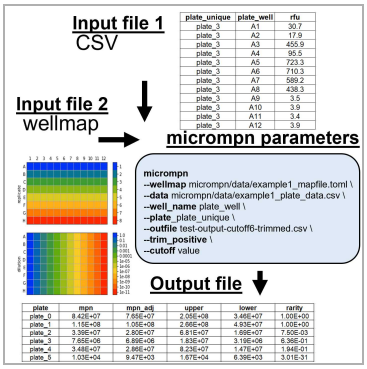


JUN 02, 2023

OPEN ACCESS

Protocol Citation: Adam R Rivers, Karla Franco Melendez 2023. MicroMPN: Software for Automating Most Probable Number Estimates from Laboratory Microplates. **protocols.io** https://protocols.io/view/micrompn-software-for-automating-most-probable-num-cunnwvde

Protocol status: In development
We are still developing and optimizing this protocol

Created: May 23, 2023

Last Modified: Jun 02, 2023

PROTOCOL integer ID: 82350

MicroMPN: Software for Automating Most Probable Number Estimates from Laboratory Microplates

Adam R Rivers[1], Karla Franco Melendez[1]

[1]United States Department of Agriculture, Agricultural Research Service

Adam R Rivers: code author
Karla Franco Melendez: application of code, protocol author

Karla Franco Melendez

DISCLAIMER

The mention of trade names or commercial products on this project is solely for the purpose of providing specific information and neither implies recommendation nor endorsement by the USDA.

ABSTRACT

The most probable number assay is a commonly used method for estimating the concentration of viable microbes, such as bacteria and fungi (1, 2) present in environmental samples, including soil (3 - 6), sediment (7 - 10), water (11-15), and food (1, 16, 17). An MPN count is based on the presence or absence of a measurable signal, such as media turbidity, relative fluorescence units or luminescence. MPN calculations require two components: the volumes of a sample used at each dilution step and the number of replicates per dilution. Results from consecutive dilutions are used to calculate the total cell density in the original sample. Although the MPN assay was originally designed to be run using culture tubes, it was later adapted to microplates to simplify data collection (18) and enable high-throughput screenings of samples (19). We developed a user-friendly command-line python package named "MicroMPN" to enable automation of MPN value estimations from microplates. Users can calculate an MPN estimate from microplates of different formats, arbitrary dilution factors, from samples spread across multiple plates or within the same plate. Users must input two data files for MicroMPN to work. Additionally, users must specify a cutoff/threshold value above which wells from a microplate are considered positive for growth of a microbial target. Input file 1: user specifies the layout of a microplate with a TOML-based format file. This file contains the number of replicates, dilution series, and the number of samples per plate. The python package “wellmap” (20) parses the information of a microplate from the TOML file for downstream calculation of an MPN value. Input file 2: user provides a CSV file with plate name, the well location in a microplate, and the spectrofluorometric data. Output file: MicroMPN will calculate an MPN value (21), a bias corrected MPN value (22), 95% confidence intervals (21), and a rarity index (23, 24). The calculation of MPN is modeled on code in the “MPN” R package created by Ferguson and Ihrie (2019).

GUIDELINES

- As an alternative to command line, there are web-based MPN calculators. One is the Environmental Protection Agency MPN calculator (https://mostprobablenumbercalculator.epa.gov/mpnForm). Another is MPNCalc created by M. Ferguson and J. Ihrie (galaxytrakr.org)https://mpncalc.galaxytrakr.org/). We do not cover their usage in this protocol.

- Additionally, users can use the R package "MPN" written by the same authors that created MPNCalc, and whose code we transpiled to python to run MicroMPN from the terminal.

BEFORE START INSTRUCTIONS

Github repository:
https://github.com/USDA-ARS-GBRU/micrompn.git

Installation of "micrompn" via command line:

git clone git@github.com:USDA-ARS-GBRU/micrompn.git
cd micrompn
pip install .

In this protocol we cover how to run MicroMPN from a Jupyter Notebook.

Example of Input file 1: namefile.csv

1 Example of CSV file with relative fluorescence units (RFU) data input_protocols_io.csv. In this example, a microplate contains 8 replicates by 11 10-fold serial dilutions.

Column names:
plate_unique
plate_well
rfu

	"undiluted"	1E^-1	1E^-2	1E^-3	1E^-4	1E^-5	1E^-6	1E^-7	1E^-8	1E^-9	1E^-10	1E^-11
T48	1	2	3	4	5	6	7	8	9	10	11	12
A	30.731	17.946	455.882	95.484	723.326	710.278	589.153	438.318	3.51	3.889	3.435	3.949
B	31.577	15.791	344.181	645.707	659.22	609.989	550.889	3.987	4.657	3.928	3.034	4.466
C	32.668	14.326	341.61	330.182	6.429	619.234	566.758	4.032	3.786	3.83	3.444	4.009
D	30.411	14.158	469.088	172.275	707.629	671.296	578.231	479.924	4.13	3.539	3.642	3.685
E	31.595	14.33	499.213	580.556	645.975	584.489	479.636	3.918	4.489	3.962	4.204	4.169
F	32.206	13.835	399.741	599.977	620.921	619.279	482.488	380.272	3.624	3.662	3.956	3.55
G	30.276	15.835	7.797	628.614	625.079	618.613	517.805	3.878	3.735	4.255	3.89	4.266
H	30.048	20.28	396.572	327.656	664.395	617.465	501.418	443.293	386.115	3.755	4.01	4.374

