cmine, minerva & minepy: a C engine for the MINE suite and its R and Python wrappers

Davide Albanese [1,+], Michele Filosi [1,2,+], Roberto Visintainer [1,3,+], Samantha Riccadonna [1], Giuseppe Jurman [1] and Cesare Furlanello [1*]

[1]Fondazione Bruno Kessler, via Sommarive 18, I-38123 Povo (Trento), Italy
[2]CIBIO, University of Trento, via delle Regole 101, I-38123 Mattarello (Trento), Italy
[3]DISI, University of Trento, Via Sommarive 5, I-38123 Povo (Trento), Italy

ABSTRACT

Summary: We introduce cmine, a novel implementation in ANSI C of the MINE family of algorithms for computing maximal information-based measures of dependence between two variables in large datasets. We also provide two interfaces, minerva and minepy, for the C engine through the R environment and the Python scripting language, respectively. The cmine solution reduces the large memory requirement of the first Java implementation for both the R and Python interfaces. Results on microarray and RNA-seq transcriptomics datasets are described.

Availability and Implementation: Source code implemented in ANSI C (cmine) with wrappers in R (minerva) and Python (minepy) are freely available for download under GPL3 licence[1]. The R package minerva is also available through the CRAN repository[2]. The Python library minepy is also in SourceForge[3]. All software is multiplatform (MS Windows, Unix/Linux and Mac OS X).

Contact: furlan@fbk.eu

Supplementary information: Supplementary information are available at the cmine website `http://mpba.fbk.eu/cmine`.

1 INTRODUCTION

The Maximal Information-based Nonparametric Exploration (MINE) family of statistics, including the Maximal Information Coefficient (MIC) measure, was recently introduced in (Reshef *et al.*, 2011), aimed at fast exploration of two-variable relationships in many-dimensional data sets. MINE consists of the algorithms for computing four measures of dependence — MIC, Maximum Asymmetry Score (MAS), Maximum Edge Value (MEV), Minimum Cell Number (MCN) — between two variables, having the generality and equitability property. Generality is the ability of capturing variable relationships of different nature, while equitability is the property of penalizing similar levels of noise in the same way, regardless of the nature of the relation between the variables. The MINE suite received immediate appraisal as a real breaktrough in the data mining of complex biological data (Speed, 2011) as well as criticisms (Simon and Tibshirani, 2012; Gorfine *et al.*, 2012). Many groups worldwide have already proposed its use for explorative data analysis in computational biology, from networks interaction dynamics to virus ranking (Weiss *et al.*, 2012; Das *et al.*, 2012; Anderson *et al.*, 2012; Karpinets *et al.*, 2012; Faust and Raes, 2012). Together with the algorithm description, the MINE authors provided a Java implementation (MINE.jar), two wrappers (R and Python), and four reference datasets (Reshef *et al.*, 2011). However, applicability of MINE.jar on all pairs of features on large datasets is currently limited due to memory requirements and computing time (Miller, 2012). It is also clear the interest for a native parallelization of MINE tasks, which is currently unavailable. These issues represent an obstacle for a systematic application of MINE algorithms to high-throughput omics data — for example, as a substitute of Pearson correlation in network studies. Inspired by these considerations, we propose cmine, a C implementation of the MINE algorithms, and two interfaces to cmine from R (minerva) or Python (minepy).

2 THE MINE C ENGINE AND ITS WRAPPERS

The cmine engine is written in ANSI C by reimplementing *ex novo* the algorithms originally described in (Reshef *et al.*, 2011) and its supplementary material (the source code of MINE.jar is not distributed). The C code provides three structures describing respectively the data, the parameter configuration and all the corresponding maximum normalized mutual information scores. The core function *mine_compute_score* takes a dataset structure and a configuration structure as input, returning the score structure as output. Given a score structure, four functions compute the MINE statistics. The minepy Python module works with Python $\geq 2.6, 3.X$, with Numpy $\geq 1.3.0$ as the sole requirement: the interface consists of the class *minepy.MINE* whose methods correspond to the cmine functions. The R package minerva is built as an R wrapper (R $\geq$ 2.14) to cmine following the standard procedure detailed in (R Core Team, 2012). The main function *mine* takes the dataset and the parameter configuration as inputs and returns the four MINE statistics. Minerva allows native parallelization: based on the R package *parallel*, the number of cores can be passed as parameter to the *mine* function, whenever multi-core hardware is available. The curated version of the CDC15 Spellman yeast dataset (Spellman *et al.*, 1998) used in (Reshef *et al.*, 2011) is included as example. Documentation (on-line or as a PDF) for cmine and minepy is available at the cmine website, while minerva

*to whom correspondence should be addressed; + authors equally contributed to this work.

