Bioinformatics, 37(16), 2021, 2365–2373
doi: 10.1093/bioinformatics/btab105
Advance Access Publication Date: 20 February 2021
Original Paper

Gene expression

CuBlock: a cross-platform normalization method for gene-expression microarrays

Valentin Junet [1,2], Judith Farrés[1], José M. Mas[1,*] and Xavier Daura [2,3,*]

[1]Anaxomics Biotech SL, Barcelona 08008, Spain, [2]Institute of Biotechnology and Biomedicine, Universitat Autònoma de Barcelona, Barcelona 08193, Spain and [3]Catalan Institution for Research and Advanced Studies (ICREA), Barcelona 08010, Spain

*To whom correspondence should be addressed.

Associate Editor: Jonathan Wren

Received on July 14, 2020; revised on February 4, 2021; editorial decision on February 10, 2021; accepted on February 16, 2021

Abstract

Motivation: Cross-(multi)platform normalization of gene-expression microarray data remains an unresolved issue. Despite the existence of several algorithms, they are either constrained by the need to normalize all samples of all platforms together, compromising scalability and reuse, by adherence to the platforms of a specific provider, or simply by poor performance. In addition, many of the methods presented in the literature have not been specifically tested against multi-platform data and/or other methods applicable in this context. Thus, we set out to develop a normalization algorithm appropriate for gene-expression studies based on multiple, potentially large microarray sets collected along multiple platforms and at different times, applicable in systematic studies aimed at extracting knowledge from the wealth of microarray data available in public repositories; for example, for the extraction of Real-World Data to complement data from Randomized Controlled Trials. Our main focus or criterion for performance was on the capacity of the algorithm to properly separate samples from different biological groups.

Results: We present CuBlock, an algorithm addressing this objective, together with a strategy to validate cross-platform normalization methods. To validate the algorithm and benchmark it against existing methods, we used two distinct datasets, one specifically generated for testing and standardization purposes and one from an actual experimental study. Using these datasets, we benchmarked CuBlock against ComBat (Johnson et al., 2007), UPC (Piccolo et al., 2013), YuGene (Lê Cao et al., 2014), DBNorm (Meng et al., 2017), Shambhala (Borisov et al., 2019) and a simple $\log_2$ transform as reference. We note that many other popular normalization methods are not applicable in this context. CuBlock was the only algorithm in this group that could always and clearly differentiate the underlying biological groups after mixing the data, from up to six different platforms in this study.

Availability and implementation: CuBlock can be downloaded from https://www.mathworks.com/matlabcentral/fileexchange/77882-cublock.

Contact: xouse@anaxomics.com or xavier.daura@uab.cat

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Since the first whole-genome microarray study of gene expression was published in 1997 (Lashkari et al., 1997; Schena et al., 1995), high-throughput gene-expression microarrays have been a standard in many experimental designs in biological and biomedical research. Although their use is being replaced by next-generation sequencing techniques such as RNA-Seq (Nagalakshmi et al., 2008), the large amounts of microarray data relevant to an equally large variety of biological and biomedical problems and available in public databases constitutes a valuable resource that will remain in use for many years. The potentiality of resources such as the Gene Expression Omnibus (GEO) as sources of Real-World Data (RWD)—data derived from a number of sources, outside the context of Randomized Controlled Trials (RCTs), and associated with outcomes in an heterogeneous patient population (Berger *et al.*, 2017)—may in fact boost the use of the wealth of available microarray data in the near future. The importance of RWD as a complementary information source in drug-evaluation studies is based on the observation that data from RCTs does not always match results from observational studies (Trotta, 2012), mostly owing to the limited number of RCT patients, their over-monitoring and the limited follow-up time. Thus, certain adverse drug reactions or lack-of-efficacy problems are hidden until RWD studies are performed, and drug administrations have come to encourage the extraction of information from sources complementary to RCTs to increase evidence around treatments (U.S. Food and Drug Administration, 2018; Sherman et al., 2017). A problem of RWD is that it tends to be highly heterogeneous, thus requiring careful analysis and statistical treatment (Bartlett *et al.*, 2019; Berger *et al.*, 2017). Translated to the context of this study, microarray data relevant to a particular problem will often originate from different laboratories and experiments, possibly using different microarray platforms (Bumgarner, 2013) and almost certainly obtained in different batches. In order to make a sensible use of such a heterogeneously sourced data, a data-normalization step is required before data analysis. Normalization can be relatively straightforward when dealing with different batches of a same experiment using the same biological samples, platform and operator, but gets increasingly complex as different operators, platforms and sample sources are introduced. This often leads to studies discarding part of the available data, which could otherwise be used to increase the chances of discovery of meaningful patterns or improve their statistics.

A main source for sample differences arising from systematic biases is the mixing of data from different microarray platforms. Unfortunately, most standard and widely used normalization methods are applicable to or have been developed for the single-microarray-platform context (Rudy and Valafar, 2011), making them generally inappropriate for the cross-study analysis of existing datasets. On the other hand, most existing cross-platform normalization methods, such as ComBat (Johnson *et al.*, 2007), XPN (Shabalin et al., 2008) or DWD (Benito *et al.*, 2004), require the data from different platforms to be normalized together—XPN and DWD were in fact developed for pairwise cross-platform normalization. For large datasets, normalizing platforms together can be restrictive. In addition to involving the normalization of a large joined dataset, the eventual addition of new microarray data requires global renormalization. This led to the more recent development of sample-wise, cross-platform normalization methods such as SCAN (Piccolo et al., 2012) and UPC (Piccolo *et al.*, 2013), YuGene (Lê Cao *et al.*, 2014), DBNorm (Meng *et al.*, 2017)—which can operate sample or platform wise—and Shambhala (Borisov *et al.*, 2019). SCAN performs a sample-wise normalization assuming a double Gaussian mixture distribution. It was, however, specifically designed for Affymetrix and two-channel Agilent platforms, thereby restricting its general use. Developed by the same authors, the Universal exPression Code (UPC) builds on SCAN to generate standardized estimates of expression that have a consistent interpretation across platforms, measuring how much the expression of the gene deviates from model-estimated background levels within the sample. Although, the derivation of UPCs is platform specific and, to our knowledge, it is currently available only for Affymetrix and Agilent microarrays through the package SCAN/UPC, the program offers also a generic UPC function applicable to any microarray platform, by making assumptions on the background and background-plus-signal distributions. The other distribution-based normalization method, DBNorm, scales the data distributions from the individual microarrays to a common form, which does not need to be predetermined (e.g. the distribution from a reference microarray). As a downside, it is very slow. On the other end, YuGene uses a simple transform that assigns a modified cumulative proportion value to each measurement, making the normalization very fast. Finally, Shambhala uses a harmonization method that transforms each profile so that it approaches the output of a chosen golden-standard platform.

Some methods like UPC, Shambhala and MatchMixeR (Zhang et al., 2020) have specifically included in their design the possibility to integrate data from both microarray and RNA-seq sources. As a matter of fact, any method that can be applied at the gene level could be adapted for such studies. It should be kept in mind, however, that this requires source-specific preprocessing steps taking into account the fundamental differences between these two types of data. In fact, the conceptual differences between microarray and RNA-seq gene-expression measurements are so significant that they may require distinct normalization procedures (Rapaport et al., 2013). In this study, we will solely focus on the application of normalization methods for the integration of gene-expression microarrays.