8 by 12 microplate layout example. In this microplate, column 1 has 8 replicates of

resuspended soil sample. Columns 2 - 12, contain 10-fold serially diluted samples. Values correspond to fluorometric measurements collected 48 h after incubating the plate. Highlighted in green are wells positive for rfp-tagged bacterial pathogen. Since the MPN assay is based on the principle of dilution to extinction, columns 1 and 2 are also considered positive.

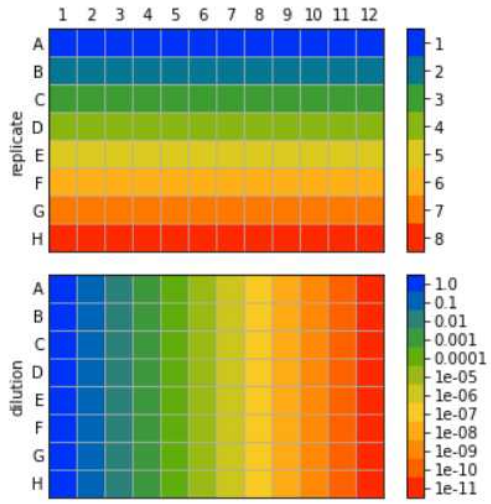
Before running MicroMPN, this data was transposed to three columns [plate_unique, plate_well, rfu].

Example of Input file 2: namefile.toml

2 In this .toml file microplate.toml , each row (A – H) was assigned a replicate value (1 – 8), while each column (1 – 12) represented a particular dilution factor (1e00 – 1e-11). For this reason, each plate was divided into a single 12x8 sample block, although multiple sample blocks on a plate are possible.

```
In [2]: wellmap.show("microplate.toml")#prints two versions of the same image (png, img)
```

Out[2]:



Essentially, the python package "wellmap", which is a dependency of "micrompn", will parse the data from the CSV file based on the .toml specifications. For more information on how to create a .toml file with different plate formats visit [wellmap- File format for 96-well plate layouts]

Example of how to run MicroMPN from a Jupyter Notebook

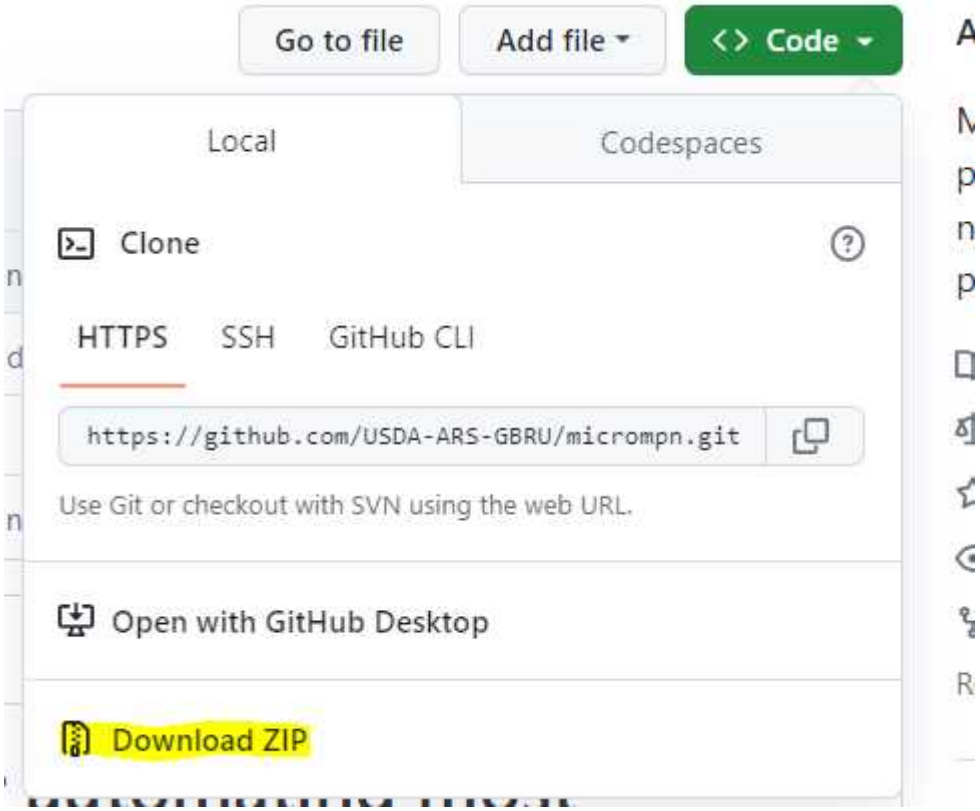
3 Step-by-step example on how to run the command-line tool MicroMPN from a Jupyter Notebook.

