Multi-modal learning with incomplete data

Supplementary Information File

Contents

1 Supplementary Information

1.1 Nomenclature

The terms multi-modal and multi-view learning are sometimes distinguished, with multi-view referring to different representations of the same modality (e.g., different crops of an image). However, these terms (as well as multi-source) are often used interchangeably in research. Advances in data preprocessing, particularly deep learning embeddings, have enabled the transformation of unstructured data into structured dataframes, bridging the gap between these fields. In the current era, where almost all data can be vectorized, many algorithms can be applied universally across both multi-modal and multi-view settings. Therefore, we advocate for unifying these concepts under the broader term "multi-modal data". This unification reflects the increasing overlap between these domains and facilitates the application of similar techniques to diverse data types. If specific distinctions are necessary, we suggest using homogeneous modalities (multiple views of the same data type) and heterogeneous modalities (distinct data types).

1.2 Algorithm selection

Figure 1 presents the number of available methods at the time of writing. From the pool of methods, we selected them according to a set of practical and methodological criteria. First, each method must be able to operate without requiring fully observed samples during either training or testing, and without assuming a specific missingness pattern. Second, a method must support an arbitrary number of modalities, rather than being tied to a very narrow multi-modal setting. Third, a method should not be restricted to a specific data type (such as biological pathways, hyperspectral images, histopathological images), so that the package remains relevant across multiple application domains. Fourth, we prioritized methods that are highly cited. Fifth, the code should be open and sufficiently documented.

Methods that did not meet these criteria were set aside for potential future inclusion. However, to maximize the practical impact of the package, we also incorporated a subset of methods specifically tailored to vision-language tasks, which represent a prominent use case in the field. For this particular use case, we included RAGPT (state-of-the-art for image-language classification), as well as MUSE and M^{3}Care (which support multiple modality types, such as images, text, time-series and tabular data).

In summary, the methods that have been included are general enough to handle any number of modalities, and all of them can work with complete or incomplete multi-modal datasets. Thus, the package is designed to provide a broad and useful collection of established, emerging and state-of-the-art algorithms.

Nevertheless, the empirical validation settings used in the original works differ across methods, reflecting the view of the original authors about the miss-

ingness scenarios they were thinking when developing their methods. To make this more transparent, we have summarized the missingness patterns and incompleteness levels used in the original papers when each method was developed and evaluated (Supplementary Table 3). This table helps the reader distinguish between the broad applicability of the package and the more specific conditions under which each algorithm was originally validated.

1.2.1 Python as programming language in the field.

Python is the most widely used programming language for machine learning, supported by various reports and surveys[1] (Supplementary Figure 11a). However, as shown in Fig. 11b, the vast majority of MML methods for incomplete data are based on deep learning, which limit the use of Python in multiple situations (as they require large volumes of data, computational power, and often long execution time for training and testing the model). To address this, we selected a diverse set of algorithms in iMML, including not only deep learning but also alternative strategies such as kernel methods and graph-based approaches. This diversity ensures broader usability and impact across the future community.

While two methods implemented in R were identified, they were not displayed in Fig. 1b due to their low overall frequency.

1.3 Future work

The development of iMML is an ongoing process, with many exciting possibilities for its future growth. As multi-modal learning continues to evolve, it is essential that the tools designed to support it also adapt and expand. In this section, we outline the key areas of improvement and expansion that will help solidify iMML as a comprehensive and powerful library for multi-modal learning tasks. Our roadmap includes enhancing algorithmic diversity, optimizing performance, and expanding compatibility with other data analysis frameworks.

A primary focus is to continue translating algorithms from other programming languages into native Python, as we have already done with some, to minimize dependencies and streamline the user experience. Additionally, we aim to support new data manipulation libraries, such as Polars[2], to enhance compatibility and performance. We will also focus on optimizing computational efficiency. Examples on how to reduce processing times would be by incorporating Cython (a Python compiler that yields performance comparable to C) and/or JAX (a library for high-performance machine learning research with GPU support), although multiple options will be explored. Furthermore, we plan to expand the library with more algorithms, both for existing modules and new ones, enabling iMML to address a wider range of learning tasks.

By implementing these enhancements, we aim to make iMML the one-stop shop for researchers and practitioners in the area, ensuring it remains at the

[1] `https://github.blog/news-insights/octoverse/octoverse-2024/`
[2] https://pola.rs/

forefront of the field.

1.4 Limitations

This work was primarily focused on the development of the iMML package and showcasing its practical benefits in real-world case studies. Therefore, an in-depth analysis of the algorithms and their performance across various scenarios is beyond the scope of this work. While the use cases demonstrated that iMML provides a robust framework for handling IMMD, the evaluation and optimization of specific algorithms for different application domains are left to the end users and future studies. Additionally, while we aimed to replicate the original implementations of the methods as accurately as possible, discrepancies may arise due to differences in frameworks or the limited availability of detailed implementation information in the original publications or source codes. These factors may result in slight variations between our implementations and the original ones. We encourage the authors of the methods to contact us in case they see major discrepancies or when they have new versions available.

2 Supplementary Methods

2.1 Literature review

The publication counts in Fig. 1 were obtained using Scopus[3] with comprehensive search queries, such as "incomplete multi-modal", "incomplete multi-view", "partial multi-modal", "partial multi-view", "missing modal" or "missing view". These terms were required to appear in the title, abstract or keywords. The process was further supplemented by a manual review of relevant literature and online repositories[4] [5] [6] in the field. Lastly, for the plots, in cases where publications provided open-source code in more than one programming language, we divided the count by the number of languages. This ensures that the total annual number of publications remains unchanged. Additionally, some algorithms address multiple tasks (e.g., imputation and clustering), and we categorized them based on the primary focus of the paper.

