The Journal of Open Source Software

AIgarMIC: a Python package for automated interpretation of agar dilution minimum inhibitory concentration assays

Alessandro Gerada [1,2], **Nicholas Harper** [1], **Alex Howard** [1,2], **and William Hope** [1,2]

1 Antimicrobial Pharmacodynamics and Therapeutics Group, Department of Pharmacology and Therapeutics, Institute of Systems, Molecular & Integrative Biology, University of Liverpool, United Kingdom **2** Department of Infection and Immunity, Liverpool Clinical Laboratories, Liverpool University Hospitals NHS Foundation Trust, Liverpool, United Kingdom

DOI: 10.21105/joss.06826

Software

- Review
- Repository
- Archive

Editor: Antonia Mey

Reviewers:

- @gchure

Submitted: 12 April 2024
Published: 02 September 2024

Summary

Minimum inhibitory concentration (MIC) assays are used to estimate the susceptibility of a microorganism to an antibiotic. The result is broadly used within microbiology. In clinical settings, it is used to determine whether it is possible to use that same drug to treat a patient's infection. Agar dilution is a reference method for MIC measurement. However, the interpretation of agar dilution plates is time-consuming and prone to intra- and inter-operational errors when read by laboratory personnel. `AIgarMIC` is a Python package for automated interpretation of agar dilution images.

