pyComBat, a Python tool for batch effects correction in high-throughput molecular data using empirical Bayes methods

Abdelkader Behdenna[1], Julien Haziza[1], Chloé-Agathe Azencott[2,3,4] and Akpéli Nordor[1]

[1]Epigene Labs, Paris, France
[2] MINES ParisTech, PSL Research University, CBIO-Centre for Computational Biology, 75006 Paris, France
[3] Institut Curie, PSL Research University, 75005 Paris, France
[4]INSERM, U900, 75005 Paris, France

Abstract

Summary: Variability in datasets are not only the product of biological processes: they are also the product of technical biases. ComBat is one of the most widely used tool for correcting those technical biases, called batch effects, in microarray expression data.
In this technical note, we present a new Python implementation of ComBat. While the mathematical framework is strictly the same, we show here that our implementation: (i) has similar results in terms of batch effects correction; (ii) is as fast or faster than the R implementation of ComBat and; (iii) offers new tools for the bioinformatics community to participate in its development.

Availability and Implementation: pyComBat is implemented in the Python language and is available under GPL-3.0 (https://www.gnu.org/licenses/gpl-3.0.en.html) license at https://github.com/epigenelabs/pyComBat.

Contact: akpeli@epigenelabs.com

1. Introduction

Batch effects are the product of technical biases, such as variations in the experimental design or even atmospheric conditions (Lander, 1999; Thomas L. Fare *et al.*, 2003). They particularly reveal themselves when merging different datasets, which have likely been built under different conditions. If not corrected, these batch effects may lead to incorrect biological insight, since the variability can be wrongly interpreted as the product of a biological process.
Multiple methods exist that address this problem. They include approaches related to frequentist statistics, such as simple normalization (Yang *et al.*, 2002; Irizarry *et al.*, 2012) or principal component analysis (Nielsen *et al.*, 2002); and machine learning, such as support-vector machines (Benito *et al.*, 2004). One of their main flaws is, however, their incapacity to handle low sample sizes or more than two batches at the same time (Chen *et al.*, 2011).
ComBat, originally implemented in the R library sva (Leek *et al.*, 2012), is based on the mathematical framework defined in (Johnson *et al.*, 2007). This tool leverages a parametric and non-parametric empirical Bayes approach for correcting the batch effect in datasets that works for small sample sizes or in the presence of outliers. Note that the parametric method requires strong assumptions but is largely faster than the non-parametric approach.

We introduce in this article pyComBat, a new Python implementation of ComBat, following the same mathematical framework. We show that it yields comparable results for adjusting for batch effects, but is generally faster, in particular for the usually slow, but more general, non-parametric method.

2. pyComBat

pyComBat is a Python 3 implementation of ComBat. It mostly uses generic libraries like Pandas (McKinney, 2010) or NumPy (Van Der Walt *et al.*, 2011) to mimic ComBat, following the exact same mathematical framework.
Two important features are not directly related to the performances of the software but are of outmost importance. First, pyComBat is available as an open source software under a GPL-3.0 license, which means anyone can use, modify, distribute and share it. Opening pyComBat to the Python for bioinformatics community is the best way for maintaining and improving it, while increasing its robustness. Second, the reliability of pyComBat has been thoroughly checked, using unit testing (with the pytest library, cover=83%) for assessing the proper functioning of each sub-module as well as insuring an easy maintenance, in particular after modifications.

3. Comparison with ComBat

a. Dataset used

For software validation, we used the package bladderbatch version 1.22.0 (Leek, 2019), that contains microarray gene expression data on 57 samples from 5 batches and is the reference example dataset for the sva package. We then compared ComBat and pyComBat on the same dataset (corresponding to the 20,000 first genes of bladderbatch) for (i) power for batch effect correction and; (ii) computation time.

b. Batch effect correction

As an implementation of the ComBat algorithm, pyComBat is expected to have similar, if not identical, power in terms of batch effects correction. This is confirmed in Fig.1A, which shows the distribution of differences between the outputs of ComBat and pyComBat. As expected, the differences are distributed closely around zero (mean = $-9.8 \cdot 10^{-5}$, 95% CI = [-0.03,0.027]). The slight variability can be explained by the different ways R and Python (in particular NumPy) handle matrices and matrix calculation.
To further validate PyComBat, we used Principal Variant Component Analysis (PVCA) (Li *et al.*, 2009) – implemented in R in the library of the same name – to estimate the batch effect before and after applying pyComBat. Fig. 1B and Fig. 1C show that the batch effects are completely removed. We still observe variability due to the interaction between batches and cancer, which is however related to the design of the sampling and not correctable through the same means.

c. Computation time

Computation time is evaluated by running ComBat (resp. pyComBat) 100 times on the bladderbatch dataset presented in section 3a, with the parametric and the non-parametric approaches.