2.2 Implementation analysis

We tested five datasets: BBCSport[7], BDGP[8], BUAA[9], Nutrimouse[10] and sensIT300[11] (Supplementary Table 5). Hyperparameters were kept at default values, with only the number of clusters set to match the true number of classes in the datasets. Furthermore, the final K-Means clustering algorithm in Matlab engine was replaced by the Scikit-learn implementation with greedy K-Means++ initialization method, shown to produce faster, more stable, and better clustering results. The clusters identified were compared to the ground truth using adjusted Rand index across 50 repetitions with different random seeds (Supplementary Figure 1). Method stability was further assessed through pairwise comparisons of clustering solutions across repetitions, both within and between programming languages, also using the adjusted Rand index (Supplementary Figure 2).

2.3 Amputation

A method for simulating missing modalities was developed to generate various missingness patterns. The function begins in the same manner regardless of the pattern: given a specified percentage of incomplete samples, these are randomly and proportionally split into two groups. The group with incomplete samples is then processed based on the chosen missingness pattern:

[3] https://www.scopus.com/

[4] https://github.com/Jeaninezpp/Awesome-Incomplete-multi-view-clustering

[5] https://github.com/liangnaiyao/multiview_learning

[6] https://github.com/han-liu/awesome-missing-modality-for-medical-images

[7] http://mlg.ucd.ie/datasets/segment.html

[8] http://ranger.uta.edu/~heng/Drosophila/data/

[9] https://github.com/hdzhao/IMG/tree/master/data

[10] https://github.com/mvlearn/mvlearn/tree/main/mvlearn/datasets/Nutrimouse

[11] https://github.com/Liuzhenjiao123/multiview-data-sets/blob/master/sensIT300.mat

- Mutually exclusive missing (MEM): A random subset of samples is selected for the first modality, ensuring these samples are excluded from selection for other modalities. This process is repeated until each modality has a unique group of missing samples. This results in an extreme case where each incomplete sample has only one observed modality. By default, all modalities contribute equally to the dataset, unlike other patterns where the informativeness of the modality affects its impact. The amount of observed data for each modality can be also specified as input.

- Missing completely at random (MCAR): Modalities are randomly assigned as missing. For samples in this group where all modalities are either observed or missing, the status of a randomly selected modality is changed. To maintain the desired proportion of missing modalities, incomplete samples with more than two modalities with the same status are randomly selected, and the status of one modality is adjusted. This process continues until the target percentage of incomplete samples is achieved.

- Partial missing (PM): A subset of modalities is randomly chosen to be observed, while the remaining modalities follow the logic of the missing completely at random pattern.

- Missing not at random (MNAR): A random factor is generated from a distribution within the range (1, number of modalities). Weights for this distribution can be specified as input. For each factor value, a corresponding random set of modalities is designated as missing. All incomplete samples with the same factor value will share the same missing modality profile.

2.4 Imputation

Decomposition methods create condensed, low-dimensional representations that capture the primary sources of heterogeneity in the data. By reversing this process, these methods can also be utilized for data reconstruction, where missing values in the input data are replaced with predictions generated by the model.

2.5 Feature selection

We provide a jNMF-based method for selecting the most important features of an IMMD. While jNMF was originally designed for dimensionality reduction by aggregating features into components, we extend its functionality to measure the contributions of the original features, enabling the identification of those most relevant to the underlying structure of the data. Given a particular number of features required as input, the method provides three selection strategies:

- Component-based selection (default option): the most important features of each component are selected.

- Averaged contribution selection: average the contributions of all features across components and select those with the largest values.

- Absolute contribution selection: identify and select features with the absolute largest contributions across all components.

2.6 Illustrative case studies

All experiments were conducted on a machine equipped with 32 GB of RAM and 20 CPU cores, with the exception of the deep learning models, which were executed on a separate machine with 64 GB of RAM and an NVIDIA A100 GPU. Key hyperparameters used in each case study are summarized in Supplementary Table 4.

2.6.1 Classification and retrieval of an incomplete image-language dataset

The deep learning models were executed using their default hyperparameters and trained for 10 epochs with batch size of 64 and 10 neighbors.

2.6.2 Clustering incomplete multi-modal data in text analysis

We simulated the four block-wise amputation patterns using iMML. For all methods, data were normalized prior to clustering. In each baseline, missing blocks were replaced with the feature-wise mean. The number of clusters was set to the actual number of classes in the dataset.

2.6.3 Imputation of block- and feature-wise incomplete multi-modal data in computer vision

We simulated the four block-wise amputation patterns using iMML, as well as random feature-wise missing data, and applied imputation. Both for MOFA and DFMF, data were z-score scaled before imputation, then reconverted to the original scale by applying an inverse transformation. The baseline imputation, which replaced missing values with the feature-wise mean, did not include this scaling step. The imputed multi-modal data from each method were then concatenated, z-score scaled, and used as inputs for logistic regression to predict their labels and compare with the ground truth (with balanced classes). We also examined the impact of the number of components on performance, finding that while more components generally improved results, both MOFA and DFMF consistently outperformed the baselines across all conditions.

2.6.4 Feature extraction and feature selection on incomplete multi-modal data in biomedicine

We simulated various block- and feature-wise missing data patterns using iMML and applied the jNMF algorithm for feature extraction. For feature selection, the most influential feature from each component was chosen. To provide comparative benchmarks, we included baselines using randomly selected features and all available features. The outputs from these methods were then used

as inputs for a support vector machine to predict genetic type (with balanced classes). As the feature selection process does not replace missing values, an imputation step was applied prior the classification. Relative modality importance was computed by summing the contributions of all features per modality, and then dividing them by the total.