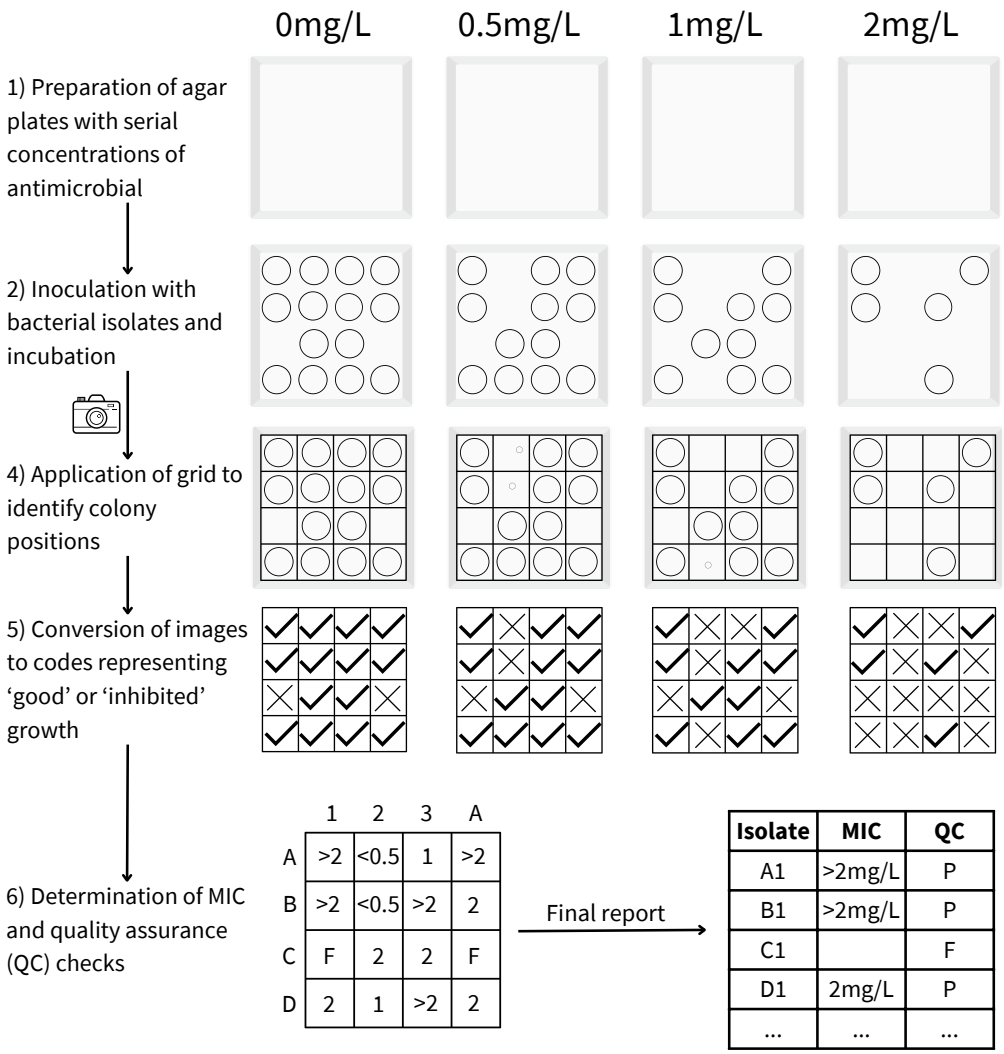

Figure 1: High-level overview of the integration of `AIgarMIC` within the laboratory pathway of minimum inhibitory concentration measurement using agar dilution. `AIgarMIC` performs the interpretative steps of the pathway (from step 5), taking a set of agar plates with a colony-locating grid as an input, and reporting an MIC for each isolate. In this example, 4x4 strains are inoculated onto agar plates, giving a total of 16 strains. F = quality control failed (no growth in positive control plate).

From an input of agar plate images generated through agar dilution (usually consisting of a positive control plate and multiple plates with serial dilutions of antimicrobial concentration), `AIgarMIC` returns an MIC for each microorganism strain in the experiment. Figure 1 provides a high-level overview of how `AIgarMIC` achieves this. Firstly, each agar plate image is split into smaller images for each bacterial strain. Next, using a pre-trained image classification model, the small colony images are converted to a code representing growth level (e.g., good growth, inhibited growth) and stored in a matrix for each plate. Finally, `AIgarMIC` uses the growth matrices from all plates to identify the antimicrobial concentration at which microbial growth is inhibited – the minimum inhibitory concentration. `AIgarMIC` can be imported for use in Python scripts, or can be run through a command-line interface. Users can customise `AIgarMIC` to their workflow with bespoke models, or use the pre-trained models provided. `AIgarMIC` automates the collection of multiple data, reduces human error, and reduces subjective operator variability.

Software design

`AIgarMIC` can be used through a collection of command-line scripts; knowledge of Python scripting is not necessary. Given a collection of images from one or more agar dilution experiments, `AIgarMIC` can calculate the MIC from a single script:

```
AIgarMIC -m model/ -t binary -n 0 -d 8 12 -r 160 160 -o results.csv images/
```

Where,

- `-m, --model` specifies the path to the pre-trained model,
- `-t, --type_model` specifies the type of model (binary or softmax),
- `-n, --negative_codes` specifies the growth code/s that should be classed as no growth,
- `-d, -dimensions` specifies the dimensions of the agar plate images (8 rows and 12 columns in this example),
- `-r, --resolution` specifies the resolution of the images (x and y) on which the model was trained,
- `-o, --output_file` specifies the target `.csv` output file

`AIgarMIC` is designed to be extensible through a Python package. The core functionality is provided through the `Plate` and `PlateSet` classes (see Figure 2 for user-interface API). A `PlateSet` instance is in essence a collection of `Plate` instances, where each `Plate` corresponds to an agar plate with a particular antimicrobial concentration. A minimal example is shown below, consisting of 4 strains tested over 3 dilutions/plates (+ one positive control plate):

```
from aigarmic.plate import Plate, PlateSet

# set up 4 plates of ciprofloxacin (including a positive control plate)
antibiotic = ["ciprofloxacin"] * 4
plate_concentrations = [0, 0.125, 0.25, 0.5]

# temporary list of growth codes for each plate
# each plate has 2x2 inoculated strains
plate_growth_matrices = []
plate_growth_matrices.append([[1, 1],
                              [0, 1]])
plate_growth_matrices.append([[1, 1],
                              [0, 0]])
plate_growth_matrices.append([[1, 0],
                              [0, 0]])
plate_growth_matrices.append([[1, 0],
                              [0, 0]])

# combine data into Plate instances
plates = []
for ab, conc, growth in zip(antibiotic,
                            plate_concentrations,
                            plate_growth_matrices):
    plates.append(Plate(drug=ab,
                        concentration=conc,
                        growth_code_matrix=growth))

# create PlateSet instance using list of Plates
plate_set = PlateSet(plates_list=plates)

plate_set.calculate_mic(
    no_growth_key_items = tuple([0]) # growth codes that indicate no growth
```

```
)
plate_set.mic_matrix.tolist()
# [[1.0, 0.25], [0.125, 0.125]]

# convert to traditional MIC values:
plate_set.convert_mic_matrix(mic_format='string').tolist()
# [['>0.5', '0.25'], ['<0.125', '<0.125']]

# check QC:
plate_set.generate_qc().tolist()
# [['P', 'P'], ['F', 'P']]
```