One should also note that the methods mentioned above were not necessarily developed with a same purpose. For example, ComBat was developed for the adjustment of batch effects and is often used in combination with other methods in cross-platform normalization procedures; although it does not normalize platforms separately, we introduced it in this study because of its broad use. Thus, the SCAN/UPC package offers the possibility to apply ComBat after SCAN normalization and summarization at gene-level—and before transformation to UPCs if so chosen. Thus, methods like ComBat are often called *integration* methods, in so that they integrate previously normalized data. Nevertheless, ComBat has become, on its own, a popular first choice for cross-platform normalization and a frequent benchmark standard for other methods (Irigoyen et al., 2018; Walsh et al., 2015). Shambhala, on the other hand, is classified as a *harmonization* method because it uses a golden standard as reference, while YuGene is a *transformation* and XPN, DBNorm and SCAN are referred to as *normalization* methods. In this study, like in Rudy and Valafar (2011), we will refer to all these methods as *cross-platform normalization methods* so far as they are being used in the literature to make data across different platforms comparable for the purpose of analysis.

Although the number of normalization methods proposed in the literature is large, to our knowledge there are no other major cross-platform normalization methods that can be applied to gene-expression microarrays in a platform agnostic way and that have been tested and validated as such. To enable systematic studies involving the download of microarray data from databases (possibly at different times) and its normalization and storage for later retrieval, allowing a non-linear use of the data—for example, in successive analyses incorporating different amounts of data as available or necessary, it is essential that a downloaded microarray set need not be normalized more than once. Here, we introduce a novel cross-platform normalization method fulfilling all these conditions. The algorithm is called CuBlock, which stands for Cubic approximation by Block. We validate its performance using various metrics and compare it to six methods that can be used in a cross-platform context, namely, the $\log_2$ transform of raw data, ComBat, YuGene, DBNorm, Shambhala and UPC. Overall, CuBlock shows the best performance in this group.

2 Materials and methods

In this section, we introduce the datasets used for the validation of CuBlock and describe the data preprocessing approach and the methods used for benchmarking and validation.

2.1 The datasets

We selected two benchmark datasets previously used in similar studies (Borisov *et al.*, 2019; Rudy and Valafar, 2011). The first set (here called the reference dataset) originates from projects MAQC (MAQC-I) (MAQC Consortium, 2006) and SEQC/MAQC-III (SEQC/MAQC-III Consortium, 2014), which made use of reference RNA samples to assess repeatability of gene-expression microarray data within a specific site, reproducibility across multiple sites and comparability across multiple platforms. The second set (here called the experimental dataset) originates from a study trying to assess profile differences of human spermatozoal transcripts from fertile and teratozoospermic males (Platts et al., 2007). The use of these two datasets allows us to assess, independently, effects from technical replicates (same biosample, analyzed in different labs with repetition) and biological replicates (different biosamples corresponding to a same condition).

2.1.1 The reference dataset

The data of this set are accessible in GEO with accession numbers GSE5350 (MAQC-I) and GSE56457 (MAQC-III), respectively. The set contains microarray gene-expression data corresponding to four titration pools from two distinct reference RNA samples: (*A*) Stratagene's Universal Human Reference RNA pool; (*B*) Ambion's Human Brain Reference RNA pool; (*C*) pool with an *A*:*B* ratio of 3:1; (*D*) pool with an *A*:*B* ratio of 1:3. These biosamples had been analyzed using different platforms and in different sites, as described (MAQC Consortium, 2006; SEQC/MAQC-III Consortium, 2014). Following the work from Rudy and Valafar (2011) and Borisov *et al.* (2019), we selected data from six of the platforms (between parentheses, data-set identifier in this study, GEO platform ID and project of origin):

- Affymetrix Human Genome U133 Plus 2.0 Array (AFX, GPL570, MAQC-I): three experiments (sites) (AFX_1 to AFX_3), with four biosamples (*A*–*D*) per experiment and five replicates per biosample (60 samples)
- Agilent-012391 Whole Human Genome Oligo Microarray G4112A (AG1, GPL1708, MAQC-I): three experiments (AG1_1 to AG1_3), with four biosamples (*A*–*D*) per experiment and five replicates per biosample (60 samples)
- Illumina Sentrix Human-6 Expression BeadChip (ILM, GPL2507, MAQC-I): three experiments (ILM_1 to ILM_3), with four biosamples (*A*–*D*) per experiment and five replicates per biosample (59 valid samples)
- Illumina HumanHT-12 V4.0 Expression Beadchip (HT12, GPL10558, MAQC-III): two experiments (ILM_COH and ILM_UTS), with four biosamples (*A*–*D*) per experiment and three replicates per biosample (24 samples)
- GeneChip® PrimeView^{TM} Human Gene Expression Array (PRV, GPL16043, MAQC-III): one experiment (AFX_USF_PRV), with four biosamples (*A*–*D*) and four replicates per biosample (16 samples)
- Affymetrix Human Gene 2.0 ST Array (HUG, GPL17930, MAQC-III): one experiment (AFX_USF_HUG), with four biosamples (*A*–*D*) and four replicates per biosample (16 samples)

Note that in the MAQC-I study the following microarrays from AG1 were discarded as outliers after the Agilent's Feature Extraction QC Report: AG1_1_A1, AG1_2_A3, AG1_2_D2, AG1_3_B3. Since the data for these microarrays is nevertheless deposited and we wanted our analysis to be as independent as possible of platform-dependent data-preprocessing steps, we considered also their inclusion. To this end, we evaluated the correlation of the data between all AG1 samples and observed that the 'outliers' are highly correlated to the non-outliers of the same experiment and of the other two experiments (about 0.97 in both cases). A dimension reduction of the raw data showed also no outliers. We therefore decided to include these four microarrays in the dataset.

2.1.2 The experimental dataset

This dataset contains spermatozoal RNA samples from normally fertile (*N*) and heterogeneously teratozoospermic (*T*) subjects and is accessible in GEO with accession number GSE6969. The samples had been analyzed on three different platforms (between parentheses, data-set identifier in this study and GEO platform ID):

- Affymetrix Human Genome U133 Plus 2.0 Array (AFF, GPL570): 13 independent biosamples of type *N* and 8 of type *T*
- Illumina Sentrix Human-6 Expression BeadChip (ILL1, GPL2507): five independent biosamples of type *N* and eight of type *T*. All ILL1 biosamples are replicates of AFF biosamples.
- Illumina Sentrix HumanRef-8 Expression BeadChip (ILL2, GPL2700): four independent biosamples of type *N* and 6 of type *T*