We also evaluated the effect of the number of components/features on performance. Accuracy was particularly low when using only a few components in the extracted features, likely because jNMF, as an unsupervised method, may capture patterns unrelated to the target label. In fact, as explained before, this dataset have two possible outcomes: genotype and diet. In general, across all missing rates, jNMF -both for feature extraction and selection- outperformed the baseline.

3 Supplementary Tables

Supplementary Table 1: **Computing performance comparison between the original and the new Python implementations per dataset**. For each algorithm, the average speed-up and memory gain are reported over fifty repetitions, with minimum and maximum values indicated in parentheses. Values larger than 1 indicate improved performance of the Python implementation (e.g., a value of 2 corresponds to a twofold speed-up or a twofold reduction in memory usage).

Algorithm	Dataset	Speed-up factor (×)	Memory reduction factor (×)
DAIMC	BBCSport	0.68 (0.42,0.88)	1.03 (1.03,1.03)
DAIMC	BDGP	1.37 (0.84,2.09)	1.10 (1.10,1.10)
DAIMC	BUAA	0.27 (0.24,0.29)	1.44 (1.38,1.62)
DAIMC	Nutrimouse	1.32 (1.07,1.83)	2.13 (2.04,2.19)
DAIMC	sensIT300	1.21 (0.98,1.55)	1.83 (1.78,1.85)
EEIMVC	BBCSport	4.04 (3.80,4.46)	1.43 (1.38,1.53)
EEIMVC	BDGP	2.02 (1.78,2.33)	1.07 (1.07,1.07)
EEIMVC	BUAA	13.33 (12.00,15.51)	1.42 (1.09,1.54)
EEIMVC	Nutrimouse	14.96 (13.42,16.23)	2.27 (2.14,2.35)
EEIMVC	sensIT300	7.91 (7.08,8.77)	1.50 (1.48,1.53)
IMSR	BBCSport	2.61 (1.97,4.83)	1.35 (1.18,1.43)
IMSR	BDGP	1.46 (1.11,1.57)	0.54 (0.54,0.55)
IMSR	BUAA	5.25 (4.65,5.86)	0.91 (0.44,1.38)
IMSR	Nutrimouse	5.88 (2.74,13.86)	1.96 (1.20,2.23)
IMSR	sensIT300	4.20 (2.78,7.25)	1.11 (1.10,1.12)
LFIMVC	BBCSport	3.14 (3.08,3.21)	1.65 (1.59,1.74)
LFIMVC	BDGP	3.18 (3.12,3.36)	1.09 (1.09,1.10)
LFIMVC	BUAA	14.87 (14.32,15.36)	1.74 (1.73,1.75)
LFIMVC	Nutrimouse	16.10 (15.82,16.55)	2.91 (2.80,3.20)
LFIMVC	sensIT300	9.38 (9.28,9.49)	1.71 (1.68,1.75)
NEMO	BBCSport	9.51 (8.01,10.81)	2.10 (1.92,2.29)
NEMO	BDGP	3.91 (2.45,6.11)	1.01 (1.01,1.02)
NEMO	BUAA	3.16 (2.75,3.66)	2.12 (1.79,2.32)
NEMO	Nutrimouse	4.89 (4.22,5.40)	3.63 (2.84,4.71)
NEMO	sensIT300	6.57 (5.11,7.50)	1.21 (1.18,1.26)
SIMCADC	BBCSport	2.57 (2.40,2.65)	1.64 (1.60,1.76)
SIMCADC	BDGP	2.78 (2.19,3.27)	1.10 (1.10,1.11)
SIMCADC	BUAA	7.71 (7.35,8.39)	1.63 (1.62,1.65)
SIMCADC	Nutrimouse	10.48 (10.34,10.64)	2.12 (1.94,2.26)
SIMCADC	sensIT300	3.83 (3.03,4.36)	1.64 (1.58,1.82)

Supplementary Table 2: **Multi-modal statistics of survival prediction in pancreatic cancer**. The modality combination for predicting survival that provides most information is proteins and gene expression. Information refers to amount of information that the modality combination provides about survival prediction [1]. The table indicates the average across 25 repetitions.

Modality(1)	Modality(2)	Information	Redundancy(%)	Uniqueness(1)(%)	Uniqueness(2)(%)	Synergy(%)
Proteins	GEx	0.42	0.12	0.10	0.02	0.76
CNA	Proteins	0.40	0.17	0.05	0.06	0.72
CNA	GEx	0.36	0.17	0.08	0.00	0.74
CNA	Mutations	0.18	0.18	0.31	0.00	0.51
Mutations	GEx	0.17	0.18	0.02	0.17	0.63
Proteins	Mutations	0.17	0.18	0.36	0.02	0.44

Supplementary Table 3: **Conditions tested during algorithm development in the original publications**. None of the method included in iMML is based on a specific mathematical assumption for the missingness pattern, and therefore each can operate on any missingness pattern; this table shows which missingness patterns were used by the authors to validate the algorithms. Algorithms or conditions that were not fully described or could not be inferred from the paper were not included in this table.

Algorithm	Missingness patterns				Missing data		Maximum Rate (%)
	MCAR	MEM	MNAR	PM	Simulated	Real	
DAIMC			✓		✓		50
EEIMVC	✓				✓		90
IMSCAGL	✓				✓		90
IMSR	✓				✓		50
IntegrAO	✓			✓	✓	✓	90
LFIMVC	✓				✓		90
M^{3}Care				✓	✓	✓	60
MKKMIK	✓				✓		90
MRGCN				✓	✓		70
MOFA	✓			✓	✓	✓	90
MONET	✓		✓		✓		40
MUSE	✓			✓	✓	✓	90
NEMO				✓	✓		80
OMVC			✓		✓		40
OPIMC			✓		✓		50
OSLFIMVC	✓				✓		50
PID				✓	✓		50
PIMVC			✓	✓	✓		70
RAGPT		✓		✓	✓		90
SIMCADC			✓		✓		50
SUMO				✓	✓	✓	90

Abbreviations: MCAR, missing completely at random; MEM, mutually exclusive missing; MNAR, missing not at random; PM, partial missing.

