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

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 is 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 and https://pypi.org/project/combat/.

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; 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; Tai and Speed, 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. In comparison to both the R implementation and the

existing Python implementation of ComBat in the single-cell analysis library Scanpy (Wolf *et al.*, 2018), we show that it yields similar results for adjusting for batch effects, but is generally faster, in particular for the usually slow, but more loose, 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=84%) 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. Datasets used

For software validation, we created two meta-datasets from public data, all originated from the Gene Expression Omnibus repository, one on Ovarian Cancer (6 datasets), one on Multiple Myeloma (4 datasets). Both meta-datasets are described in more details in Table 1. We then compared ComBat, Scanpy's implementation of ComBat and pyComBat on both datasets 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, on the Ovarian Cancer dataset. As expected, the differences are distributed closely around zero (mean = $-1.06 \cdot 10^{-7}$, 95% CI = $[-1.28 \cdot 10^{-3}, 1.32 \cdot 10^{-4}]$). The slight variability can be explained by the different ways R and Python (in particular NumPy) handle matrices and matrix calculation.

c. Computation time

Computation time is evaluated by running pyComBat (resp. Scanpy's implementation of ComBat and ComBat itself) respectively 100 times on both datasets presented in section 3a, with the parametric approach. As Scanpy doesn't handle the non-parametric approach, only ComBat and pyComBat have been tested with it, on the Ovarian Cancer dataset.

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 pyComBat indeed appears 4 to 5 times as fast as ComBat, and around 1.5 times as fast as the Scanpy implementation of ComBat, in terms of computation time (fig.1B, fig.1C), on both datasets.
The same results are observed with the non-parametric version (fig.1D), which is more time consuming, but also less dependent on the distribution of the data. In this case pyComBat is

also approximatively 4 to 5 times faster than ComBat, going from more than an hour to around 15 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 shorter computation time for the parametric method compared to other implementations, and significantly shorter time for the slower non-parametric version compared to ComBat. 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. The authors also thank Aryo Gema and Guillaume Appé for their feedback on the code for pyComBat.

References

Benito,M. *et al.* (2004) Adjustment of systematic microarray data biases. *Bioinformatics*, **20**, 105–114.
Bonome,T. *et al.* (2008) A gene signature predicting for survival in suboptimally debulked patients with ovarian cancer. *Cancer Res.*, **68**, 5478–5486.
Chen,C. *et al.* (2011) Removing batch effects in analysis of expression microarray data: An evaluation of six batch adjustment methods. *PLoS One*, **6**.
Dhodapkar,M. V. *et al.* (2014) Clinical, genomic, and imaging predictors of myeloma progression from asymptomatic monoclonal gammopathies (swog s0120). *Blood*, **123**, 78–85.
Driscoll,J.J. *et al.* (2010) The sumoylation pathway is dysregulated in multiple myeloma and is associated with adverse patient outcome. *Blood*, **115**, 2827–2834.
Fare,T.L. *et al.* (2003) Effects of atmospheric ozone on microarray data quality. *Anal. Chem.*, **75**, 4672–4675.
Gandal,M.J. *et al.* (2018) Shared molecular neuropathology across major psychiatric disorders parallels polygenic overlap. *Science (80-.).*, **359**, 693–697.
Huang,C. *et al.* (2018) Machine learning predicts individual cancer patient responses to therapeutic drugs with high accuracy. *Sci. Rep.*, **8**.
Johnson,W.E. *et al.* (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, **8**, 118–127.
Khan,R. *et al.* (2015) Four genes predict high risk of progression from smoldering to symptomatic multiple myeloma (SWOG s0120). *Haematologica*, **100**, 1214–1221.
Lander,E.S. (1999) Array of hope. *Nat. Genet.*, **21**, 4.
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,C. *et al.* (2021) Genetic analysis of multiple myeloma identifies cytogenetic alterations implicated in disease complexity and progression. *Cancers (Basel).*, **13**, 1–15.

Lili,L.N. *et al.* (2013) Molecular profiling predicts the existence of two functionally distinct classes of ovarian cancer stroma. *Biomed Res. Int.*, **2013**.

