Journal of Machine Learning Research 21 (2020) 0 Submitted 4/20; Published 0/00

dame-flame and FLAME: Python and R Libraries Providing Fast Interpretable Matching for Causal Inference

Neha R Gupta* NEHA.R.GUPTA@DUKE.EDU
Vittorio Orlandi* VITTORIO.ORLANDI@DUKE.EDU
Department of Statistical Science
Duke University
Durham, NC 27708, USA
Chia-Rui Chang CHIARUI_CHANG@G.HARVARD.EDU
Tianyu Wang TIANYU@CS.DUKE.EDU
Marco Morucci MARCO.MORUCCI@DUKE.EDU
Pritam Dey PRITAM.DEY@DUKE.EDU
Sudeepa Roy SUDEEPA@CS.DUKE.EDU
Cynthia Rudin CYNTHIA@CS.DUKE.EDU
Alexander Volfovsky ALEXANDER.VOLFOVSKY@DUKE.EDU

Editor: editor name here

Abstract

`dame-flame` and `FLAME` are Python and R packages for performing *matching* for *observational causal inference* on datasets containing discrete covariates. These packages implement the *Dynamic Almost Matching Exactly (DAME)* and *Fast, Large-Scale Almost Matching Exactly (FLAME)* algorithms, which match treatment and control units on subsets of the covariates. The resulting matched groups are interpretable, because the matches are made on covariates (rather than, for instance, propensity scores), and high-quality, because machine learning is used to determine which covariates are important to match on. DAME solves an optimization problem that matches units on as many covariates as possible, prioritizing matches on important covariates. FLAME approximates the solution found by DAME via a much faster backward feature selection procedure. While the `FLAME` package does not currently offer an implementation of DAME, `dame-flame` does. It also provides a hybrid algorithm that first uses FLAME to quickly remove less relevant features, and then switches to DAME to produce higher-quality matches on the more important features. This combination strikes a balance between scalability and match quality. The packages provide several adjustable parameters to adapt the algorithms to specific applications.

Keywords: matching, causal inference, interpretability, Python, R

1. Introduction

The *Dynamic Almost Matching Exactly* (*DAME*) (Dieng et al., 2019) and *Fast Large Scale Almost Matching Exactly* (*FLAME*) (Wang et al., 2019) algorithms address the problem of *matching* in observational causal inference. Matching units with identical covariate values helps to reduce bias of treatment effect estimates and is useful in health or social science

*. Primary developers and maintainers of `dame-flame` and `FLAME`

applications where studies cannot randomize individuals into treatment due to ethical, cost, or time constraints. Matching units on covariates permits interpretable analyses that are easier to troubleshoot than other types of analysis for observational causal studies. However, matching is not trivial; in high dimensional settings, few individuals can be matched exactly on all covariates. To handle this problem, DAME and FLAME support *almost exact matching* by identifying important subsets of the covariates using machine learning and matching units exactly on those subsets.

For each unit, DAME solves an optimization problem that finds the largest set of covariates a unit can be matched to another on, prioritizing matches on covariates it learns to be more important. FLAME approximates the solution to the problem solved by DAME; at each step, it drops the covariate leading to the smallest drop in *match quality* `MQ`, defined as $\texttt{MQ} = \texttt{C} \cdot \texttt{BF} - \texttt{PE}$. Here `PE` denotes the *predictive error*, which measures how important the dropped covariate is for predicting the outcome. The *balancing factor* `BF` measures the number of matches formed by dropping that covariate and the discrepancy between the number of treated and control units after the matching. A machine learning algorithm trained on a holdout dataset is responsible for learning the importance of covariates. For more details on the algorithms, see Wang et al. (2019) and Dieng et al. (2019).

The `FLAME` and `dame-flame` packages implement these algorithms, providing users with information about the matched groups, showing which covariates were used to match each unit, all treated and control units that form the *matched group* of each unit, and the estimated individual treatment effect of a match assignment (which is the difference between average treatment and control outcomes within the group). Users are offered several ways to control the matching procedure, via options for how to handle missing data, when to terminate the algorithm, and what machine learning algorithms to use to determine covariate importance. The default parameters for the package were chosen for their versatility and speed, so the algorithm can be relevant and easy-to-use for a range of users.