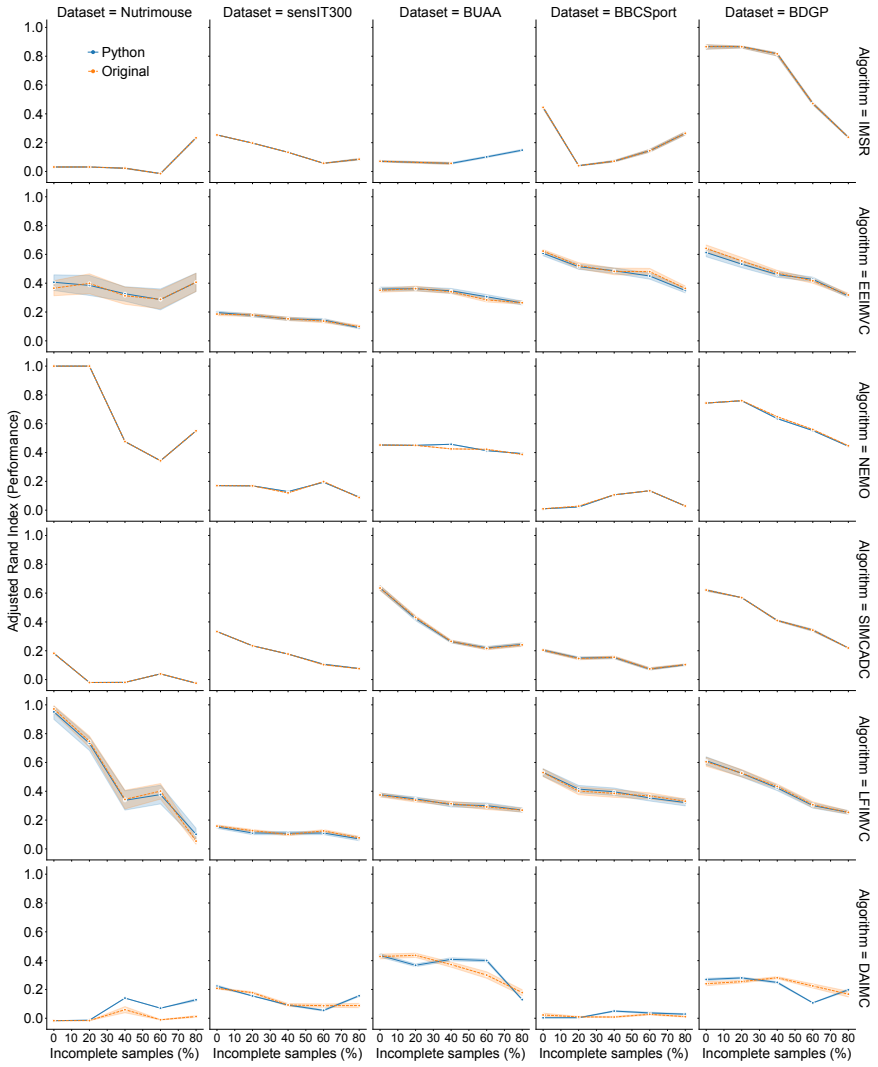
Supplementary Table 4: **Hyperparameters of the methods used in the experiments**. For other hyperparameters not indicated in this table, default parameters in the library were used (iMML v0.3.1). When a list is indicated, benchmarking with all the values was performed and results provided.

Use case	Algorithm	Hyperparameters
Classification	RAGPT	Batch size=32, n_neighbors=20, epochs=10
Classification	M^3Care	Batch size=32, epochs=10
Clustering	IMSCAGL	Clusters=5 (#classes)
Clustering	IMSR	Clusters=5 (#classes)
Clustering	LFIMVC	Clusters=5 (#classes)
Clustering	PIMVC	Clusters=5 (#classes)
Clustering	SIMCADC	Clusters=5 (#classes)
Imputation	MOFA	Components=[1,2,4,8,16]
Dimensionality reduction	jNMF	Components=[1,2,4,8,16]
Feature selection	jNMF	Components=[1,2,4,8,16]