[1] `http://mpba.fbk.eu/cmine`

[2] `http://cran.r-project.org`

[3] `http://minepy.sourceforge.net`

```
>>> # imports the numpy module
>>> import numpy
>>> # imports the minepy module
>>> import minepy
>>> # create x = [0, 0.001, 0.002, ..., 0.998, 0.999, 1]
>>> x = numpy.linspace(0, 1, 1001)
>>> # y = sin(10 * pi * x) + x
>>> y = numpy.sin(10 * numpy.pi * x) + x
>>> # build the MINE object
>>> mine = MINE(alpha=0.6, c=15)
>>> # computes the information scores
>>> mine.score(x, y)
>>> # returns the Maximal Information Coefficient (MIC)
>>> mine.mic()
0.9999992800928936
>>> # returns the Maximum Asymmetry Score (MAS)
>>> mine.mas()
0.7281444902035837
```

(a)

```
> # load the minerva package
> library(minerva)
> # create x = c(0, 0.001, 0.002, ..., 0.998, 0.999, 1)
> x <- seq(0, 1, 0.001)
> # y = sin(10 * pi * x) + x
> y <- sin(10 * pi * x) + x
> # computes the information scores
> res <- mine(x, y, alpha=0.6, C=15)
> # returns the Maximal Information Coefficient (MIC)
> res$MIC
0.9999993
> # returns the Maximum Asymmetry Score (MAS)
> res$MAS
0.7281445
```

(b)

Fig. 1. Code usage example: computing MIC and MAS on two vectors with minepy (a) and minerva (b).

documentation is accessible in R as on-line help or from the CRAN repository. In Fig. 1 we show a usage example for both wrappers.

Performance comparison The cmine suite was tested for consistency with the original MINE.jar v1.0.1 implementation on the Spellman and microbiome reference datasets, available at `http://www.exploredata.net`. For the CDC15 Spellman yeast dataset, 4381 transcripts measured at 23 timepoints, we report in Fig. 2 the comparison of MIC values computed for all features pairs by using the original MINE.jar and minepy, with α=0.67 for both implementations. Most values are identical; few discrepancies (*i.e.*, points deviating from the diagonal) may be due to a different implementation in the clump computation and to the different data type for the floating point values (java float versus C double).

We performed the same all features pairs computations on 23 time points and increasing feature set sizes with MINE.jar and the two cmine wrappers: the RAM and CPU usage are diplayed in Fig. 3. While MINE.jar cannot perform computation for the dataset instance with more than 1000 features, minerva and minepy fulfill all the tasks with a considerable RAM allocation saving (Fig. 3(a)). Computational times are comparable among all the methods even in the parallel implementation of minerva (Fig. 3(b)).

On the microbiome dataset, we computed the MINE functions for all the 6696×6696 pairs. Comparing with Tab S13 of (Reshef *et al.*, 2011), Supplementary Material (77 top ranked association pairs) we

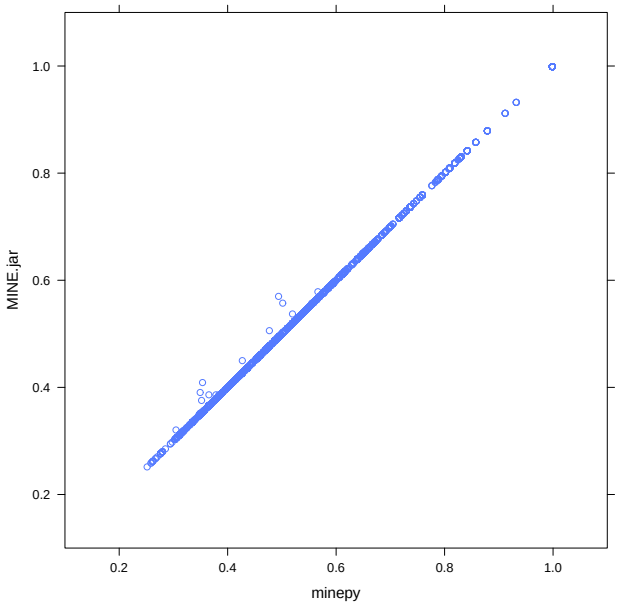


Fig. 2. Comparison of MIC values for variable #1 (time) *vs.* all the other 4381 variables of the CDC15 Spellman yeast dataset by using the original MINE.jar (y axis) and minepy, with α=0.67 for both implementations. The small differences between the two implementations may be due to a varying number of clumps in the MINE.jar and to the different data type for the floating point values.

