# FedPyDESeq2: a federated framework for bulk RNA-seq differential expression analysis

Boris Muzellec, Ulysse Marteau-Ferey and Tanguy Marchand

Owkin France, 14/16 boulevard Poissonnière, Paris, 75009, France.

*Corresponding author(s). E-mail(s): boris.muzellec@owkin.com;
Contributing authors: ulysse.marteau@owkin.com;
tanguy.marchand@owkin.com;

**Abstract**

Despite opportunities for deeper clinical insights, large-scale transcriptome studies are often limited by data silos and risks of privacy leakage. While meta-analysis can be employed to aggregate local results, this is at the price of lower statistical power, especially in heterogeneous settings. A recent paradigm in distributed computing, federated learning (FL) is a means of fitting models from siloed data, while ensuring that private data does not leave its storage facilities. Here, we introduce `FedPyDESeq2`, a software for differential expression analysis (DEA) on siloed bulk RNA-seq. Building on FL tools, `FedPyDESeq2` implements the `DESeq2` pipeline for DEA on siloed datasets in a privacy-enhancing manner. We benchmark `FedPyDESeq2` on datasets from The Cancer Genome Atlas corresponding to 8 different indications, split by geographical origin. `FedPyDESeq2` achieves near-identical results on siloed data compared with `DESeq2` on pooled data, and significantly outperforms meta-analysis baselines on siloed data.

Differential expression analysis (DEA) is the primary application of RNA-seq, whether at bulk or single-cell resolution [1]. From sequenced transcriptomic profiles, DEA aims to identify differentially expressed genes (DEGs) between two conditions (e.g., treated and control samples), using statistical software such as `edgeR` [2], `limma` [3] or `DESeq2` [4] and its recently proposed python alternative, `PyDESeq2` [5].

To obtain relevant results, sample size is a critical factor. Several studies [6, 7] have come to the conclusion that replicate numbers have a larger impact than library sizes on the power of DE analyses, especially in the context of high sample variability, as pointed out in [1]: "for highly diverse samples, such as clinical tissue from cancer

patient tumours, many more replicates are likely to be required in order to pinpoint changes with confidence."

The costs of RNA sequencing put aside, one of the major obstacles to the existence of larger cohorts is the fact that genomic data is very sensitive, and its exchanges tightly regulated by privacy-preserving legal constraints. As a result, DEA is often limited by the quantity of RNA-seq data held by a single institution. When it is impossible to share data between institutions, meta-analysis [8] may be employed to aggregate the local results of running independent analyses in each data center. However, this comes at the cost of statistical power, especially in the presence of data heterogeneity [9].

Recently, federated learning (FL) [10, 11] has emerged as a promising technology to face difficulties coming from siloed data. FL is a distributed computing paradigm originating from the machine learning (ML) community, in which multiple parties (clients, or centers) collaborate in fitting a joint model under the supervision of a central server, without exchanging private data. The chief benefit of FL is that it allows clients to keep control of their data, while allowing them to build ML models as if the data were pooled together. Schematically, FL consists of the following steps: (1) clients perform computations on their private data, and then (2) transfer their local models to the server. The server (3) aggregates models to form a single global model, which it (4) shares back to clients. The process is then repeated from step (1), and so on.

To overcome to the inherent privacy constraints linked to medical data, several recent works have successfully applied FL to biomedical tasks. For example, FedECA [12] introduces a federated learning inverse probability of treatment weighting (IPTW) method as a means to implement external control arms. FL has also been employed for drug discovery [13], or to predict histological response to neoadjuvant chemotherapy in triple-negative breast cancer [14]. Finally, Flimma [15] aims to reproduce the limma-voom [3] DEA workflow in a FL setting. To the best of our knowledge, Flimma is the only federated framework for DEA so far.

In this paper, we introduce `FedPyDESeq2`, a software for federated DEA which aims to reproduce the results of `DESeq2` [4] in the presence of data silos. `FedPyDESeq2` is based on `Substra` [16], an open-source FL software hosted by the Linux foundation for artificial intelligence and data, and on `PyDESeq2` [5], a python re-implementation of the original R `DESeq2` package [4]. The goal of `FedPyDESeq2` is to output, from siloed bulk RNA-seq data, results that are as close as possible to those which would have been obtained if `PyDESeq2` had been applied after pooling the data together. To do so, `FedPyDESeq2` follows the `DESeq2` pipeline, implementing a equivalent version of each step, while ensuring that no individual data is transferred from the centers.