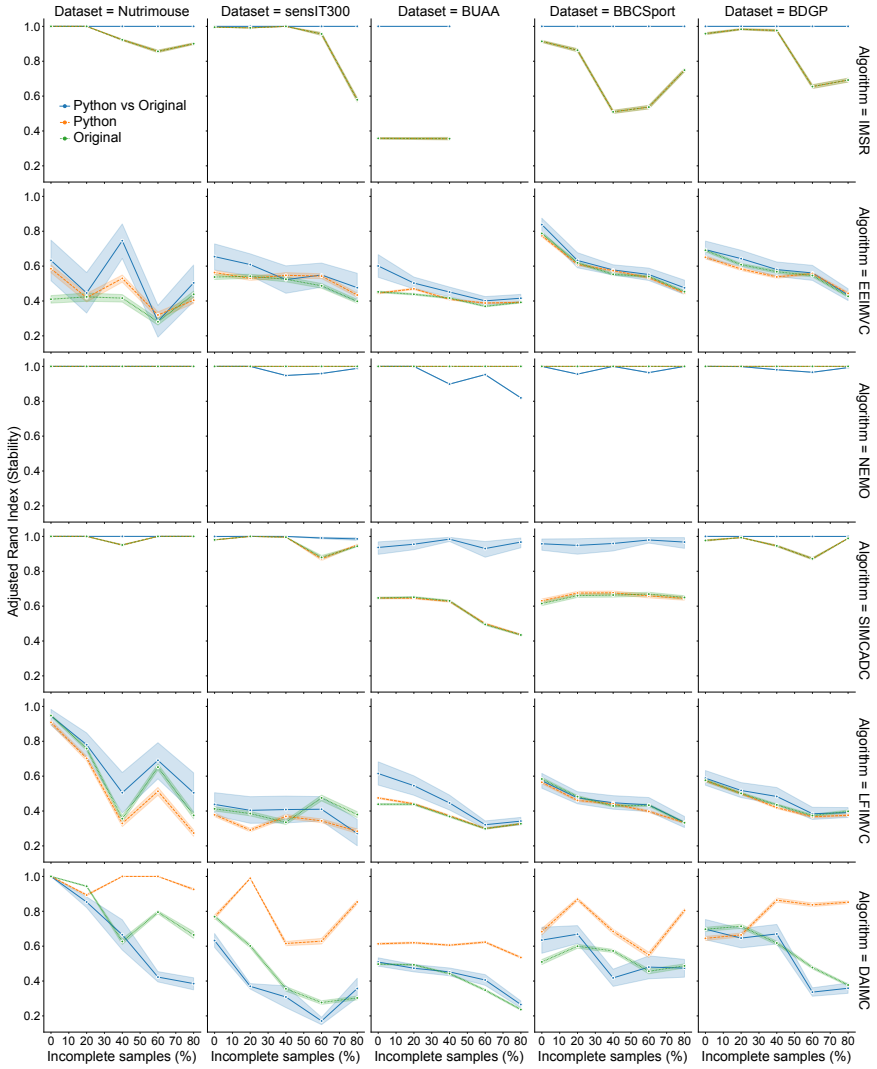
Supplementary Table 5: **Description of datasets used for evaluating translation.**

	Modalities	Samples	Features	Classes
BBCSport	4	116	[1991, 2063, 2113, 2158]	5
BDGP	2	2500	[1750, 79]	5
BUAA	2	90	[100, 100]	10
Nutrimouse	2	40	[120, 21]	2
sensIT300	2	300	[50, 50]	3

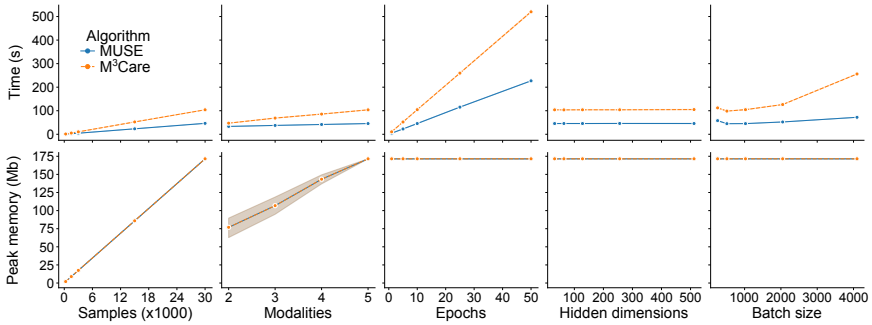
4 Supplementary Figures



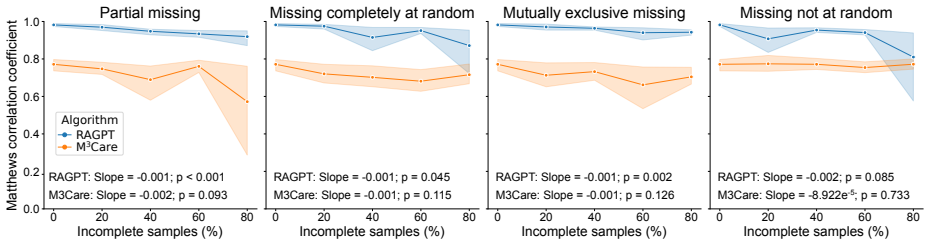
Supplementary Figure 1: **Performance comparison between the original and the new Python implementations**. Six clustering algorithms (rows) were translated to Python and evaluated on five multi-modal datasets (columns). Clustering solutions were compared against ground truth using adjusted Rand index. Larger values indicate better performance. For all algorithms, the results of the two implementations are nearly identical, with all $p = 1$ using Nemenyi-Wilcoxon-Wilcox test, indicating no evidence against the null hypothesis that the implementations produce the same distributions. IMSR failed to converge in the original implementation on the BUAA dataset when the proportion of incomplete samples exceeds 60%. Results are averaged over 50 repetitions, with 95% confidence intervals indicated.



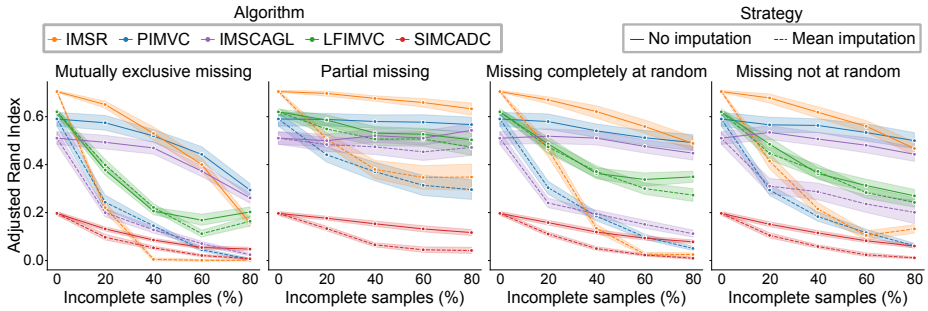
Supplementary Figure 2: **Stability comparison between the original and the new Python implementations**. Stability was measured by pairwise comparison of clustering solutions across repetitions using adjusted Rand index. Larger values indicate greater stability. We assessed the stability of the clustering solutions by repeating the algorithms 50 times within the same programming language, and also comparing the clustering solutions between the two programming languages. Stability for IMSR on the BUAA dataset was not evaluated when the proportion of incomplete samples exceeded 60%, since the original implementation did not converge under those conditions. For NEMO (a deterministic method), the difference was due to floating-point precision in intermediate steps across languages, with an average $RMSE = 1.72e-16$. Results are averaged across repetitions, with 95% confidence intervals indicated.