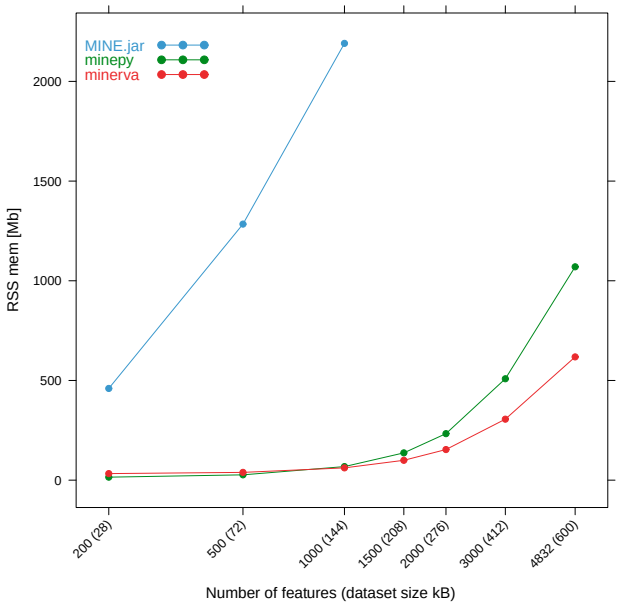
obtained 44 identical results and 73 values whose difference is less than 0.01. The median of all differences is 0, the 3rd quartile is 0.003, and the largest observed difference is 0.014 (complete table available on the cmine website).

We additionally tested the cmine suite on two recent high-throughput transcriptomics datasets, from Affymetrix HumanExon 1.0ST of human brain tissues and Illumina Genome Analyzer II sequencing of human non-small cell lung cancer respectively. Numerical details on datasets and performance of computing the MINE statistics, first variable vs all the others, are reported in Tab. 1. Finally, we tested the scaling properties of minerva with varying values of the α parameter on two uniformly distributed random

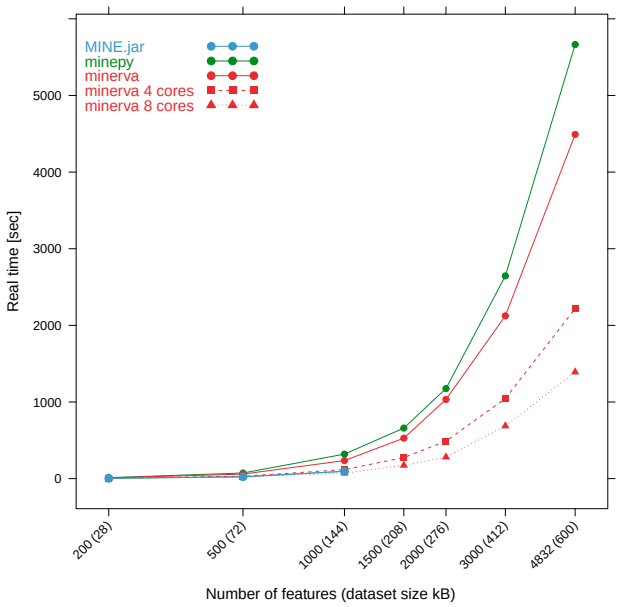
Table 1. Performance of cmine (one versus all) on microarray and sequencing datasets identified by GEO accession number and original reference. n: number of sample. p: number of features. CPU: Elapsed time used by the process (in seconds). RAM: resident set size, *i.e.*, the non-swapped physical memory that a task has used (in kilobytes), for minerva (R) and minepy (P).

GEO Acc. no.	n	p	CPU R	CPU P	RAM R	RAM P
GSE25219[1]	1,340	17,566	21,273	24,276	898,712	1,503,004
GSE34914[2]	16	22,316	1.89	3.74	35,996	32,452

[1] Kang *et al.* (2011) [2] Kalari *et al.* (2012)



(a)



(b)

Fig. 3. (a) Resident set size, *i.e.*, the non-swapped physical memory that a task has used (in MegaBytes) and (b) elapsed time used by the process (in seconds) versus increasing number of features (log scale) to simultaneously compute the MINE statistics for all pairs of features of the CDC15 Spellman yeast dataset, comparing MINE.jar v1.0.1, minerva and minepy. MINE.jar can complete the task only for the first 3 datasets (with 200, 500, and 1,000 features). The number in parentheses in the x-axis labels is the dataset dimension in kilobytes (kB).