We evaluate `FedPyDESeq2` on bulk RNA-seq datasets from The Cancer Genome Atlas (TCGA, https://www.cancer.gov/tcga) corresponding to 8 different indications, which we split according to two different methodologies. First, following [17], we recover realistic federated datasets by splitting the data according to their sampling source sites, which we group in 7 regions. Second, to further benchmark the robustness of our approach, we generate a suite of synthetic splits of increasing heterogeneity. We

then run FedPyDESeq2 on a federated network in which each split is stored in an independent machine. In both settings, our experiments show that FedPyDESeq2 yields results that are extremely close to those of PyDESeq2 on pooled data. Finally, we compare FedPyDESeq2 with various meta-analysis methods, which output global results by aggregating local DEA results from each center. Our experiments show that using our federated method instead of meta-analysis leads to improved statistical power and sensitivity.

FedPyDESeq2 is available at https://github.com/owkin/fedpydeseq2 under an MIT license. The scripts to reproduce our experiments are available at https://github.com/owkin/fedpydeseq2-paper-experiments.

# Results

## Overview of FedPyDESeq2

FedPydDESeq2 is a python software for differential expression analysis on siloed bulk RNA-seq data. FedPydDESeq2 operates on a network consisting of a set of centers, each with their own private RNA-seq data and sample covariates, and a central server to orchestrate them. For each gene, FedPyDESeq2 derives a log-fold change (LFC) parameter and a p-value. Computations are performed without any exchange of raw individual RNA-seq or covariate data over the network, neither between the centers and the server, nor between the centers themselves.

FedPyDESeq2 is built on Substra [16], an open-source federated learning framework which has already been applied in federated biomedical studies [13, 14]. Substra can be deployed in real-world federated settings, but also supports a simulated mode which can be run on a single machine, such as a modern laptop.

From a bird's eye view, FedPyDEseq2 consists of a succession of two types of routines:

1. Local steps, run independently by each center. During a local step, centers may use their own private data and quantities shared by the central server. The results are then stored locally in each center, and part of them may be transferred to the central server in the form of shared states;
2. Aggregation steps, run on the central server, using the shared states sent by the centers. The output of aggregation steps are then shared back to the centers.

The succession of local steps and aggregation steps is statically fixed in advance and communicated to the centers and the server by Substra. See Fig. 1 for a schematic depiction of FedDESeq2 and Substra.

Formally, FedPyDESeq2 follows the same mathematical procedure as DESeq2 [4]. For each gene, raw counts are modeled as a negative binomial distribution. Dispersion parameters are first estimated independently for each gene by fitting a generalized linear model (GLM). They are then shrunk towards a global trend curve, to pool information among genes. Next, dispersions are used to fit gene-wise log-fold changes (LFC), which encode the differences in expression levels between conditions. Outliers are then detected and removed using Cook's distances [18]. Finally, p-values are computed from LFCs and dispersions using Wald tests, and adjusted to control the false