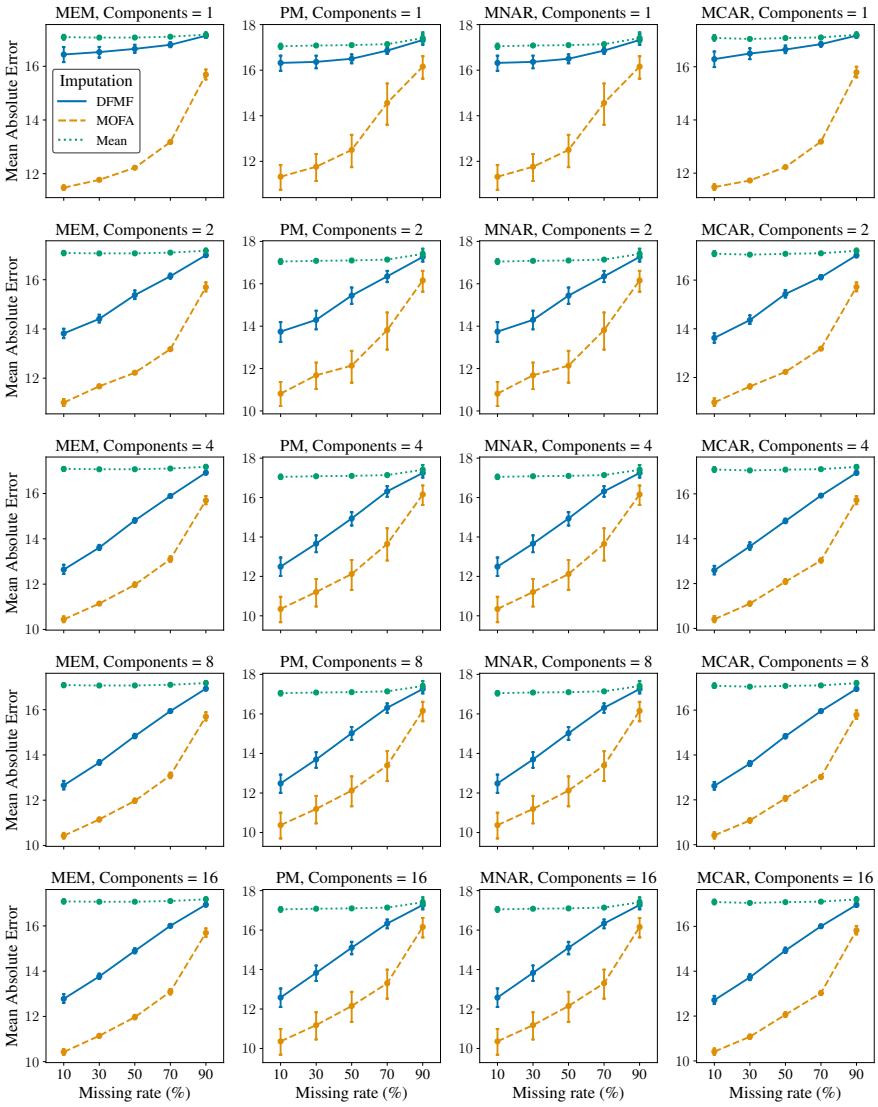
Supplementary Figure 3: **Scalability comparison**. MUSE and M^{3}Care were evaluated for computing time (top) and peak memory (bottom), while varying the number of samples, modalities, epochs, hidden dimensions and batch size on the large-scale NUSWIDE dataset. Runtime is strongly dependend on the number of epochs, whereas peak memory is most strongly affected by the number of samples, followed by the number of modalities. Only a single variable was adjusted at a time, while the others remained at their default values (samples = 30000, modalities = 5, epochs = 5, hidden dimensions = 128, batch size = 1024) Results are averaged over 10 repetitions, with 95% confidence intervals indicated.