2.2 Data processing

To make the analysis as platform agnostic as possible, we took the image-processed raw intensities for all non-control probes and disregarded any platform-dependent background-signal correction such as that provided by mismatch probes in Affymetrix platforms. CEL files for Affymetrix and txt files for the other platforms were used. Probes with invalid intensities (NaN) in datasets HT12 and HUG were ignored. For Affymetrix microarrays, the intensities of probes constituting a probe set were averaged. We note that in the context of this study preprocessing of Affymetrix probe sets by simple averaging performed just as well as more complex treatments including background correction and RMA median polish (Irizarry et al., 2003), which was not completely unexpected—for example, Hubbell et al. (2002) found that simple averaging performed comparably to more robust approaches from the Affymetrix Micro Array Suite under low-noise conditions. From this point onward, the Affymetrix probe-set-average intensities were treated the same way as the raw probe intensities from the other platforms (the number of Affymetrix probe sets per gene being on the same order as the number of probes per gene in other platforms). Note that we could have skipped the probe-set-averaging step and worked directly with all Affymetrix probes. While we tried this, it increased significantly the computational cost of the normalization procedure (due to the several-fold increase in number of probes) at a marginal gain. At this point, probe-set average intensities (Affymetrix) and raw probe intensities (other platforms) were $\log_2$ transformed—we also tried applying quantile normalization in the preprocessing without it improving significantly the results. CuBlock normalization was then applied for each platform separately. Since probes vary among the different microarray platforms, the normalized datasets were then transformed from the probe level to the protein level by mapping probes to UniProtKB accession numbers (ACs) and keeping only those probes that map to an AC present in all platforms. The mapping to proteins was performed by taking the gene identifiers from the GEO tables containing the microarray data. Each selected AC was then assigned an intensity equal to the average of the normalized intensities of associated probes in the given microarray. The choice of UniProt ACs, rather than gene identifiers, was made to facilitate streamlining with protein-level post-normalization analysis in studies where microarray data is used to infer protein expression (i.e. probes matching CDS regions). We note, however, that CuBlock delivers normalized data at the probe level, meaning that the user is free to summarize the data at the gene level if appropriate, and that potential information regarding, for example, probes matching non-coding exon regions, will remain available.

To benchmark CuBlock against established normalization methods applicable in a generic cross-platform context, we compared it to a simple $\log_2$ transform and to the methods ComBat (Johnson *et al.*, 2007), YuGene (Lê Cao *et al.*, 2014), DBNorm (Meng *et al.*, 2017), Shambhala (Borisov *et al.*, 2019) and UPC (Piccolo *et al.*, 2013). YuGene, DBNorm and UPC were applied following the same procedure used for CuBlock, i.e. normalization of the $\log_2$ transform of the probe intensities and successive mapping to ACs. ComBat requires all microarrays to be normalized together, which implies their merging before normalization. Therefore, in this case the mapping to UniProtKB ACs and selection of ACs present in the different platforms was performed after $\log_2$ transform and before ComBat normalization. We note that DBNorm allows normalization per sample and per platform. We performed both, but show only the results obtained with sample-wise normalization since they are better. Comparison to Shambhala was done only for the datasets AFX, AG1 and ILM from the reference dataset, since Shambhala-normalized data for these sets has been already reported by the authors as supplementary data to Borisov *et al.* (2019). DBNorm was only used on the experimental dataset, as the calculations turned out to be forbiddingly slow. To perform the calculations we used the R package *sva* for ComBat (https://bioconductor.org/packages/release/bioc/html/sva.html) and the packages provided by YuGene (https://cran.r-project.org/web/packages/YuGene/index.html) and DBNorm (https://github.com/mengqinxue/dbnorm) authors in the respective papers. Calculations with these programs were performed with default settings. For DBNorm, in order to reproduce the general case (e.g. this study), in which a reference microarray cannot be straightforwardly selected, we used the option of normalization into a normal distribution. For UPC (https://www.bioconductor.org/packages/release/bioc/html/SCAN.UPC.html), we used its generic function for expression set, which takes expression × samples data, with default parameters from the package.

2.3 Comparison and validation methods

To validate and compare the cross-platform normalization methods evaluated in this study we used the methodology described below. The objective was to increase the sensitivity, i.e. the identification of true biological differences, while minimizing platform and various kinds of replica effects. All validation methods were applied on a subset of 500 proteins that best distinguish two given biological groups.

To select the 500 proteins we first performed a differential analysis on the normalized data, for all platforms in the reference or experimental dataset. To this end, we performed Welch's t-test to evaluate, for each protein, the difference between the associated mean intensities in units of uncertainty (the t-statistic) in two biological groups, A and B from the reference dataset (total of 16 624 proteins) or N and T from the experimental dataset (total of 16 937 proteins). Note that we deliberately avoid considerations on whether the datasets meet the requirements of the t-test, since we used the test simply to identify the 500 proteins with largest separation of group means per uncertainty unit, that is, with lowest associated P-values, irrespective of the error in the P-value and, therefore, of its valid interpretation as a probability. Although for such purpose we do not require the calculation of FDR-adjusted P-values (q-values) (Storey, 2002), since they conserve P-value ranking, we did obtain them and show corresponding ROC-like curves (the cumulative distribution function of the q-values) in Supplementary Figure S1 (Supplementary Information). We decided to select a fixed number of proteins, rather than proteins with a P- or q-value below a given arbitrary threshold, to enable the comparison of methods using datasets of equal and reasonably large dimensionality. We note, nevertheless, that the 500-protein cut corresponds to an FDR well below 10^{-2} (Supplementary Fig. S1). The differential analysis was performed with the MATLAB function *mattest*.

2.3.1 Silhouette plot

Silhouette plots are graphical displays of data partitions (Rousseeuw, 1987), where clusters are represented by so-called *silhouettes* generated by comparison of cluster tightness and separation. The method assigns a silhouette value between -1 and 1 to each element of a cluster, indicating if the element is well clustered (value close to 1), lies between two or more clusters (close to 0) or is likely misclassified (close to -1). The silhouette plot is then generated by representing the values for all elements as bars, for the different cluster partitions. We computed three silhouette plots for the reference dataset: one identifying clusters with platforms, one where the data was assigned to groups $A \cup C$ and $B \cup D$ and one where the partitioning was represented by sets A, B, C and D. For the experimental dataset, silhouette plots based on platform partitioning and T versus N partitioning were computed. The MATLAB function *silhouette* was used to compute the silhouette plots.

2.3.2 *t*-SNE dimension reduction

t-SNE (Maaten and Hinton, 2008) is a stochastic dimension-reduction method aimed to preserve the local structure of data (keeping the low-dimensional representation of very similar data points close together) while retaining essential traits of global structure. It analyzes the neighborhood of the data points by calculating pairwise conditional probabilities representing their similarity. The method then tries to find a low-dimensional representation that minimizes the difference between the high-dimensional and low-dimensional conditional probabilities. The parameter controlling the number of neighbors is called perplexity, and is typically given values between 5 and 50. Due to its stochastic nature and the dependence on the chosen perplexity parameter, the algorithm may converge to irrelevant solutions. We thus performed 10 runs for each of a number of perplexity values and selected the one producing the most consistent biological partitioning according to the average silhouette values. For the reference dataset we used perplexity values from 5 to 50, in increments of 5, and selected the representation giving the best clustering relative to sets A, B, C and D. For the experimental dataset, we used perplexity values 5, 10 and 15 and selected the representation giving the best clustering relative to sets T and N. The MATLAB function *tsne* was used to perform the t-SNE dimension reduction.

2.3.3 Dendrogram

We performed a hierarchical clustering analysis using the Euclidean distance as metric and the arithmetic mean as linkage criterion, and represented the resulting cluster hierarchy as a dendrogram. To assess the significance of the clusters, we applied multiscale bootstrap resampling as provided in the R package *pvclust* (Suzuki and Shimodaira, 2006). By default, this package considers 10 relative bootstrap sample sizes (bootstrap sample size divided by total sample size), from 0.5 to 1.4, with 1000 resamplings per sample size, leading to a total of 10 000 bootstrap resamples. The package provides two statistics to estimate the significance of the obtained clusters: the bootstrap probability (BP) or frequency (expressed as percentage) of observation of a given cluster in the bootstrap resamples, and the approximately unbiased P-value (AU), an unbiased version of BP. More details on multiscale bootstrap resampling can be found in Shimodaira (2004). We plot the dendrograms using the R package *dendextend*.