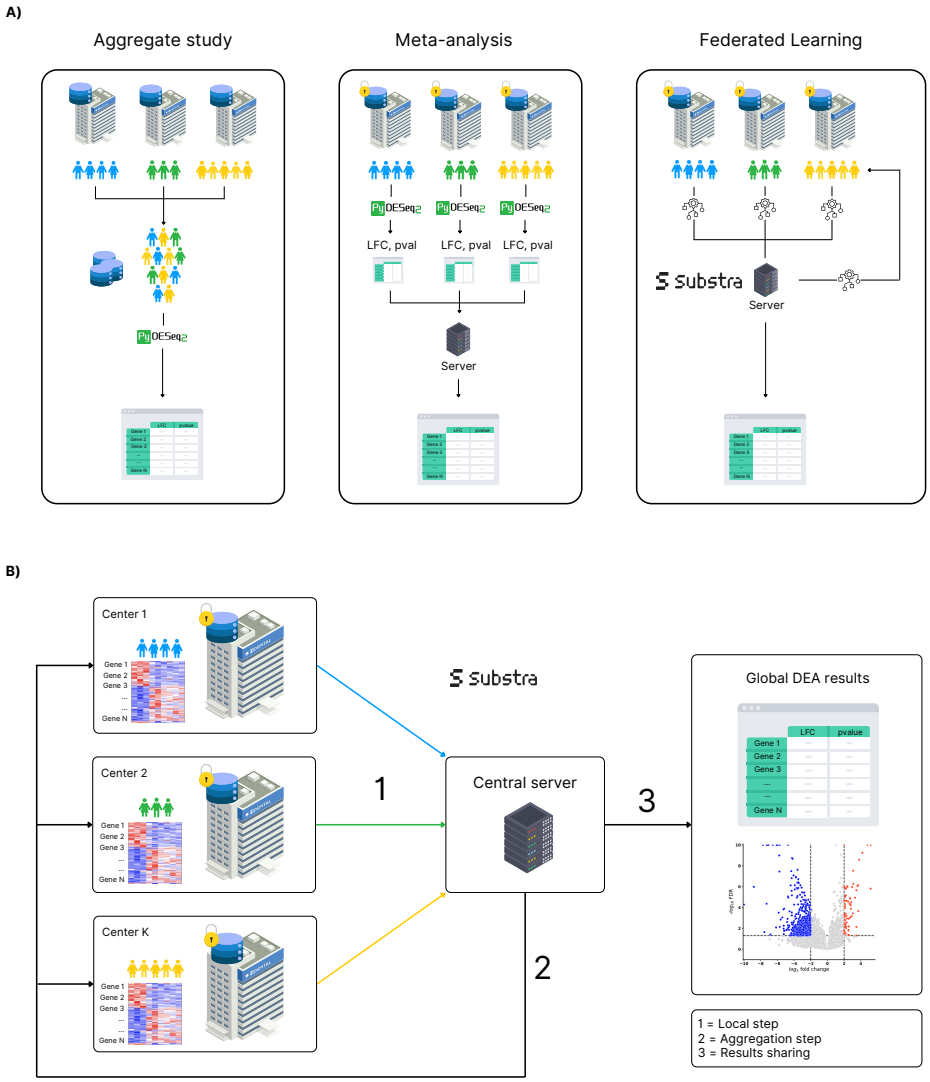


**Fig. 1** **(A)** Comparison of aggregate analysis (left), meta-analysis (middle) and federated learning (right) methods for DEA on siloed RNA-seq data. While aggregate analysis is the preferred method, it requires pooling data from all centers in a single processing center, which is often unfeasible. Meta-analysis circumvents this issue by running a full DEA pipeline locally on each center, sharing the resulting log-fold changes and p-values to a server for aggregation, at the loss of statistical power. FL, on the other hand, aims to reproduce the algorithm of the aggregate study by distributing the computation over a network, without private data leaving their premises. **(B)** `FedPyDESeq2` alternates between (1) `local steps` run independently in each center and (2) `aggregation steps` run centrally by the server. This is done for multiple rounds over a federated network orchestrated by `Substra`, before (3) the server outputs final results. To ensure privacy, centers may only share aggregate data.

discovery rate (FDR). We refer to the Methods section for more details on `DESeq2` and `PyDESeq2`.

FedPyDESeq2 is structured along the same architecture as PyDESeq2, but implements each step in a manner that is compatible with federated settings. To do so, a few adjustments are required. FedPyDESeq2 fits dispersions parameters using grid search instead of l-bfgs-b [19], which is less amenable to federation. To fit LFCs, we implement a federated version of the iteratively reweighted least squares algorithm [IRLS, see, e.g., 20]. A final challenge lays in the computation of Cook's distances for outlier detection [18], for which we design a federated quantile computation procedure. Each of the above procedures are detailed in the Methods section.

To prevent data leakage, FedPyDESeq2 adheres to the following privacy guideline: data at the sample or patient level, such as individual RNA transcripts, design covariates or size factors, is never shared. These quantities are considered private and cannot leave their premises. Only gene-level information that is part of the output of the algorithm (such as dispersions, LFCs or p-values) is exchanged over the federated network, between the centers and the server.

We refer to the supplementary material for a detailed account of quantities shared between the centers and the server.

The features currently available in FedPyDESeq2 cover the majority of DEA cases. FedPyDESeq2 supports multi-factor designs for both categorical and continuous covariates, the refitting of outliers based on Cook's distances, p-value computation using Wald tests, and p-value adjustment using the Benjamini-Hochberg procedure and independent filtering (see [4] for details).

## FedPyDESeq2 yields near-identical results on siloed RNA-seq data compared with PyDESeq2 on pooled data

The goal of FedPyDESeq2 is to obtain, from siloed RNA-seq data, LFCs and p-values that are as close as possible ti those that would be output by PyDESeq2 on the same data, if it were pooled together in a single database. We evaluate this by comparing the outputs of both methods on RNA-seq datasets from The Cancer Genome Atlas (TCGA, https://www.cancer.gov/tcga) corresponding to 8 different indications. In a first set of experiments, we simulate a realistic federated scenario by splitting each dataset according to the geographical origin of each sample (available in the metadata), resulting in 7 sub-datasets. This methodology was introduced in [17]. We refer to the Datasets paragraph in the Methods section for more details.