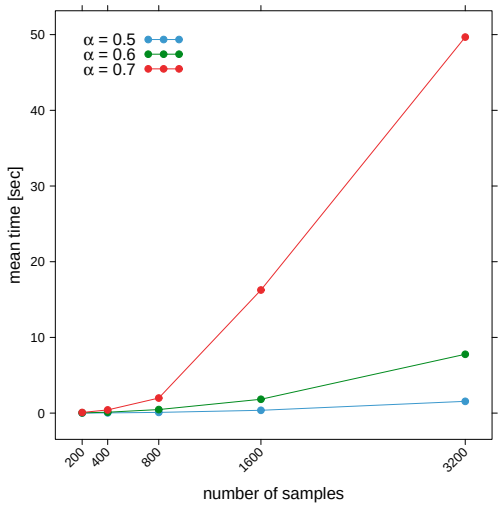


Fig. 4. Average of the elapsed time on 100 repetitions computing the MINE statistics with the minerva package on 2 uniformly distributed random variables for an increasing number of samples and $\alpha = 0.5, 0.6,$ and 0.7.

vectors with increasing length. In Fig. 4, we show the average of 100 replicates for 5 different vector lengths. Due to the linearity in computing MINE statistics on n pairs of variables, Fig. 4 can be used to derive a rough estimate of the total time required to perform a MINE computation on a given dataset.

ACKNOWLEDGEMENTS

Funding: This work was supported by the EU FP7 Project HiperDART.

REFERENCES

Anderson, T., Laegreid, W., Cerutti, F., Osorio, F., Nelson, E., Christopher-Hennings, J., and Goldberg, T. (2012). Ranking viruses: measures of positional importance within networks define core viruses for rational polyvalent vaccine development. *Bioinformatics*, **28**(12), 1624–1632.

Das, J., Mohammed, J., and Yu, H. (2012). Genome-scale analysis of interaction dynamics reveals organization of biological networks. *Bioinformatics*, **28**(14), 1873–1878.

Faust, K. and Raes, J. (2012). Microbial interactions: from networks to models. *Nature Rev Microbiol*, **10**, 538–550.

Gorfine, M., Heller, R., and Heller, Y. (2012). Comment on ”Detecting Novel Associations in Large Data Sets”. Preprint, available at the website http://iew3.technion.ac.il/ gorfinm/files/science6.pdf.

Kalari, K., Rossell, D., Necela, B., Asmann, Y., Nair, A., Baheti, S., Kachergus, J., Younkin, C., Baker, T., Carr, J., Tang, X., Walsh, M., Chai, H., Sun, Z., Hart, S., Leontovich, A., Hossain, A., Kocher, J.-P., Perez, E., Reisman, D., Fields, A., and Thompson, E. (2012). Deep Sequence Analysis of Non-Small Cell Lung Cancer: Integrated Analysis of Gene Expression, Alternative Splicing, and Single Nucleotide Variations in Lung

Adenocarcinomas with and without Oncogenic KRAS Mutations. *Front Oncol*, **2**, 12.

Kang, H., Kawasawa, Y., Cheng, F., Zhu, Y., Xu, X., Li, M., Sousa, A., Pletikos, M., Meyer, K., Sedmak, G., Guennel, T., Shin, Y., Johnson, M., Krsnik, Z., Mayer, S., Fertuzinhos, S., Umlauf, S., Lisgo, S., Vortmeyer, A., Weinberger, D., Mane, S., Hyde, T., Huttner, A., Reimers, M., Kleinman, J., and Sestan, N. (2011). Spatio-temporal transcriptome of the human brain. *Nature*, **478**(7370), 483–489.

Karpinets, T., Park, B., and Uberbacher, E. (2012). Analyzing large biological datasets with association networks. *Nucleic Acids Research*, **First published online May 25, 2012**, gks403v1–gks403.

Miller, S. (2012). Putting Information Relationships to the Test. Blog post, available at the website http://www.information-management.com/blogs/MIC-MINE-predictive-big-data-Harvard-R-10022590-1.html.

R Core Team (2012). *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-11-9.

Reshef, D., Reshef, Y., Finucane, H., Grossman, S., McVean, G., Turnbaugh, P., Lander, E., Mitzenmacher, M., and Sabeti, P. (2011). Detecting novel associations in large datasets. *Science*, **6062**(334), 1518–1524.

Simon, N. and Tibshirani, R. (2012). Comment on ”Detecting novel associations in large data sets” by Reshef et al, Science Dec 16, 2011. Preprint, available at the website http://www-stat.stanford.edu/ tibs/reshef/comment.pdf.

Speed, T. (2011). A Correlation for the 21st Century. *Science*, **6062**(334), 1502–1503.

Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, **9**(12), 3273–3297.

Weiss, J., Karma, A., W.R., M., Deng, M., Rau, C., Rees, C., Wang, J., Wisniewski, N., Eskin, E., Horvath, S., Qu, Z., Wang, Y., and Lusis, A. (2012). ”Good enough solutions” and the genetics of complex diseases. *Circ Res*, **111**(4), 493–504.