2.3.4 SVM classification

The goal of this analysis was to assert whether relevant patterns can be found using the data from only one platform. Support vector machines (SVM) (Cortes and Vapnik, 1995) are binary classifiers applicable to problems that are reducible to a binary outcome, such as the T and N phenotypes in our experimental dataset. We trained a linear support vector machine model for each platform using the following approach. To reduce feature-vector dimensionality, where dimensions are proteins (more specifically their microarray-derived intensities), while retaining the capacity to asses how well the 500 proteins separate the T and N populations, the training was performed six times, starting with dimension 5 and increasing it up to dimension 10. For each of 1000 runs with a given dimensionality, we selected randomly from the 500 protein set as many proteins as dimensions, extracted the corresponding data from sets T and N, trained a linear SVM model for each platform, separately, and tested it on the other two platforms. This led to a total of 6000 models per platform. For each platform, we calculated the mean and standard deviation of different classification scores over the 6000 models, namely, Accuracy, Matthews Correlation Coefficient (MCC) (Boughorbel et al., 2017), Balanced Accuracy and Area Under the ROC Curve (AUC) (Fawcett, 2006). The MATLAB function *svm* was used to train the SVM models.

3 Algorithm

The CuBlock algorithm relies on the simple and widely used assumption that most genes are neither over- nor under-expressed (Yang et al., 2002). Thus, a transformation following the cubic polynomial x^3 will leave most of the genes around 0 and slowly differentiate the extremes, i.e. the under- and over-expressed genes. The complementary idea in CuBlock is the use of a block-wise transformation, which had been already implemented successfully in XPN (Shabalin *et al.*, 2008). To this end, CuBlock partitions probes into clusters and, for each sample and probe cluster (i.e. for each data block), transforms the data by a procedure that involves its mapping to objective values between -1 and 1 (with density increasing toward 0) and the fitting of a cubic polynomial to the resulting

```
CuBlock($X, k, N$)
$X$ is the log2 transform of micro array, columns are samples and rows are probes.
$k$ is the number of clusters.
$N$ is the number of repetitions.
The algorithm calls 2 other algorithms: GetTargetValues returns the target values
for a polynomial fitting. ModPol modifies the fitted polynomial to make it increasing.
mean, resp. std, returns the mean value, resp. standard deviation, of an array.
$[a:b]$ is the set of natural numbers between $a$ and $b$.
$X[\cdot,\cdot]$ access the element(s) in bracket.

    $\hat{X} \leftarrow NaN$ of same size as $X$
    count $\leftarrow 0$ of same size as $X$
    $m \leftarrow$ number of samples
    REPEAT
      $C \leftarrow k$-means clustering partition of the probes of $X$ into $k$ clusters
      FOR1 $i \in [1:m]$
        FOR2 $j \in [1:k]$
          IF there are more than 100 probes in cluster $j$
            $B \leftarrow X[j^{th}$ Cluster of $C, i^{th}$ Sample$]$
            $B \leftarrow \frac{B-\mathrm{mean}(B)}{\mathrm{std}(B)}$
            $BS \leftarrow$ sort $B$ in ascending order
            $D \leftarrow$ GetTargetValues($BS$)
            $P \leftarrow$ get the coefficients for a cubical polynomial
                 such that its evaluation at $BS$ best fits the target values $D$
            $\hat{B} \leftarrow$ ModPol($B, P$)
            $\hat{X}[j^{th}$ Cluster of C, $i^{th}$ Sample$] \leftarrow \hat{X}[j^{th}$ Cluster of C, $i^{th}$ Sample$] + \hat{B}$
            count$[j^{th}$ Cluster of C, $i^{th}$ Sample$] \leftarrow$ count$[j^{th}$ Cluster of C, $i^{th}$ Sample$] + 1$
          END IF
        END FOR2
      END FOR1
    UNTIL $N$ repetitions
    $\hat{X} \leftarrow$ elementwise division of $\hat{X}$ by count
    END FUNCTION
```

Fig. 1. Pseudocode describing the CuBlock algorithm (see description in Section 3)

distribution (see below). By using data blocks, different profiles present (mixed) in the full dataset are considered, and as many different cubic polynomials are fitted to them, underlying different shapes contained within the original distribution. A pseudo code of the CuBlock algorithm is described in Figure 1. It calls two additional algorithms with pseudo codes provided in Supplementary Figures S2 and S3 (Supplementary Information).

The input to CuBlock is a matrix X containing the $\log_2$ transform of the gene-expression microarray intensities, where columns are samples and rows are probes. As discussed in Section 2.2 it is up to the user to decide any level of preprocessing of the input $\log_2$-transformed intensities, for example, probe-set summarization for Affymetrix microarrays.

CuBlock makes use of the k-means clustering algorithm (Lloyd, 1982) to partition probes in the space defined by the samples—a probe data point is a vector of probe intensities of dimension equal to the number of samples—and is applied per platform, i.e. the k-means clustering is performed for all samples of a given platform. k-means is an iterative algorithm that tries to partition the data into a predefined number k of non-overlapping clusters, starting from a random initialization of their centroids. Because of the random initialization, clusters from different runs may differ and the core part of the CuBlock algorithm is repeated several times for different solutions of k-means. Thus, input parameters k and N in Figure 1 refer to the chosen number of k-means clusters and repetitions, which in this work took values of 5 and 30, respectively. The number of clusters was chosen to be low enough that the k-means algorithm will not, for some partitions, converge always to the same solution and high enough that blocks with different distributions will be obtained. The CuBlock algorithm finds first a probe-cluster partition in the space defined by the samples and then applies its normalization scheme to data blocks defined as those ($\log_2$) probe-intensity values from a sample that belong to a given cluster. Therefore, for k clusters and m samples we have a total of $k \cdot m$ blocks. The advantage of the normalization by block is that it decomposes the distribution of probe intensities of a sample into its different block distributions, according to similarities between probes found by the clustering algorithm in the space of all samples. These different distributions will enable the emergence of different patterns present in the data. Instead, if the blocks were selected at random or the whole sample was used, the normalization method would estimate parameters based on a unique distribution, masking these different patterns.

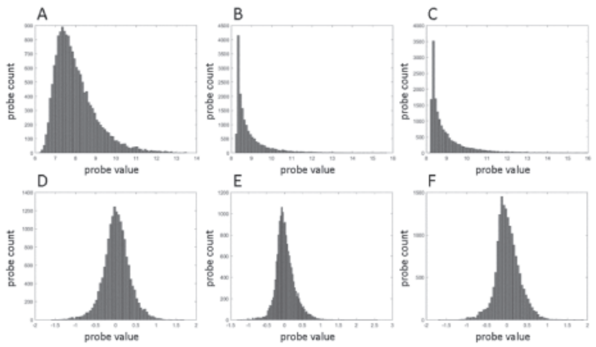

Fig. 2. Example histograms for samples from different platforms. (**A–C**) Histograms before CuBlock normalization and after $\log_2$ transform. (**D–F**) Histograms after CuBlock normalization. A, D: biosample *A*, platform AFX; B, E: biosample *B*, platform AG1; C, F: biosample *A*, platform AG1

Although we initially determine the probe clusters using all samples, we then normalize sample by sample to reduce the dependence of the normalization on the full sample collection. The strategy of normalization by block is similar to that used by the cross-platform normalization method XPN (Shabalin *et al.*, 2008). However, XPN defines probe clusters and sample clusters with two independent application of k-means (one on the input matrix and the other on its transpose) and blocks are then constituted by all possible combinations of one sample cluster with one probe cluster.