Plate
float concentration -> Antibiotic concentration
str drug -> Antibiotic name
ndarray image -> Plate image
list[list[ndarray]] image_matrix -> Small colony images
list[list[int]] growth_code_matrix -> Growth codes for each colony image
int n_row -> Plate row dimensions
int n_col -> Plate column dimensions
Model model -> Model to predict colony images growth codes
add_growth_code_matrix(list[list[int]] growth_code_matrix) : None -> Manually add growth code matrix
split_images() : None -> Split plate image into small colony images
import_image(ndarray image) : None -> Associate image with plate
get_colony_image() : tuple[ndarray, str] -> Get small colony image from image_matrix
link_model(Model model) : None -> Link image prediction model
get_key() : list[str] -> Get growth key of linked model
set_key(list[str] key) : None -> Set growth key of linked model
annotate_images() : list[list[str]] -> Convert colony images to growth codes using linked model
print_matrix() : None -> Print growth matrix
review_poor_images() : list[tuple[int, int]] -> Get images with poor prediction accuracy

PlateSet
str drug -> Antibiotic name
Plate positive_control_plate -> Plate without antibiotic
list[Plate] antibiotic_plates -> Plates with antibiotic
list[str] key -> Key to convert growth codes to string
get_all_plates() : list[Plate] -> Get all attached plates
convert_mic_matrix() : array[str] -> Convert calculated MIC format
calculate_mic() : array -> Calculate MIC from attached plates
generate_qc() : array -> Generate QC from attached plates
review_poor_images() : list[list[tuple[int, int]]] -> Get images with poor prediction accuracy
get_csv_data() : list[dict] -> Export to CSV-compatible format

Figure 2: `AIgarMIC` Plate and PlateSet API.

In this example, images were not used – growth codes were provided directly in matrix format. By providing images (imported using the `opencv` library to `Plate` instances, `AIgarMIC` can automatically classify growth codes using a pre-trained model. `AIgarMIC` comes with a collection of assets (example images and pre-trained models) to help users get started with the software (Gerada, Harper, Howard, Reza, Hope, & Liverpool Clinical Laboratories, 2024). Details of the built-in models, which are implemented as `keras` models (convolutional neural networks), can be found in the accompanying laboratory validation manuscript (Gerada, Harper, Howard, Reza, & Hope, 2024).

Alternatively, users can provide a custom model by inheriting from the base `Model` class (or the `KerasModel` class if using a `keras` model). Custom models must implement the `predict` method, which takes a colony image as input, and returns a `dictionary` containing, at a minimum, a `growth_code` member. Figure 3 shows the API for the `Model` class and subclasses.

To support users in developing custom models, `AIgarMIC` provides an annotator script that allows users to generate annotated colony images to train and test a custom model. The generated labelled images can be used to train a model using a training script (which uses the neural network architecture design reported in Gerada, Harper, Howard, Reza, & Hope (2024)). Other convenience features are available within the command line interface.

Further examples and tutorials can be found in the documentation.