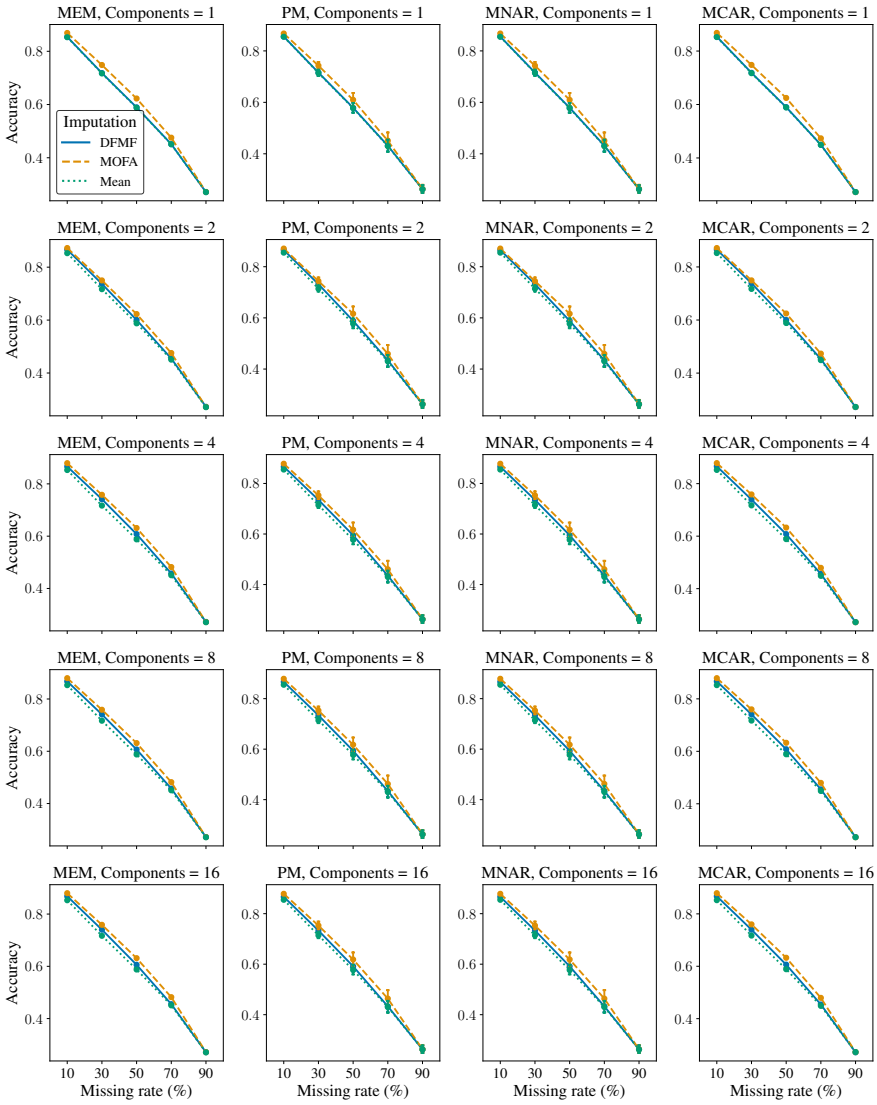
Supplementary Figure 4: **Classification performance on an incomplete vision-language dataset**. Stratified five-fold cross-validation of RAGPT and M^{3}Care algorithms on the Food101 dataset across various missing rates (from 0% to 80%) and different data missingness patterns using Matthew's correlation coefficient. A linear model was fitted for each model and missingness pattern, to estimate how the performance varies when using incomplete data. Results are averaged across folds, with 95% confidence intervals indicated.



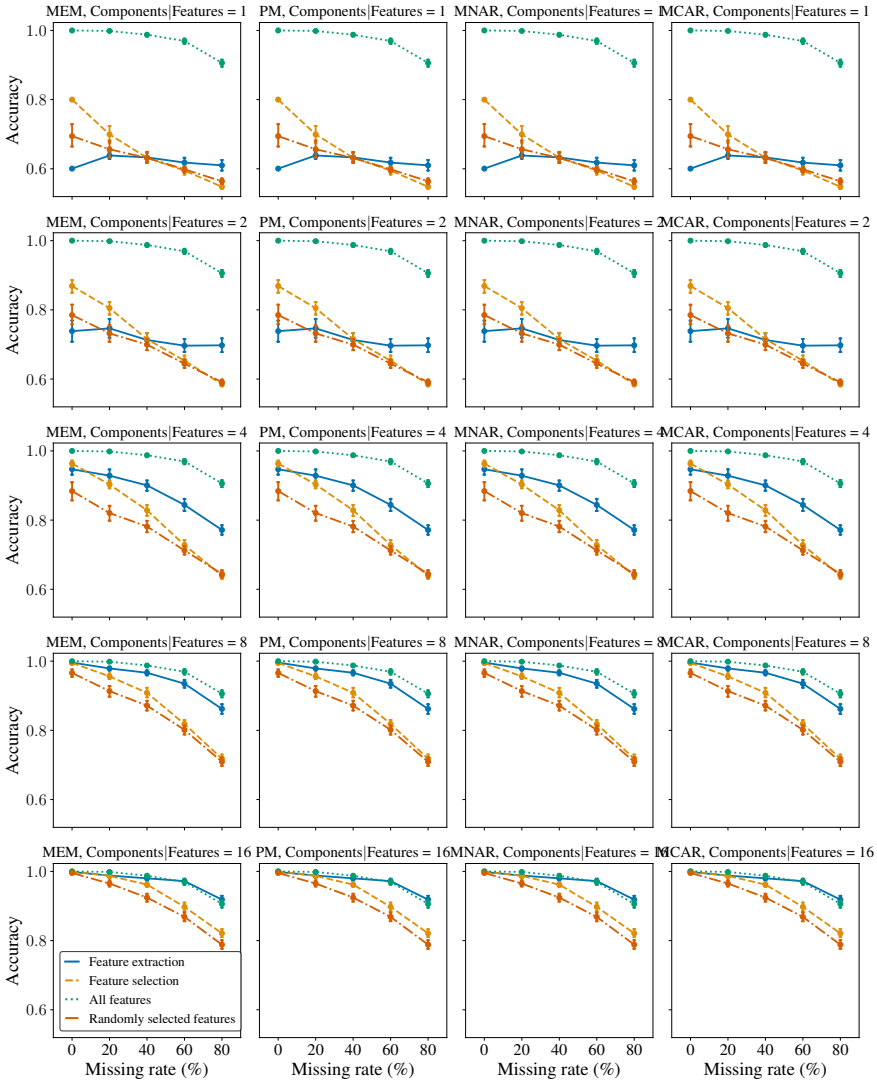
Supplementary Figure 5: **Clustering performance on incomplete multi-modal text data**. The clustering solutions by IMSCAGL, IMSR, LFIMVC, PIMVC and SIMCADC on the BBCSport dataset, as well as their respective baselines (where missing values were replaced with the feature-wise mean), are compared to ground truth topics across various missing rates (from 0% to 80%) under different data missing patterns using adjusted Rand index. The mean of 50 repetitions with 95% confidence intervals are shown.