For each block, and each of the N repetitions of the k-means clustering, CuBlock fits a cubic polynomial to a mapped set of points symmetrically distributed between -1 and 1 with density increasing toward zero. This is performed in four steps, as shown in Figure 1. First, the block data is linearly transformed to z-scores (zero mean and unit standard deviation). These are then used as input values of a mapping function whose output values will be used to fit the cubic polynomial, as described in the pseudocode shown in Supplementary Figure S2. The mapping associates the sorted values present in the block to an equal number of equidistant points between -1 and 1, and takes these new points to an uneven power in order to have their distance decrease as they approach zero from either side (Supplementary Fig. S4). The exact uneven power will determine how slow is the growth of the points around zero, and is selected such that, on average, the values of the block that are within standard deviation, i.e. the block values between -1 and 1, are mapped to a value smaller than 0.1 (Supplementary Fig. S5). The algorithm tries uneven powers between 3 and 21 and the first one that fulfills the criterion is selected. Next, the algorithm finds the coefficients of a cubic polynomial that, when evaluated on the sorted block data (input values), best fits the output values from the mapping function (Supplementary Fig. S4). We chose to fit a cubic polynomial instead of a higher degree one to avoid overfitting. Polynomial coefficients were obtained with the MATLAB function *polyfit*, with degree 3.

If the block data is not symmetric or contains many outliers, a cubic polynomial will produce a poor fit. Thus, the polynomial will increase along the symmetric part of the block and decrease as it reaches the outliers (Supplementary Fig. S6). Despite leading to a poor fit, this feature can be used to identify asymmetry issues and outliers. When this is the case and decreasing values are identified after evaluating the polynomial on the block data, the decreasing values are corrected in order to preserve data sorting upon normalization. Roughly, the correction equates the decreasing values to the last increasing value (after an increasing section) or to the last decreasing value (before an increasing section). The precise corrections are described in Supplementary Figure S6, and the cases where the cubic polynomial might decrease are considered in Supplementary Figure S3.

The output of CuBlock is a matrix of normalized gene-expression values, where columns are samples and rows are probes. As discussed in Section 2.2 it is up to the user to decide at which level and using which database codes for the mapping to that level, the probe values should be summarized.

Figure 2 shows the histograms of different samples before and after normalization. While before normalization the samples follow clearly different distributions, after normalization the distributions are much more homogeneous. We note that before normalization the distributions are clearly platform dependent (compare A and C, which correspond to the same biosample but different platforms, and B and C, which correspond to different biosamples and the same platform). This effect is remarkably corrected after normalization.

4 Results and discussion

The algorithm described in the previous section was applied to the data introduced in Section 2.1 after preprocessing (see Section 2.2), and the results were compared to those obtained with other normalization methods as explained in Section 2.3. In line with the objectives stated in the introduction, the discussion of the results evolves around the ability of the methods to highlight biological patterns in a multi-platform context. We want to note, however, that although CuBlock was not developed for single-platform normalization—this being already well covered by other methods, it can be also used for this purpose. Thus, as a side example, Supplementary Figure S7 shows results for a third dataset, available in GEO with accession number GSE65212 and involving five biological groups (Maire et al., 2013a,b; Maubant et al., 2015), demonstrating that results from CuBlock are very consistent with those from RMA normalization.

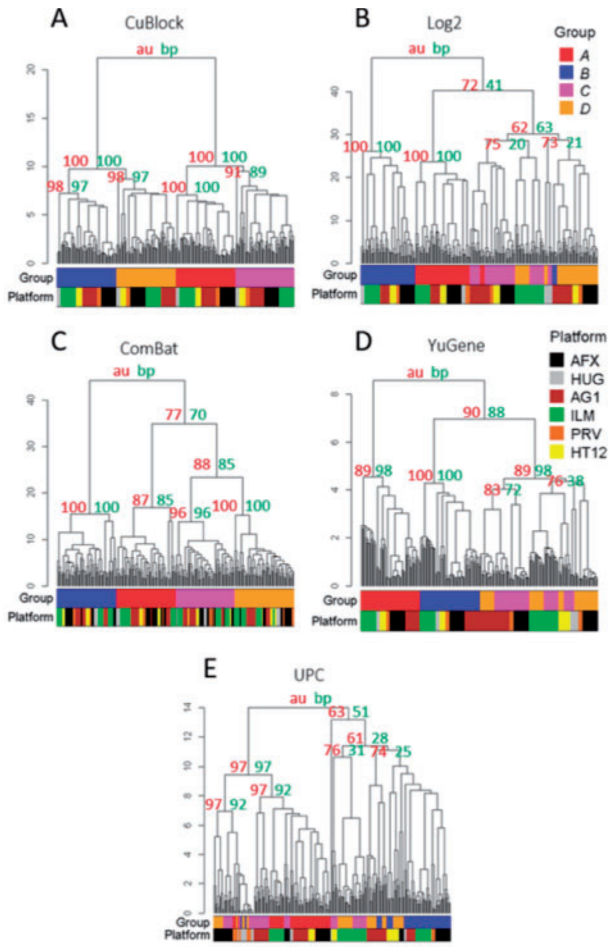

Fig. 3. Dendrogram analysis of the reference dataset (six platforms) after normalization with CuBlock (**A**), $\log_2$ (**B**), ComBat (**C**), YuGene (**D**) and UPC (**E**). Color bars under the dendrograms indicate the biological group and platform corresponding to each leaf; the BP (green) and AU (red) values (see Section 2.3.3) for some selected clusters are indicated at the origin of the branches

4.1 Reference dataset: six platforms

Figure 3 and Supplementary Figures S8–S11 show the results obtained with the different normalization methods using the dendrogram, silhouette and *t*-SNE analyzes, respectively. The three validation methods show that CuBlock and ComBat separate very clearly the biological groups *A*, *B*, *C* and *D* (except for a couple of *A* points in ComBat's case). ComBat tends to produce tighter but less cleanly separated clusters for these four groups, as illustrated by both the *t*-SNE (Supplementary Fig. S9C) and silhouette (Supplementary Fig. S8C) plots. CuBlock is the only method that clusters the biological groups *A* and *C*, and *B* and *D* together in the dendrogram plot (Fig. 3A), and this is also underlined in the corresponding silhouette plot in Supplementary Figure S10A, showing high and homogeneous silhouette values. On the contrary, $\log_2$, ComBat, YuGene and to a smaller extent UPC tend to cluster *C* with *D* (Fig. 3B–E). In fact, $\log_2$ and YuGene have difficulties to separate these two groups at all, while UPC has serious difficulties to separate the groups *C* and *D* from the parent groups (*A* and *B*; see also Supplementary Fig. S9). Supplementary Figures S10 and S11 show silhouette plots using the groups $A \cup C$ and $B \cup D$ and the platforms as given clusters, respectively. We note that even though CuBlock is shown to emphasize the biological differences and Supplementary Figure S11A indicates weak platform clusters, both the *t*-SNE (Supplementary Fig. S9A) and dendrogram (Fig. 3A) plots show that, within each of the *A*, *B*, *C*, *D* clusters, the samples are subclustered by platform. As can be seen in these Figures, ComBat mixes the data from the different platforms best, while YuGene, UPC and $\log_2$ are, approximately in this order, worst at mixing platform data.

