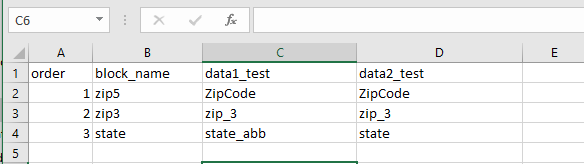
MAMBA2 User Guide

*Introduction*

MAMBA 2 represents a substantial improvement in the flexibility and scalability of the original MAMBA, enabling the use on datasets beyond the Census’ Business Register, using Natural Language Processing techniques to stem and clean textual data, and allow for comparisons of different match metrics in the same model. It also selects the best performing model from a list of Machine Learning algorithms.

*Getting started*

* System requirements: Python 3.7
* Download from our GitHub repo, <https://github.com/john-cuffe/MAMBA2>
* MAMBA is going to match two datafiles. You can call them whatever you want, and save the names of the .csv files (without the .csv ending) in the run\_Match.bash file as data1\_name and data2\_name respectively. This .bash file will then feed these names through all of the subsequent programs.
* Other files:
  + /data/\*data1\_name\*.csv
    - The name of the first dataset you want to match. Must contain a unique column ‘id’ to server as your unique identifier for that record.
  + /data/\*data2\_name\*.csv
    - The name of the second dataset you want to match. Must contain a unique column ‘id’ to server as your unique identifier for that record.
  + Block\_names.csv
    - This file is going to tell MAMBA which variables in your dataset serve as blocks. MAMBA does a comparison if all records within a block against each other.
    - Variables:
      * order: which order (1 being the lowest level) you want the blocks run in. If a record on a dataset is matched in a block, it is not examined in any subsequent blocks.
      * block\_name: A naming convention for the block
      * \*data1\_name\*: This column header should be your data1\_name in your run\_match.bash file. This is the name of the variable that corresponds to the block in the first dataset you wish to match
      * \*data2\_name\*: This column header should be your data1\_name in your run\_match.bash file. This is the name of the variable that corresponds to the block in the second dataset you wish to match.
  + Below: a demonstration of the block\_names file for the test data. Datanames were data1\_test and data2\_test.



* + mamba\_variable\_types.csv
    - This file tells MAMBA what kind of analysis to do on different kinds of variables
    - Variables:
      * Variable\_name: generic name for the comparison.
      * \*data1\_name\*: This column header should be your data1\_name in your run\_match.bash file. This is the name of the variable that corresponds to the variable in the first dataset you wish to match
      * \*data2\_name\*: This column header should be your data1\_name in your run\_match.bash file. This is the name of the variable that corresponds to the variable in the second dataset you wish to match.
      * match\_type: What kind of match analysis do you want to perform?
        + fuzzy: generate 12 fuzzy string comparators for the pair
        + num\_distance: difference between the two values.
        + exact: If the two values match, scored as a 1, otherwise scored as a 0.
        + geo\_distance: Distance between two points (in kilometers) based on Haversine formula. Using this type requires that each dataset has a latitude and longitude variable, and this column is completely filled for all observations.
    - All of these variables will be fed into the MAMBA models to generate matches.
  + training\_data.csv
    - This is the data that will tell MAMBA what you believe is a match and which is not. Currently, MAMBA requires a truth deck of matches in order to build off of. This file only contains three columns.
      * \*data1\_name\*\_id: the id for the record in the first dataset
      * \*data2\_name\*\_id: the id for the record in the second dataset
      * match: 1 if the pair is a match, 0 otherwise.
* Run\_match.bash
  + This is the actual .bash file you need to edit to run MAMBA. It contains the following variables:
    - Data1\_name
      * The name of the first dataset. Exclude the ‘.csv’ ending.
    - data2\_name
      * The name of the second dataset. Exclude the ‘.csv’ ending.
    - db\_name
      * The name you want to give your database. Exclude the ‘.db. ending
    - outputPath:
      * The output directory you want to use
    - Debugmode
      * True/False that skips the majority of matches if used. Set to False
    - block\_file\_name:
      * the name of the file that you are using to define the blocks.
    - create\_db\_chunksize:
      * A variable to set how big of a ‘chunk’ you push to your db at once.
    - inputPath:
      * Path to the datasets you want to match.
    - null\_match\_method:
      * placeholder. ‘zero’ means that any missing data receives a ‘0’ as a score for that match metric. Future development will add imputation methods.
    - Training\_data\_name:
      * The name of the training data you are using. Exclude the ‘.csv’ ending
    - Rf\_jobs:
      * Number of jobs you want to calculate the random forest on. Note, for python you want to have one job per CPU for reasons I don’t entirely understand.
    - clerical\_review\_candidates:
      * Boolean. If True, then python generates clerical review candidates for you to do further training data creation.
    - clerical\_review\_threshold:
      * What predicted probability do you want to limit clerical review candidates to? Generally, you want this to be closer to .5 and 0, as low probability matches won’t help the model determine harder cases.
    - match\_threshold:
      * what is the threshold you want to consider a ‘match’.
    - chatty\_logger:
      * If True, logger returns an entry after every block matched.
    - Log\_file\_name:
      * Name of the .log file you want to use. Exclude ‘.log’ from the name
    - numWorkers:
      * number of workers you want to run the matching on. As with rf\_jobs, assume one job per CPU you are utilizing.
    - Prediction:
      * Do you want MAMBA to predict matches. If set to False, then you can generate clerical review candidates only (if clerical\_review\_candidates=True)
    - Scoringcriteria:
      * A scoring criteria selected from scikit-learn’s list. See <https://scikit-learn.org/stable/modules/model_evaluation.html>
    - Ignore\_duplicate\_ids:
      * Boolean. If True, assumes that you are attempting to de-duplicate the same file, and thus does not compare records with matching ID variables. If False, then compares all records as normal.
    - Use\_logit:
      * Boolean. If True, run a logit and set scoringcriteria to accuracy. Logger will return a warning message about collinearity if it’s bad. If False, doesn’t run a logit.