Due to Python efficiency in handling matrix operations and matrix manipulations as well as thorough optimization of our code, pyComBat is also as fast or even faster than ComBat. The

parametric version of the software indeed appears twice as fast as ComBat in terms of computation time (fig.1D), with less variability.
The most striking result concerns the non-parametric version (fig.1E), which is more time consuming, but also less dependent on the distribution of the data. In this case pyComBat is approximatively 15 times faster than ComBat, going from around 100 minutes to less than 10 minutes.

4. Discussion and conclusion

We have presented a new Python implementation for ComBat, the most commonly used software for batch effects correction on high-throughput molecular data. Our implementation offers the same correcting power, with similar computation time for the parametric method, and significantly shorter time for the slower non-parametric version. This reduced computing time opens perspectives for a more generic use of the non-parametric approach to a larger range of datasets.
While developed and tested on microarray gene expression data, ComBat has also been used to correct batch effects for a wider range of high-throughput molecular profiling platforms, such as RNA sequencing platform (Gandal *et al.*, 2018). However, a prior log-transformation of the data is necessary to use ComBat. Similar tools have recently been developed to avoid this additional transformation (Zhang *et al.*, 2020).
We have attached importance to making the software open source and as documented as possible while providing tools for testing modifications to the code. We believe that this will be benefiting the Python bioinformatics community and opening the way towards the translation of other widely used software from R to Python.

Acknowledgements

The authors thank Phuong Pham for his advice about the estimation of the efficiency of the adjustments.

References

Benito,M. *et al.* (2004) Adjustment of systematic microarray data biases. *Bioinformatics*, **20**, 105–114.
Chen,C. *et al.* (2011) Removing batch effects in analysis of expression microarray data: An evaluation of six batch adjustment methods. *PLoS One*, **6**.
Gandal,M.J. *et al.* (2018) Shared molecular neuropathology across major psychiatric disorders parallels polygenic overlap. *Science (80-.).*, **359**, 693–697.
Irizarry,R.A. *et al.* (2012) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Sel. Work. Terry Speed*, 601–616.
Johnson,W.E. *et al.* (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, **8**, 118–127.
Lander,E.S. (1999) Array of hope. *Nat. Genet.*, **21**, 4.
Leek,J.T. (2019) bladderbatch: Bladder gene expression data illustrating batch effects.
Leek,J.T. *et al.* (2012) The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics*, **28**, 882–883.
Li,J. *et al.* (2009) Principal Variance Components Analysis: Estimating Batch Effects in Microarray Gene Expression Data. In, *Batch Effects and Noise in Microarray Experiments: Sources and Solutions*. John Wiley & Sons, Ltd, Chichester, UK, pp. 141–154.
McKinney,W. (2010) Data Structures for Statistical Computing in Python. *Proc. 9th Python Sci. Conf.*, **1697900**, 51–56.

Nielsen,T.O. *et al.* (2002) Molecular characterisation of soft tissue tumours: a gene expression study. *Lancet*, **359**, 1301–1307.

Thomas L. Fare *et al.* (2003) Effects of Atmospheric Ozone on Microarray Data Quality.

Van Der Walt,S. *et al.* (2011) The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.*, **13**, 22–30.

Yang,Y.H. *et al.* (2002) Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.*, **30**, 15e – 15.

Zhang,Y. *et al.* (2020) ComBat-Seq: batch effect adjustment for RNA-Seq count data. *bioRxiv*, 2020.01.13.904730.

Chen,C. *et al.* (2011) Removing batch effects in analysis of expression microarray data: An evaluation of six batch adjustment methods. *PLoS One*, **6**.

Gandal,M.J. *et al.* (2018) Shared molecular neuropathology across major psychiatric disorders parallels polygenic overlap. *Science (80-.).*, **359**, 693–697.

Irizarry,R.A. *et al.* (2012) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Sel. Work. Terry Speed*, 601–616.

Johnson,W.E. *et al.* (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, **8**, 118–127.

Lander,E.S. (1999) Array of hope. *Nat. Genet.*, **21**, 4.

Leek,J.T. (2019) bladderbatch: Bladder gene expression data illustrating batch effects.

Leek,J.T. *et al.* (2012) The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics*, **28**, 882–883.

Li,J. *et al.* Principal Variance Components Analysis: Estimating Batch Effects in Microarray Gene Expression Data. In, *Batch Effects and Noise in Microarray Experiments*. John Wiley & Sons, Ltd, Chichester, UK, pp. 141–154.

McKinney,W. (2010) Data Structures for Statistical Computing in Python. *Proc. 9th Python Sci. Conf.*, **1697900**, 51–56.

Nielsen,T.O. *et al.* (2002) Molecular characterisation of soft tissue tumours: a gene expression study. *Lancet*, **359**, 1301–1307.

Price,E.M. and Robinson,W.P. (2018) Adjusting for Batch Effects in DNA Methylation Microarray Data, a Lesson Learned. *Front. Genet.*, **9**.