In Figure 3 and throughout this study, the $\log_2$ transform plays the role of control method. As a second potential control, we also used a common approach consisting in platform-specific normalization of the samples followed by a centering transformation. Supplementary Figure S12 shows the dendrogram plot of the RMA method (background correction, quantile normalization, summarization at the probe-set level for Affymetrix platforms and $\log_2$ transformation) followed by Z-score transformation (subtracting the sample's mean and dividing by its standard deviation). It can be observed that the results are only slightly better than those from a $\log_2$ transformation, for which reason we kept the latter as the simplest approach.

4.2 Reference dataset: three platforms

To compare the results from CuBlock and Shambhala [the latter reported by Borisov *et al.* (2019) for the same dataset], we also performed the analysis for the three-platform subset used by the authors of Shambhala, namely AFX, ILM and AG1. They had concluded that Shambhala separates well $A \cup C$ from $B \cup D$ but not *A* from *C* or *B* from *D*. Using our selection of 500 proteins that best distinguish *A* from *B*, when looking at the results for Shambhala in Supplementary Figure S13B, D we observe that, while $A \cup C$ forms a relatively clear cluster, all the $B \cup D$ points from AG1 samples are clustered with $A \cup C$, making $B \cup D$ a well defined cluster only for AFX and ILM. As illustrated by the *t*-SNE and dendrogram plots and by the negative silhouette values in Supplementary Figure S13F, Shambhala does also not distinguish *A*, *B*, *C* and *D* from each other well. The results for CuBlock in Supplementary Figure S13A, C, E show the same features already discussed in Section 4.1 using the data for six platforms.

4.3 Experimental dataset

Figure 4 and Supplementary Figures S14–S16 show the results obtained for the human sperm dataset, after normalization with CuBlock, $\log_2$, ComBat, YuGene, DBNorm and UPC. CuBlock is the only normalization method that significantly distinguishes the two biological groups, *T* and *N*. The dendrogram plots in Figure 4

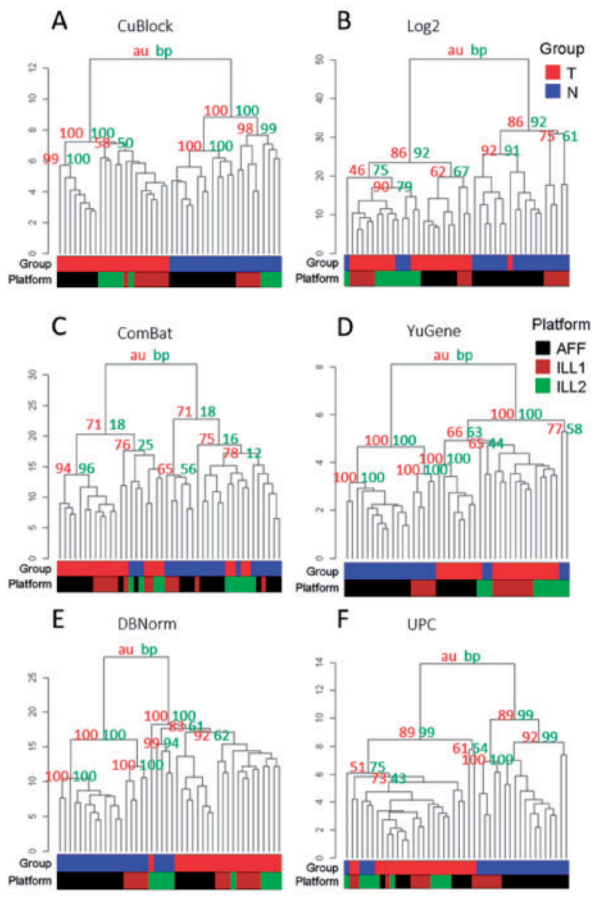

Fig. 4. Dendrogram analysis of the experimental dataset after normalization with CuBlock (**A**), $\log_2$ (**B**), ComBat (**C**), YuGene (**D**), DBNorm (**E**) and UPC (**F**). Color bars under the dendrograms indicate the biological group and platform corresponding to each leaf; the BP (green) and AU (red) values (see Section 2.3.3) for some selected clusters are indicated at the origin of the branches

and *t*-SNE plots in Supplementary Figure S15 show that ComBat has troubles to establish a clear boundary between the two groups, particularly for the ILL2 platform, while the other methods tend to misclassify the ILL2 samples corresponding to the *N* group. Similarly to the results for the reference dataset (Section 4.1), CuBlock tends to sort the samples by platform within the clusters *T* and *N* (except for one *T* sample from ILL2). To investigate whether patterns that are found using one platform can be extrapolated to the other platforms, we performed a SVM classification test as described in Section 2.3.4. The results are shown in Table 1. In all cases, CuBlock outperforms the other methods. It is also worth noting that no matter which platform is used for the training based on CuBlock, the results are always very similar. To a lesser extent, this is also true for DBNorm. However, using $\log_2$, ComBat, YuGene and UPC, training with ILL2 gives worse results than with the other platforms, probably due to the fact that this platform constitutes a better defined cluster, as shown in the silhouette plots in Supplementary Figure S16.

In Figure 4, it can be seen that ComBat clusters more strongly AFF and ILL1 (which contains replicates of AFF) than CuBlock does. In Supplementary Table S1 we provide, for the six normalization methods, average pairwise Kendall rank correlations between the replicates shared by the platforms AFF and ILL1, between the non-replicates of these two platforms and between the other two platform pairs, as well as a discussion on the correlation data. In summary, we show that methods like ComBat will tend to correlate samples from different platforms, even when they are not from the same biological group (and should be therefore less correlated), while methods like CuBlock tend to decorrelate the different platforms to highlight existing patterns in the samples. In this sense, CuBlock will never lead to a high correlation between replicates (which might be desirable in some studies), but will on the other hand enable comparison across platforms using a selected small set of differentially expressed proteins.

4.4 Reference dataset: missing biological groups

CuBlock is a method that works with the actual distribution of the data, without making any assumption on its shape. It is, in that sense, dependent on the actual differences found in the input data at the platform level—which are in turn highlighted by the block treatment enabling the uncovering of the different distributions present in the data. To test this dependence, we analyzed again the reference dataset using CuBlock and ComBat while removing the groups *B* and *D* from the platforms HUG and AG1. In other words, these two platforms were normalized only with *A* and *C* samples. The biological difference between *A* and *C* is that 25% of *C* is made of *B* RNA samples. The other four platforms were normalized with all four biological groups. As it can be seen in Figure 5A, C, CuBlock results in the clustering of the *C* samples of HUG and AG1 separately and closer to the *D* cluster of the other platforms than to their *C* cluster. However, the *A* cluster remains a well-defined cluster for all platforms. This suggests that, in the two platforms with missing groups, CuBlock emphasizes the difference between the available data, as predicted above. For HUG and AG1, this means emphasizing the differences between *A* and *C* (the only groups it sees), thus bringing *C* closer to the *D* cluster formed by the other platforms, since, as *C* itself, *D* is also a mixture of *A* and *B*. ComBat does however do similarly in this regard (Fig. 5B, D), with the *C* samples of HUG and AG1 being even more mixed with the *D* samples, and the *A* samples of the two platforms getting closer (see t-SNE plot) to the *C* samples of the four other platforms, some of them ending up clustered (see dendrogram) in this group.