2. Installation, Documentation, and Project Management

`dame-flame` and `FLAME` are available on GitHub[1], where users may also report bugs and make pull requests. Both repositories include documentation, installation instructions, quick-start tutorials, and `FLAME` also includes a vignette.

`dame-flame` is also available on PyPi and can be installed by entering `pip install dame-flame` in the command line of Python. The package is designed for Python 3, and was tested on Windows and MacOS. `dame-flame` depends on `scikit-learn` version 0.21.3 and above, `pandas` version 0.11.0 and above, and `numpy` version 1.6.1 and above.

`FLAME` is also available on CRAN and can be installed by `install.packages("FLAME")` in R. The package has been tested on MacOS, Windows, and Linux. It depends on standard `tidyverse` packages for data manipulation, `mice` for missing data handling, `gmp` for its efficient bit-vectors implementation, and `xgboost` and `glmnet` for outcome prediction.

Development for both packages underwent code review by various researchers and outside parties to ensure quality and usability. Code quality for `FLAME` is also ensured via a suite of unit tests within the `testthat` framework; coverage is currently at 100%. Testing was done to ensure that `FLAME` and `dame-flame` yield consistent results. Both packages are

1. Both packages are publicly available at `https://github.com/almost-matching-exactly`.

Table 1: Input Data

Unit	x_1	x_2	x_3	x_4	T	Y
0	0	1	1	1	0	5
1	0	1	1	0	1	6
2	1	0	1	1	1	7
3	1	0	0	1	0	4

Table 2: FLAME output

Unit	x_1	x_2	x_3	x_4
0	0	1	1	*
1	0	1	1	*
2	1	0	*	*
3	1	0	*	*

Table 3: DAME output

Unit	x_1	x_2	x_3	x_4
0	0	1	1	*
1	0	1	1	*
2	1	0	*	1
3	1	0	*	1

written in a highly modular fashion, facilitating the implementation of other work within the Almost Matching Exactly framework, such as FLAME-IV and FLAME-Networks (Awan et al., 2019, 2020), which is forthcoming.

3. Package Usage

In this section, we provide an example of running the DAME and FLAME algorithms via the `dame-flame` package. The `FLAME` package offers nearly identical functionality; for more details, users can reference the accompanying vignette.

3.1 Basic Functionality

We run the algorithms on the units in Table 1, whose outcomes were simulated according to $Y_i = 4x_{i1} + 3x_{i2} + 2x_{i3} + 0x_{i4} + T_i$, for units $i = 0$ to 3. That is, x_1 has greatest impact on the outcome, then x_2, x_3, and x_4, respectively (here, x_4 does not affect the outcome).

To run the algorithms, the user provides the parameters shown in Listing 1. Mandatory parameters are: (1) input data as a data frame or file, (2) the name of the outcome column, and (3) the name of the treatment column. In this example, due to the small data size, we opt to make the holdout data the same as the matching data via `holdout_data = 1.0`.

```
import dame_flame
df = pandas.read_csv("dame_flame/data/example.csv")

result_FLAME = dame_flame.DAME_FLAME.FLAME(input_data = df,
    treatment_column_name = "T", outcome_column_name = "Y",
    holdout_data = 1.0)

result_DAME = dame_flame.DAME_FLAME.DAME(df, "T", "Y", 1.0)

# Get matched group of unit 0
dame_flame.DAME_FLAME.mmg_of_unit(result_dame[0], 0, df)

# Get conditional average treatment effect of unit 0
dame_flame.DAME_FLAME.te_of_unit(result[0], 1, df, "treated", "outcome")
```

Listing 1: `dame-flame` example

Tables 2 and 3 summarize the matches made by FLAME and DAME. Each algorithm outputs a table consisting of the set of units that were able to be matched to at least one other unit. For each unit that was matched, the tables indicate which of the covariates were used for matching, and the covariate values that each unit was matched on. The covariates that were not used to match the unit are denoted with "*" as their values. Note that none of the units in Table 1 can be matched using all covariates. So, as FLAME proceeds, it learns that x_4 is irrelevant, and drops it first, matching units 0 and

1. It then learns that x_3 is least relevant out of the remaining covariates and drops it, matching units 2 and 3. DAME, however, which can add covariates back in after they are dropped, is able to match units 2 and 3 on three covariates, instead of just two like FLAME.