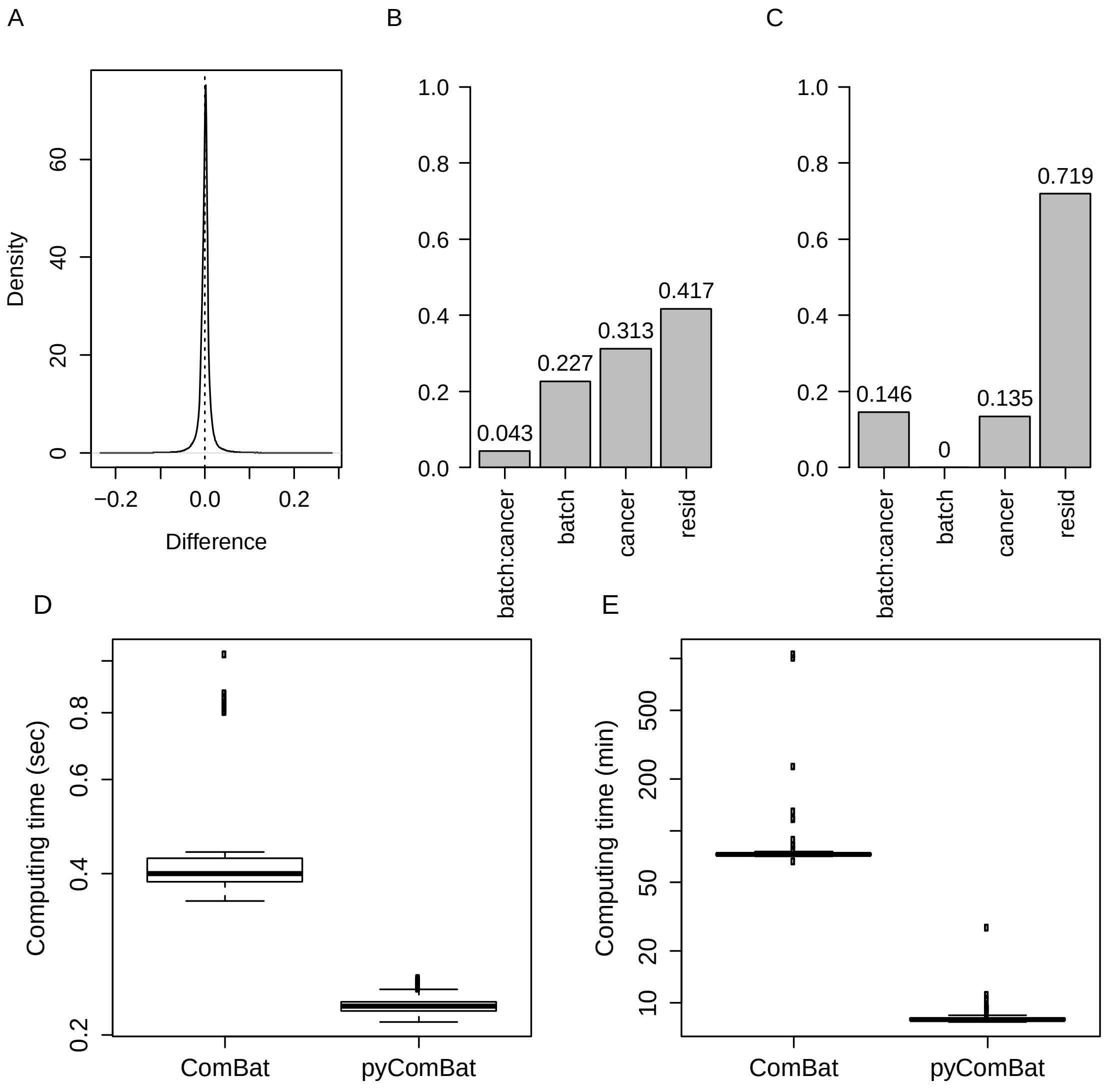
Thomas L. Fare *et al.* (2003) Effects of Atmospheric Ozone on Microarray Data Quality.

Van Der Walt,S. *et al.* (2011) The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.*, **13**, 22–30.

Yang,Y.H. *et al.* (2002) Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.*, **30**, 15e – 15.

Figures

Fig. 1



Legend of figures

Fig. 1

Performance of pyComBat vs. Combat on the bladderbatch data set. **A** Distribution of the differences between the expression matrices corrected for batch effects, respectively by ComBat and pyComBat (parametric version). The vertical dotted line corresponds to zero. **B** Principal Variance Component Analysis (PVCA) estimation of the contribution of “cancer” and “batch” as sources of variability in the raw data. “resid” bar corresponds to the residual, *i.e.* the variability unexplained by the informed sources. **C** PVCA estimation of the contribution of “cancer” and “batch” as sources of variability in the data after the pyComBat correction. **D** Computation time in seconds for ComBat (left) and pyComBat (right) for the parametric method. The y-axis is in a log scale. **E** Computation time in minutes for ComBat (left) and pyComBat (right) for the non-parametric method. The y-axis is in a log scale.