**** To run micrompn from a jupyter notebook use the symbol "!" before micrompn; micrompn was designed to run from the command-line not from a jupyter notebook or python shell.***

** For ease of documentation and a user-friendly interface, we used jupyter notebooks for MPN calculations. For more information on jupyter notebooks users are referred to (Jupyter Project Documentation).*

*For information on how to install packages in jupyter notebook visit Installing Python Packages from a Jupyter Notebook.

3.1 Download MicroMPN zipped files from the GitHub repo. Extract files locally.



3.2 Launch a jupyter notebook from the Jupyter terminal or directly from the Anaconda Navigator.

3.3 # In a jupyter notebook code cell, install micrompn by running the following line of code:

! pip install C:\Users\location_of_extracted_files\micrompn-main\micrompn-main

**In our case, "! pip install" worked because both the shell terminal and kernel are using the same python executable. However, this is not always the case. This is explained in detail in Installing Python Packages from a Jupyter Notebook.*

Alternatively, users can install micrompn with the following lines of code:

import sys
print(sys.executable)

#Copy the output from "sys.executable", which should be the path to the Python executable active in the kernel.

In a new code cell, run:
!{sys.executable output} -m pip install C:\Users\location_of_extracted_files\micrompn-main\micrompn-main

3.4

# In a new code cell, proceed to import micrompn.
import micrompn

Check proper installation by running:
! micrompn --help

If installation was successful, user should see the following output:

```
usage: micrompn [-h] --wellmap WELLMAP --data DATA --cutoff CUTOFF --outfile
                OUTFILE [--plate_name PLATE_NAME] [--value_name VALUE_NAME]
                [--well_name WELL_NAME] [--col_name COL_NAME]
                [--row_name ROW_NAME] [--zero_padded] [--trim_positives]
                [--version] [--logfile LOGFILE]

MicroMPN: Software to estimate Most Probable Number (MPN) bacterial abundance
from microplates

optional arguments:
  -h, --help            show this help message and exit
  --wellmap WELLMAP     A TOML file with plate layout specified in wellmap
                        format
  --data DATA           A csv file or a directory containing csv files with
                        the plate name, optical value, and well or row and
                        column data
  --cutoff CUTOFF       The value from the plate reader above which a well is
                        classified as positive
  --outfile OUTFILE     The file path and name for the results
  --plate_name PLATE_NAME
                        The name of the column containing the plate identifier
                        in the data file
  --value_name VALUE_NAME
                        The name of the column containing the optical signal
                        column in the data file
  --well_name WELL_NAME
                        The name of the column containing the well identifier
                        in the data file
  --col_name COL_NAME   The name of the column containing the plate column
                        identifier in the data file
  --row_name ROW_NAME   The name of the column containing the plate row
                        identifier in the data file
  --zero_padded         If present, the well value in the data file is treated
                        as zero-padded, e.g. A01
  --trim_positives      If present, the list of positive wells will be trimmed
                        to the most dilute all positive dilution and the least
                        dilute all negative dilution. This helps if early
                        dilutions are turbid.
  --version, -v         show program's version number and exit
  --logfile LOGFILE, -l LOGFILE
```

3.5 # To visualize the .toml file, run the following command:
wellmap.show("microplate.toml")

**Make sure the .toml file is located in the same directory as your jupyter notebook (nameofnotebook.ipynb).*

If for some reason this does not work, then install the package "wellmap" before running the command wellmap.show()

! pip install wellmap
import wellmap

3.6 # In a new code cell, run micrompn.
CSV file name: protocols_io_RFU_example.csv
TOML file name: microplate.toml

**Make sure the .csv and .toml files are located in the same directory as your jupyter notebook (nameofnotebook.ipynb).*

#run the following code in a single line
! micrompn --wellmap microplate.toml --data
"C:\location\of\CSV\file\protocols_io_RFU_example.csv" --cutoff 6
--outfile "C:\location\of\CSV\output\file\OUTFILE_protocols_io_RFU_example.csv"
--plate_name "plate_unique"
--value_name "rfu"
--well_name "plate_well"
--trim_positives

3.7 Example of output file output_protocols_io.csv.

	A	B	C	D	E	F	G	H
		plate	sample	mpn	mpn_adj	upper	lower	rarity
	0	plate_3	3	7646785	6885935	18317578	3192197	0.63574

Since MPN estimated values are based on the volume of a sample used at each dilution step, we must adjust the MPN estimate for the difference in volume by multiplying by 10. The original sample volume in column 1 of our microplate example was 0.1 mL. With increasing serial dilutions this original amount decreases by a factor of 10: 0.01, 0.001, 0.0001, etc. Additionally, for the example above, we would further normalize the MPN estimate by the average grams of soil in order to convert MPN/mL to MPN/g. Finally, prior to any statistical analysis, $\log_{10}$ transform MPN values.