5 Conclusion

We have introduced an algorithm for cross-platform normalization of gene-expression microarray data as well as a strategy to validate cross-platform normalization methods, with a focus on the capacity of the algorithm to properly separate samples from different biological groups after normalization and across multiple platforms. Overall, CuBlock showed good results on the two datasets used in this evaluation, a dataset specifically generated for testing and standardization purposes and a dataset from an actual experimental study. CuBlock could always differentiate, clearly, the underlying biological groups after mixing data from up to 6 different platforms. Nevertheless, we observed that within each biological group the algorithm tends to subcluster samples by platform, indicating a remaining, yet comparatively small, platform effect. The ComBat algorithm (Johnson *et al.*, 2007) showed also good performance on the reference dataset, with better mixing of data from different platforms than all the other methods tested. However, on the experimental dataset, where samples are from different individuals and the difference between biological groups might become less obvious than in the reference dataset, ComBat did not perform as well. Platform mixing was still good but the distinction between the two biological groups was not clear. To rationalize the differences in platform-mixing properties between CuBlock and ComBat, we performed a rank correlation analysis and show that methods normalizing all platforms together, like ComBat, tend to correlate the data from the different platforms, as opposed to methods normalizing platforms separately, which tend to decorrelate them. While high correlation among samples from different platforms might be desirable in some set-ups, there is no guarantee that this correlation will be meaningful, as shown for ComBat, which in our test induced correlation among platforms almost independently of the actual level of expected correlation between them. As already mentioned, ComBat also requires the platforms to be normalized together, making it a less convenient method for systematic application to multiple datasets. On the other hand, DBNorm—used only with the experimental

Table 1. SVM classification scores (see Section 2.3.4)

Training platform	Accuracy	MCC	Balanced accuracy	AUC
CuBlock				
AFF	0.88 ± 0.08	0.76 ± 0.17	0.86 ± 0.10	0.98 ± 0.03
ILL1	0.92 ± 0.06	0.85 ± 0.12	0.91 ± 0.07	0.99 ± 0.02
ILL2	0.86 ± 0.16	0.74 ± 0.32	0.86 ± 0.17	0.99 ± 0.04
$\log_2$				
AFF	0.77 ± 0.07	0.52 ± 0.16	0.71 ± 0.08	0.70 ± 0.11
ILL1	0.80 ± 0.06	0.64 ± 0.11	0.81 ± 0.06	0.81 ± 0.04
ILL2	0.44 ± 0.16	−0.13 ± 0.35	0.46 ± 0.16	0.24 ± 0.31
ComBat				
AFF	0.73 ± 0.11	0.53 ± 0.20	0.76 ± 0.10	0.90 ± 0.08
ILL1	0.76 ± 0.08	0.57 ± 0.15	0.77 ± 0.08	0.90 ± 0.06
ILL2	0.61 ± 0.30	0.22 ± 0.61	0.61 ± 0.30	0.63 ± 0.36
YuGene				
AFF	0.81 ± 0.07	0.63 ± 0.15	0.77 ± 0.08	0.91 ± 0.06
ILL1	0.87 ± 0.08	0.74 ± 0.16	0.87 ± 0.09	0.96 ± 0.04
ILL2	0.55 ± 0.07	0.07 ± 0.17	0.53 ± 0.07	0.97 ± 0.11
DBNorm				
AFF	0.81 ± 0.07	0.62 ± 0.16	0.77 ± 0.09	0.91 ± 0.07
ILL1	0.86 ± 0.06	0.76 ± 0.09	0.87 ± 0.05	0.95 ± 0.04
ILL2	0.79 ± 0.13	0.61 ± 0.24	0.77 ± 0.13	0.96 ± 0.11
UPC				
AFF	0.74 ± 0.08	0.45 ± 0.22	0.68 ± 0.10	0.82 ± 0.08
ILL1	0.82 ± 0.06	0.68 ± 0.09	0.83 ± 0.05	0.90 ± 0.05
ILL2	0.57 ± 0.10	0.10 ± 0.24	0.54 ± 0.11	0.75 ± 0.34

Note: Mean and standard deviation over the 6000 models per platform and method.

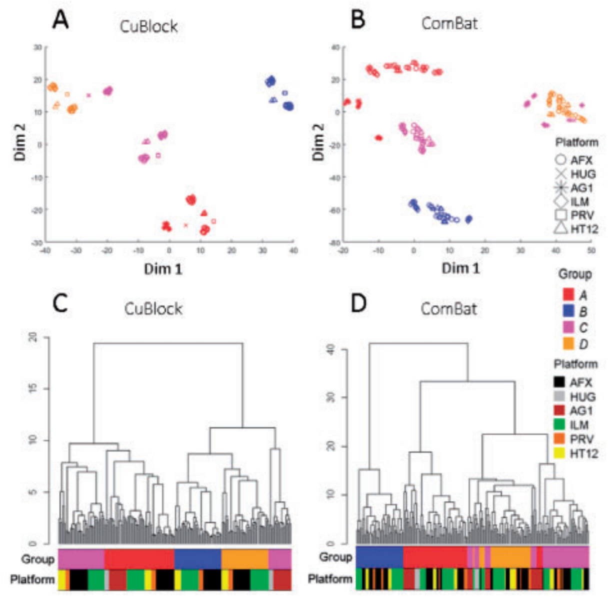

Fig. 5. *t*-SNE dimension reduction and dendrogram plots for the reference dataset, with exclusion of the *B* and *D* samples from platforms HUG and AG1, after normalization with CuBlock and ComBat. Point color and shape indicate biological group and platform, respectively (right-hand legend). (**A**) *t*-SNE for CuBlock normalized data; perplexity (Prp) and mean silhouette index (SI) values (see Section 2.3.2): Prp = 15, SI = 0.80. (**B**) Corresponding analysis for ComBat-normalized data; Prp = 10, SI = 0.67. (**C, D**) Dendrograms for CuBlock (C) and ComBat (D) normalized data; color bars under the dendrograms indicate the biological group and platform corresponding to each leaf

dataset, as it proved computationally much more time demanding than the rest, YuGene and UPC performed only slightly better than $\log_2$. For the set evaluated, Shambhala lagged clearly behind the other methods, arguably including a simple $\log_2$ transformation. Finally, we also showed, by training SVMs with single-platform data from the experimental set, that when normalizing the data with CuBlock the patterns that are found using one platform can be extrapolated to the other platforms significantly better than when the normalization is done with any other of the methods.