Lionetti,M., Barbieri,M., Todoerti,K., Agnelli,L., Fabris,S., *et al.* (2015) A compendium of DIS3 mutations and associated transcriptional signatures in plasma cell dyscrasias. *Oncotarget*, **6**, 26129–26141.

Lionetti,M., Barbieri,M., Todoerti,K., Agnelli,L., Marzorati,S., *et al.* (2015) Molecular spectrum of BRAF, NRAS and KRAS gene mutations in plasma cell dyscrasias: Implication for MEK-ERK pathway activation. *Oncotarget*, **6**, 24205–24217.

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

Mok,S.C. *et al.* (2009) A Gene Signature Predictive for Outcome in Advanced Ovarian Cancer Identifies a Survival Factor: Microfibril-Associated Glycoprotein 2. *Cancer Cell*, **16**, 521–532.

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

Tai,Y.C. and Speed,T.P. (2012) A multivariate empirical Bayes statistic for replicated microarray time course data. In, *Selected Works of Terry Speed.*, pp. 617–642.

Tothill,R.W. *et al.* (2008) Novel molecular subtypes of serous and endometrioid ovarian cancer linked to clinical outcome. *Clin. Cancer Res.*, **14**, 5198–5208.

Vathipadiekal,V. *et al.* (2015) Creation of a human secretome: A novel composite library of human secreted proteins: Validation using ovarian cancer gene expression data and a virtual secretome array. *Clin. Cancer Res.*, **21**, 4960–4969.

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

Wolf,F.A. *et al.* (2018) SCANPY: Large-scale single-cell gene expression data analysis. *Genome Biol.*, **19**, 15.

Yamamoto,Y. *et al.* (2016) In vitro and in vivo correlates of physiological and neoplastic human Fallopian tube stem cells. *J. Pathol.*, **238**, 519–530.

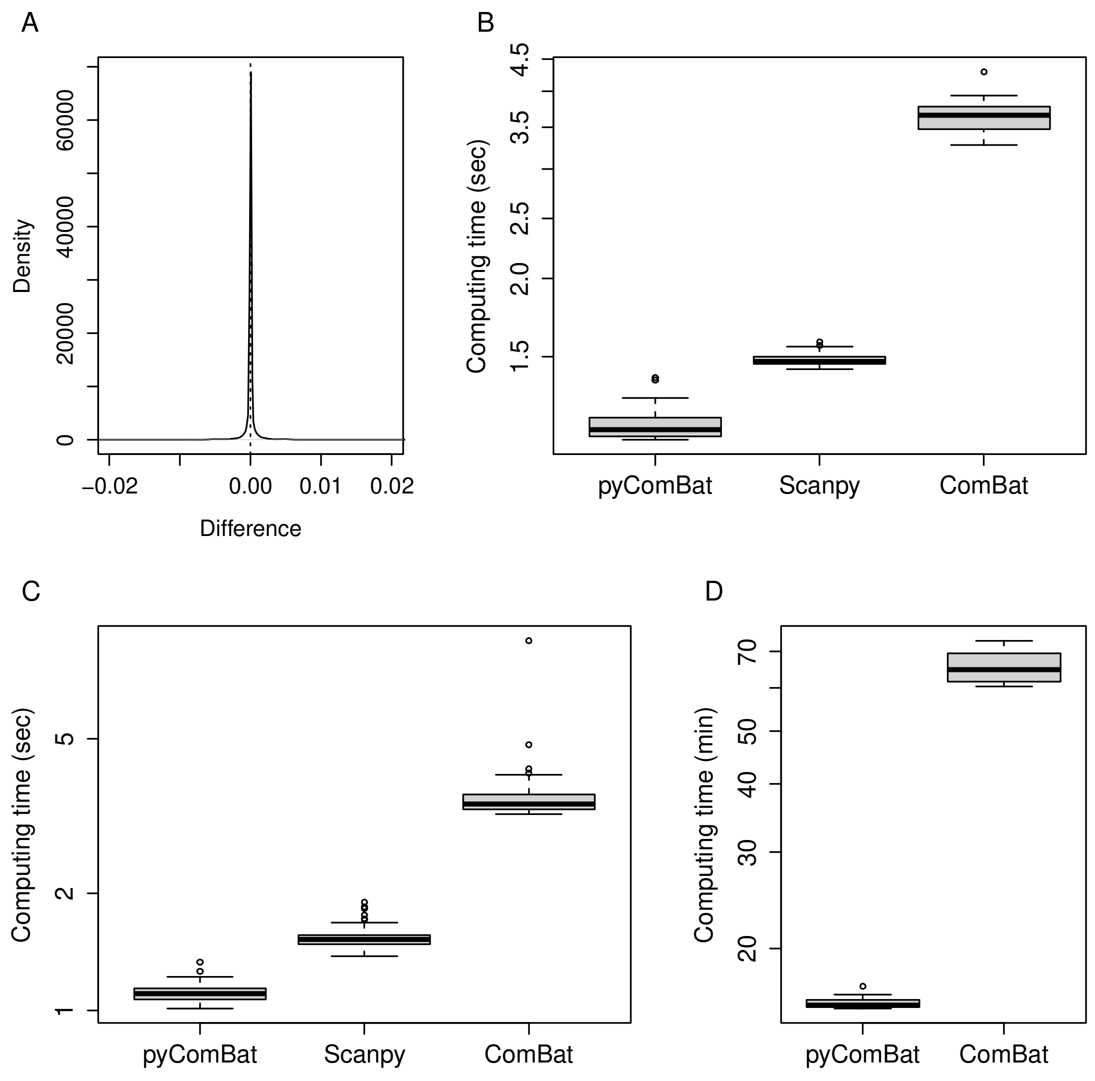
Yang,Y.H. *et al.* (2002) Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.*, **30**, 15e – 15.