Listing 1 also shows how users can find matched groups after either algorithm has been run, using the function `mmg_of_unit`. The result in Table 4 shows the matched group of the input unit, here, unit 0. Estimated treatment effects can also be queried via the function `te_of_unit`, which returns the estimated *conditional average treatment effect* (CATE) of the relevant unit, using the units in the matched group.

Unit	x_1	x_2	x_3
0	0	1	1
1	0	1	1

Table 4: Matched group of unit 0 according to FLAME

3.2 Summary of Parameters for the Algorithms

Full descriptions of algorithm parameters are provided in the respective package documentations; here, we provide a summary of the major groups of customizable parameters. Unless otherwise noted, both `FLAME` and `dame-flame` offer the functionality described below.

Learning methods: `dame-flame` users can choose ridge regression or decision trees to compute predictive error on the holdout dataset when running DAME or FLAME. `FLAME` supports ridge regression and gradient boosting, in addition to arbitrary, user-input methods for this task.

Stopping criteria: DAME and FLAME allow users to specify when the algorithms should stop, e.g., (1) when there are too few unmatched (treatment or control) units, (2) after a certain number of iterations, (3) when predictive error rises too much, or (4) when the balancing factor for a given round is not high enough.

Missing data handling: Users are offered a variety of options for handling missing data in either the holdout set or the data to be matched. They can exclude rows with missingness from the procedure or impute the missing data via MICE (Buuren and Groothuis-Oudshoorn, 2010). Lastly, in the case of missingness in the matching data, users can specify that matches should not occur on missing values, without dropping or imputing them.

Algorithm output: After calls to FLAME or DAME, the matched data is returned by default, as discussed above. Additionally, users can request that the predictive error and/or balancing factor at each iteration be returned, as part of a suite of other printing options.

4. Conclusion

The `dame-flame` and `FLAME` packages offer efficient, easy-to-use implementations of the DAME and FLAME algorithms, allowing users to perform fast, interpretable matching for causal inference for observational data with discrete covariates. The packages are easily accessible and accompanied by detailed documentations, with concrete examples and a vignette. The packages are written in a manner that facilitates the introduction of new features and variations of DAME and FLAME algorithms. Future work will focus on the implementation of other algorithms from the Almost Matching Exactly framework.

Acknowledgments: This work was supported in part by NIH award R01EB025021, NSF awards IIS-1552538 and IIS-1703431, a DARPA award under the L2M program, and a Duke University Energy Initiative Energy Research Seed Fund (ERSF).

References

M. Usaid Awan, Yameng Liu, Marco Morucci, Sudeepa Roy, Cynthia Rudin, and Alexander Volfovsky. Interpretable Almost-Matching-Exactly with Instrumental Variables. In *Proceedings of the Thirty-fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, 22-25 July 2019, Tel Aviv, Israel*, 2019.

M. Usaid Awan, Marco Morucci, Vittorio Orlandi, Sudeepa Roy, Cynthia Rudin, and Alexander Volfovsky. Almost-Matching-Exactly for Treatment Effect Estimation under Network Interference. In *The 23nd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 3-5 June 2020, Palermo, Italy*, 2020.

S van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of statistical software*, pages 1–68, 2010.

Awa Dieng, Yameng Liu, Sudeepa Roy, Cynthia Rudin, and Alexander Volfovsky. Interpretable Almost-Exact Matching for Causal Inference. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 2445–2453, 2019.

Tianyu Wang, Marco Morucci, M. Usaid Awan, Yameng Liu, Sudeepa Roy, Cynthia Rudin, and Alexander Volfovsky. FLAME: A Fast Large-scale Almost Matching Exactly Approach to Causal Inference. *CoRR*, abs/1707.06315v7, 2019. URL `http://arxiv.org/abs/1707.06315`.