Because no assumptions are made on the distributions underlying the input data, CuBlock can be thought of as a transformation which result remains close to the input. CuBlock fits cubic polynomials to data blocks that are found by *k*-means clustering, thereby trying to best fit the different distributions found in the data corresponding to a sample (different blocks need not have the same distribution) and it does so without assuming a shape for these distributions. As a consequence, CuBlock emphasizes the differences within the input data at the platform level, making it sample-composition dependent despite the only step in the algorithm where the microarray samples from a given platform are considered together is when applying the *k*-means algorithm (afterwards, each sample is considered separately). This sample dependence was highlighted in this study when analyzing the reference dataset after removing biological groups in some platforms. Nevertheless, the sample dependence is even more prominent for algorithms normalizing platforms together, such as ComBat.

We note that, while in this study the results were presented at the protein level, CuBlock is applied and returns the normalized data at the probe level, as an appropriate level for gene-expression microarrays. In fact, like most other methods, it could technically be applied at any expression × samples level. Studying the effect of the application of CuBlock and other cross-platform normalization methods at different levels with appropriate (and possibly study-specific) preprocessing steps could be an interesting development which would ideally lead to an appropriate systematic integration of RNA-seq with gene expression microarrays. This however goes beyond the scope of this study.

In summary, we have shown that CuBlock can be applied to data from multiple microarrays in a platform agnostic way and preserves the biological grouping of the samples, demonstrating a good performance for different types of samples. It is therefore a tool appropriate for gene-expression studies based on multiple microarray sets collected along multiple platforms and at different times, thus facilitating the extraction of knowledge from the wealth of microarray data available in public repositories and enabling the use of these repositories as sources of Real-World Data.

Acknowledgements

The authors thank Malu Calle for useful discussions during the preparation of the manuscript.

Funding

This project received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765158 (COSMIC; www.cosmic-h2020.eu).

Conflict of Interest: none declared.

References

Bartlett,V.L. *et al.* (2019) Feasibility of using real-world data to replicate clinical trial evidence. *JAMA Netw. Open*, **2**, e1912869.

Benito,M. *et al.* (2004) Adjustment of systematic microarray data biases. *Bioinformatics*, **20**, 105–114.

Berger,M.L. *et al.* (2017) Good practices for real-world data studies of treatment and/or comparative effectiveness: recommendations from the joint ISPOR-ISPE Special Task Force on real-world evidence in health care decision making. *Pharmacoepidemiol. Drug Saf.*, **26**, 1033–1039.

Borisov,N. *et al.* (2019) Shambhala: a platform-agnostic data harmonizer for gene expression data. *BMC Bioinformatics*, **20**, 66.

Boughorbel,S. *et al.* (2017) Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLoS One*, **12**, e0177678.

Bumgarner,R. (2013) Overview of DNA microarrays: types, applications, and their future. *Curr. Protoc. Mol. Biol.*, **101**, 22.1.1–22.1.11.

Cortes,C. and Vapnik,V. (1995) Support-vector networks. *Mach. Learn.*, **20**, 273–297.

Fawcett,T. (2006) An introduction to ROC analysis. *Pattern Recogn. Lett.*, **27**, 861–874.

U.S. Food and Drug Administration. (2018) *Framework for FDA's Real-World Evidence Program*. https://www.fda.gov/media/120060/download

Hubbell,E. *et al.* (2002) Robust estimators for expression analysis. *Bioinformatics*, **18**, 1585–1592.

Irigoyen,A. *et al.* (2018) Integrative multi-platform meta-analysis of gene expression profiles in pancreatic ductal adenocarcinoma patients for identifying novel diagnostic biomarkers. *PLoS One*, **13**, e0194844.

Irizarry,R.A. *et al.* (2003) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, **4**, 249–264.

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

Lashkari,D.A. *et al.* (1997) Yeast microarrays for genome wide parallel genetic and gene expression analysis. *Proc. Natl. Acad. Sci. USA*, **94**, 13057–13062.

Lê Cao,K.-A. *et al.* (2014) YuGene: a simple approach to scale gene expression data derived from different platforms for integrated analyses. *Genomics*, **103**, 239–251.

Lloyd,S. (1982) Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, **28**, 129–137.

Maaten,L.v.d. and Hinton,G. (2008) Visualizing data using t-SNE. *J. Mach. Learn. Res.*, **9**, 2579–2605.

Maire,V. *et al.* (2013a) Polo-like Kinase 1: a potential therapeutic option in combination with conventional chemotherapy for the management of patients with triple-negative breast cancer. *Cancer Res.*, **73**, 813–823.

Maire,V. *et al.* (2013b) TTK/hMPS1 is an attractive therapeutic target for triple-negative breast cancer. *PLoS One*, **8**, e63712–15.

MAQC Consortium (2006) The MicroArray Quality Control (MAQC) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nat. Biotechnol.*, **24**, 1151–1161.

Maubant,S. *et al.* (2015) Transcriptome analysis of Wnt3a-treated triple-negative breast cancer cells. *PLoS One*, **10**, e0122333–26.

Meng,Q. *et al.* (2017) DBNorm: normalizing high-density oligonucleotide microarray data based on distributions. *BMC Bioinformatics*, **18**, 527.

Nagalakshmi,U. *et al.* (2008) The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, **320**, 1344–1349.

Piccolo,S.R. *et al.* (2012) A single-sample microarray normalization method to facilitate personalized-medicine workflows. *Genomics*, **100**, 337–344.

Piccolo,S.R. *et al.* (2013) Multiplatform single-sample estimates of transcriptional activation. *Proc. Natl. Acad. Sci. USA*, **110**, 17778–17783.

Platts,A.E. *et al.* (2007) Success and failure in human spermatogenesis as revealed by teratozoospermic RNAs. *Hum. Mol. Genet.*, **16**, 763–773.

Rapaport,F. *et al.* (2013) Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome Biol.*, **14**, R95.

Rousseeuw,P.J. (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, **20**, 53–65.

Rudy,J. and Valafar,F. (2011) Empirical comparison of cross-platform normalization methods for gene expression data. *BMC Bioinformatics*, **12**, 467.

Schena,M. *et al.* (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, **270**, 467–470.

SEQC/MAQC-III Consortium. (2014) A comprehensive assessment of RNA-seq accuracy, reproducibility and information content by the Sequencing Quality Control Consortium. *Nat. Biotechnol.*, **32**, 903–914.

Shabalin,A.A. *et al.* (2008) Merging two gene-expression studies via cross-platform normalization. *Bioinformatics*, **24**, 1154–1160.

Sherman,R.E. *et al.* (2017) Accelerating development of scientific evidence for medical products within the existing US regulatory framework. *Nat. Rev. Drug Discov.*, **16**, 297–298.

Shimodaira,H. (2004) Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling. *Ann. Stat.*, **32**, 2616–2641.

Storey,J.D. (2002) A direct approach to false discovery rates. *J. R. Stat. Soc. B*, **64**, 479–498.

Suzuki,R. and Shimodaira,H. (2006) Pvclust: an R package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, **22**, 1540–1542.

Trotta,F. (2012) Discrepancies between observational studies and randomized controlled trials. *Focus Farmacovigilanza*, **73**, 1.

Walsh,C.J. *et al.* (2015) Microarray meta-analysis and cross-platform normalization: integrative genomics for robust biomarker discovery. *Microarrays*, **4**, 389–406.

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**, e15.

Zhang,S. *et al.* (2020) MatchMixeR: a cross-platform normalization method for gene expression data integration. *Bioinformatics*, **36**, 2486–2491.