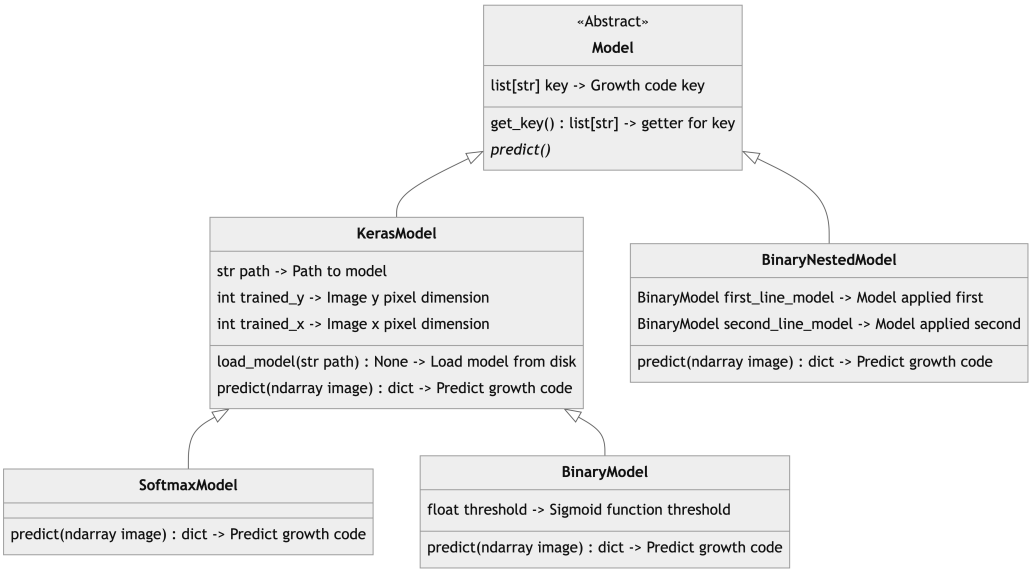

Figure 3: `AIgarMIC` Model API.

Statement of need

Antimicrobial susceptibility testing (AST) is required to ensure timely and appropriate antimicrobial therapy worldwide. Susceptibility testing is also used to quantify the incidence and prevalence of antimicrobial resistance in hospitals, regions and countries. Agar dilution is a standard AST method – it has the advantage of being relatively inexpensive and enables high throughput. However, since most agar dilution experiments are interpreted by visually inspecting agar plates of microbial growth, the implementation of agar dilution is often limited by this time-consuming step. Visual inspection is also subject to human error, and the inherent subjectivity of classifying colony growth can lead to significant intra- and inter-observer variability.

The aim of `AIgarMIC` is to standardise and automate the interpretation of agar dilution plates,

reducing the impact of human error on results. Furthermore, since the performance of `AIgarMIC` is fixed to a particular model, MIC results are not subject to operator variability. Hence, MIC results can be more reliably compared between experiments and laboratories. Figure 1 illustrates the integration of `AIgarMIC` within the laboratory workflow for agar dilution MIC measurement. Typical users of `AIgarMIC` are likely to include:

- Laboratories that are currently performing agar dilution MIC testing, but wish to automate and standardise the interpretation of their results,
- Laboratories that need moderate–high throughput MIC testing, but do not have access to other automated assays and systems.

Related resources

Users of `AIgarMIC` may also be interested in the following related resources and software:

- Laboratory protocols for agar dilution MIC assays, such as those published by the European Committee on Antimicrobial Susceptibility Testing (EUCAST) (EUCAST, 2000) or by Wiegand et al. (Wiegand et al., 2008).
- Software such as `cellprofiler` as a general biological image analysis tool that can be used for tasks beyond the scope of `AIgarMIC` (Lamprecht et al., 2007).

Laboratory validation

`AIgarMIC` has undergone research validation against a wide range of antimicrobials, against a gold standard of manual annotation. It has mainly been tested on clinical *Escherichia coli* strains (Gerada, Harper, Howard, Reza, & Hope, 2024).

Funding and acknowledgements

`AIgarMIC` was funded, in part, by UKRI Doctoral Training Program (AG) [grant ref: 2599501] and the Wellcome Trust [grant ref: 226691/Z/22/Z]. The bacterial strains used in this study were provided by Liverpool Clinical Laboratories.

References

EUCAST. (2000). Determination of minimum inhibitory concentrations (MICs) of antibacterial agents by agar dilution. *Clinical Microbiology and Infection*, *6*(9), 509–515. https://doi.org/10.1046/j.1469-0691.2000.00142.x

Gerada, A., Harper, N., Howard, A., Reza, N., & Hope, W. (2024). Determination of minimum inhibitory concentrations using machine-learning-assisted agar dilution. *Microbiology Spectrum*, e04209–23. https://doi.org/10.1128/spectrum.04209-23

Gerada, A., Harper, N., Howard, A., Reza, N., Hope, W., & Liverpool Clinical Laboratories. (2024). *Image and model assets for AIgarMIC (agar dilution minimum inhibitory concentration software)* [Data set]. DataCat. https://doi.org/10.17638/DATACAT.LIVERPOOL.AC.UK/2631

Lamprecht, M. R., Sabatini, D. M., & Carpenter, A. E. (2007). CellProfiler: Free, versatile software for automated biological image analysis. *BioTechniques*, *42*(1), 71–75. https://doi.org/10.2144/000112257

Wiegand, I., Hilpert, K., & Hancock, R. E. W. (2008). Agar and broth dilution methods to determine the minimal inhibitory concentration (MIC) of antimicrobial substances. *Nature Protocols*, *3*(2), 163–175. https://doi.org/10.1038/nprot.2007.521