Supplementary Figure 6: **Analysis on the number of components and missingness pattern on imputation performance**. Mean absolute error of imputation results for MOFA, DFMF and baseline against real values across increasing numbers of components for fixed missing rates (from 10% to 90%). The baseline imputation was performed by replacing missing values with the average of each feature. Each result is the mean of 50 repetitions, with 95% confidence intervals shown. Columns represent different missingness patterns, while rows show results for varying numbers of components.



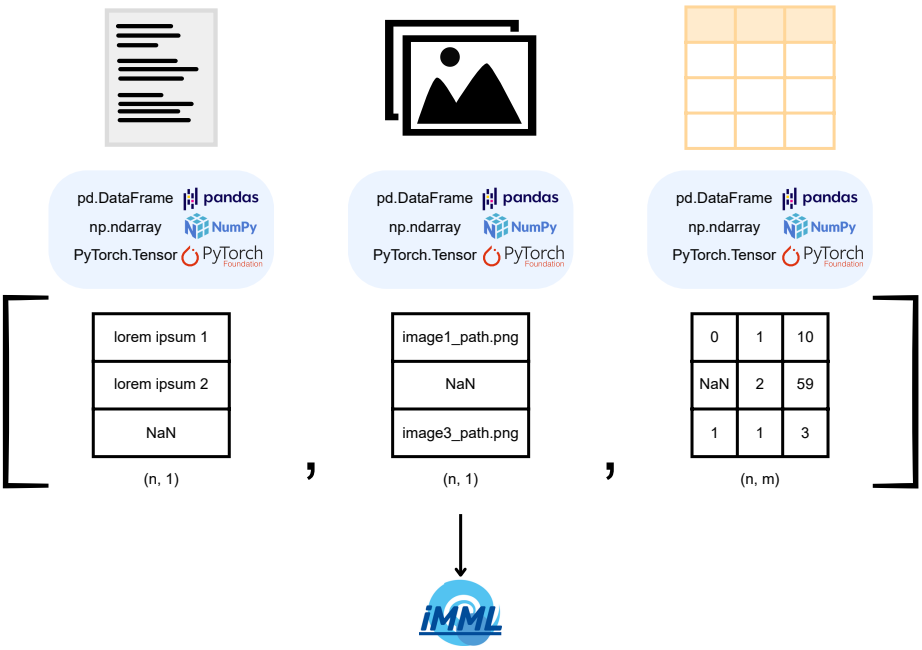
Supplementary Figure 7: **Analysis on the number of components and missingness pattern on classifying imputed data**. Classification accuracy usig imputed data from MOFA, DFMF and a baseline against for fixed missing rates (from 10% to 90%). The baseline imputation was performed by replacing missing values with the average of each feature. Each result is the mean of 50 repetitions, with 95% confidence intervals shown. Columns represent different missingness patterns, while rows show results for varying numbers of components.



Supplementary Figure 8: **Feature extraction and selection with jNMF across different numbers of components / features and missing rates**. Classification accuracy achieved using extracted features, selected features, all features and randomly selected features across varying missing rates (0% to 80%). Results are averaged over 50 repetitions, with 95% confidence intervals shown. Columns represent different missingness patterns, while rows show results for varying numbers of components/features.

```python
import numpy as np
from imml.cluster import NEMO
# Load an (incomplete) multi-modal dataset
Xs = [np.random.random((10,5)) for i in range(3)]
# Define an instance of an algorithm
estimator = NEMO(n_clusters=3)
# Fit the model and get predictions
labels = estimator.fit_predict(Xs)
```

Supplementary Figure 9: **Sample code**. With iMML, once the functions are imported and a multi-modal dataset is loaded, only two lines of code are needed: one to define the method and parameters, and another to obtain the results.



Supplementary Figure 10: **Input format and processing flow in iMML package**. The package accepts a list of matrix-like objects, such as a NumPy array, pandas dataframe, or PyTorch tensor for text, images and tabular data. Text is stored as a one-column representation containing the raw text entries, while image inputs are stored as paths so that loading can occur efficiently within each batch and avoid memory overload. Missing values are represented as NaN, which allows the same pipeline to handle structured and unstructured modalities in a unified way. This constitutes the input format for methods implemented in iMML, so the users do not need to add any preprocessing steps, as each algorithm applies its own transformation. Typically, images are loaded using the PIL library, converted to a PyTorch tensor, and then encoded using a convolutional neural network or visual transformer; texts are tokenized and encoded with a language model such as BERT. n is the number of samples, and m is the number of features in a table.

(a) Popularity of programming languages. (b) Number of deep learning approaches.

Supplementary Figure 11: (a) Worldwide relative interest in Python, Matlab, C++, R and Java, measured by the number of searches in Google (source: Google Trends, `https://trends.google.com/trends`). (b) Publications with Python open-source code (from Fig. 1b) are further categorized into "Deep Learning" and "Others". The proportion of publications by category is indicated in the middle of each bar, highlighting the proportion of deep learning approaches compared to non-deep learning methods in the field of multi-modal learning with incomplete data.

References

[1] Paul Pu Liang, Yun Cheng, Xiang Fan, Chun Kai Ling, Suzanne Nie, Richard Chen, Zihao Deng, Nicholas Allen, Randy Auerbach, Faisal Mahmood, et al. Quantifying & modeling multimodal interactions: An information decomposition framework. *Advances in Neural Information Processing Systems*, 36:27351–27393, 2023.