifier for `input`.
- **`skip_table_prefix`** – True if the prefix of the table name should be excluded from the names of columns.

`cytominer_database.ingest.`**`seed`**(*source*, *target*, *config_file*, *skip_image_prefix=True*)
Read CSV files into a database backend.

Parameters

- **config_file** – Configuration file.
- **source** – Directory containing subdirectories that contain CSV files.
- **target** – Connection string for the database.
- **skip_image_prefix** – True if the prefix of image table name should be excluded from the names of columns from per image table

1.2.2 munge

cytominer_database.munge.**munge**(*config_file*, *source*, *target=None*)

Searches `source` for directories containing a CSV file corresponding to per-object measurements, then splits the CSV file into one CSV file per compartment.

For instance, the CSV file may comprise of measurements combined across Cells, Cytoplasm, and Nuclei. `munge` will split this CSV file into 3 CSV files: Cells.csv, Cytoplasm.csv, and Nuclei.csv.

Parameters

- **config_file** – Configuration file.
- **source** – Directory containing subdirectories that contain an object CSV file.
- **target** – Output directory. If not specified, then it is same as `source`.

Returns list of subdirectories that have an object CSV file.

Example:

```
import cytominer_database.munge

cytominer_database.munge.munge(source, target, config)
```

1.2.3 utils

cytominer_database.utils.**collect_csvs**(*config*, *directory*)

Collect CSV files from a directory.

This function collects CSV files in a directory, excluding those that have been specified in the configuration file. This enables collecting only those CSV files that correspond to cellular compartments. e.g. Cells.csv, Cytoplasm.csv, Nuclei.csv. CSV files corresponding to experiment, image, or object will be excluded.

Parameters

- **config** – configuration file.
- **directory** – directory containing the CSV files.

Returns a list of CSV files.

cytominer_database.utils.**find_directories**(*directory*)

List subdirectories.

Parameters **directory** – directory

Returns list of subdirectories of `directory`

cytominer_database.utils.**read_config**(*filename*)
Read a configuration file. A default config file is read first, and the values are overriden by those in the specified configuration file.

Parameters **filename** – configuration filename

Returns a configuration object

cytominer_database.utils.**validate_csv**(*csvfile*)
Validate a CSV file.

The CSV file typically corresponds to either a measurement made on a compartment, e.g. Cells.csv, or on an image, e.g. Image.csv. The validation performed is generic - it simply checks for malformed CSV files.

This uses csvclean to check for validity of a CSV file.

Parameters **csvfile** – CSV file to validate

Returns True if valid, False otherwise.

cytominer_database.utils.**validate_csv_set**(*config*, *directory*)
Validate a set of CSV files.

This function validates a set of CSV files in a directory. These CSV files correspond to measurements made on different cellular compartments, e.g. Cells.csv, Cytoplasm.csv, Nuclei.csv. An Image.csv file, corresponding to measurements made on the whole image, along with metadata, is also typically present.

Parameters

- **config** – configuration file - this contains the set of CSV files to validate.
- **directory** – directory containing the CSV files.

Returns a tuple where the first element is the list of compartment CSV files, the second is the image CSV file.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index