Zhan,F. *et al.* (2007) Gene-expression signature of benign monoclonal gammopathy evident in multiple myeloma is linked to good prognosis. *Blood*, **109**, 1692–1700.

Zhang,Y. *et al.* (2020) ComBat-seq: batch effect adjustment for RNA-seq count data. *NAR Genomics Bioinforma.*, **2**.

Figures

Fig. 1



Legend of figures

Fig. 1

Performance of pyComBat *vs.* Combat *vs.* Scanpy's implementation of ComBat. **A** Distribution of the differences between the expression matrices corrected for batch effects, respectively by ComBat and pyComBat (parametric version), on the Ovarian Cancer dataset. The vertical dotted line corresponds to zero. **B** Computation time in seconds for pyComBat, Scanpy and ComBat for the parametric method, on the Multiple Myeloma dataset. The y-axis is in a log scale. **C** Computation time in seconds for pyComBat, Scanpy and ComBat for the parametric method, on the Ovarian Cancer dataset. The y-axis is in a log scale. **D** Computation time in minutes for pyComBat (left) and ComBat (right) for the non-parametric method, on the Ovarian Cancer dataset. The y-axis is in a log scale.

Tables

Table 1

Dataset	**Reference(s)**
Ovarian Cancer	
GSE18520	(Mok *et al.*, 2009)
GSE66957	
GSE69428	(Yamamoto *et al.*, 2016)
GSE9891	(Tothill *et al.*, 2008)
GSE26712	(Bonome *et al.*, 2008; Vathipadiekal *et al.*, 2015)
GSE38666	(Lili *et al.*, 2013; Huang *et al.*, 2018)
Multiple Myeloma	
GSE5900	(Zhan *et al.*, 2007; Driscoll *et al.*, 2010; Li *et al.*, 2021)
GSE66291	(Lionetti, Barbieri, Todoerti, Agnelli, Marzorati, *et al.*, 2015; Lionetti, Barbieri, Todoerti, Agnelli, Fabris, *et al.*, 2015)
GSE68891	
GSE122231	(Dhodapkar *et al.*, 2014; Khan *et al.*, 2015)

Legend of tables

Table 1
Composition of each meta-dataset used for benchmarking pyComBat, Scanpy's implementation of ComBat, and ComBat.