For each indication, we run FedPyDESeq2 on a network of 8 machines, comprising a server and 7 clients (one for each sub-dataset), and compare the results with those obtained by running PyDESeq2 from the original undivided data on a single machine.

This experiment is repeated with three different designs, using covariates corresponding to clinical data from TCGA : (i) a single-factor design, (ii) a multi-factor design with categorical factors, and (iii) a multi-factor design which includes a numerical factor.

The results for design (i) are summarized in Fig. 2, and those of the remaining experiments are detailed in the supplementary material.

Across all 8 indications, FedPyDESeq2 retrieves the vast majority of genes marked as differentially expressed by PyDESeq2. Looking closer, it appears that differences are mainly due to genes identified as DE by PyDESeq2, but not FedPyDESeq2. Overall,

FedPyDESeq2 tends to be less sensitive than PyDESeq2, but seldom introduces new DEGs. As can be seen from Fig. 2 C, differences in DEGs are mainly due to p-values that are close to the 0.05 threshold. We refer to the supplementary material for additional plots illustrating this point.

Next, we validate the biological soundness of our results by performing gene set enrichment analysis (GSEA) using the fgsea package [21] on both sets of DEGs. As shown in Fig 2 B, this results in near-identical sets of significantly enriched pathways.

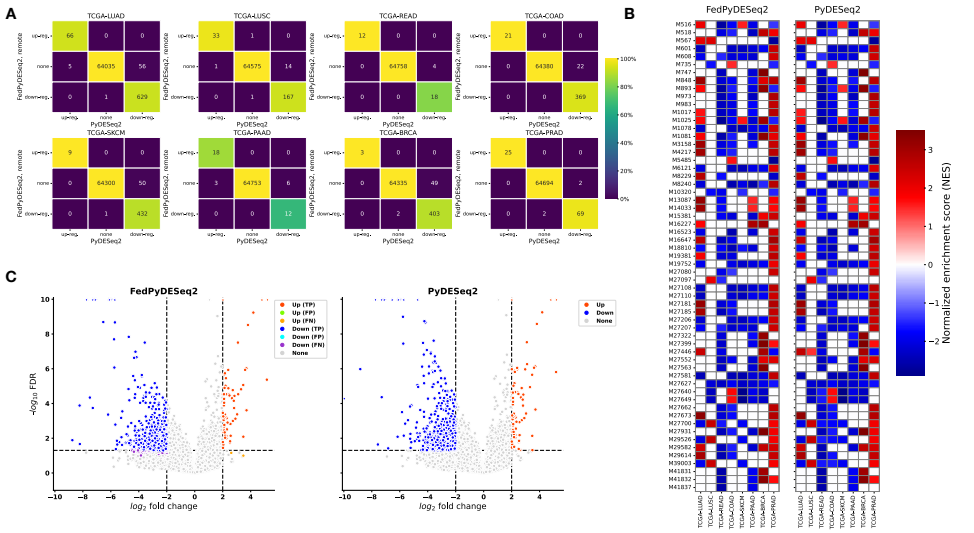


**Fig. 2** Results of the single-factor design experiment (Advanced vs Non-advanced tumoral stage) with a geographical split. **(A)** DEGs (with padj $\leq 0.05$ and $|\text{LFC}| \geq 2$) according to FedPyDESeq2 (from siloed data) and PyDESeq2 (from pooled data). Percentages are expressed w.r.t. column totals. **(B)** Significantly enriched pathways (padj $\leq 0.05$) obtained with the fgsea package, using Wald statistics as gene-ranking metric. Only pathways that are in the top 10 (according to the adjusted p-value) for at least one indication are represented. If for a given indication, a pathway is not significantly enriched, the corresponding square is left blank. **(C)** Volcano plots of DEGs in TCGA-LUAD (left: FedPyDESeq2, siloed data; right: PyDESeq2, pooled data). Differences in DEGs are marked as false positives/negatives (FP/FN) on the FedPyDESeq2 plot.

## FedPyDESeq2 significantly outperforms local methods on siloed RNA-seq data

We compare FedPyDESeq2 to a set of meta-analysis baselines [8] which can be run on siloed data without sharing local transcriptomic or covariate data. The meta-analysis methods benchmarked in our work include: (i) a fixed-effects model, (ii) random-effects models, with either the Paule-Mandel [22] iterative estimator or the DerSimonian and Laird one-step estimator [23], and (iii) p-value combination with Stouffer and Fisher methods (see, e.g., [24]). We also include in our benchmark a naive method which

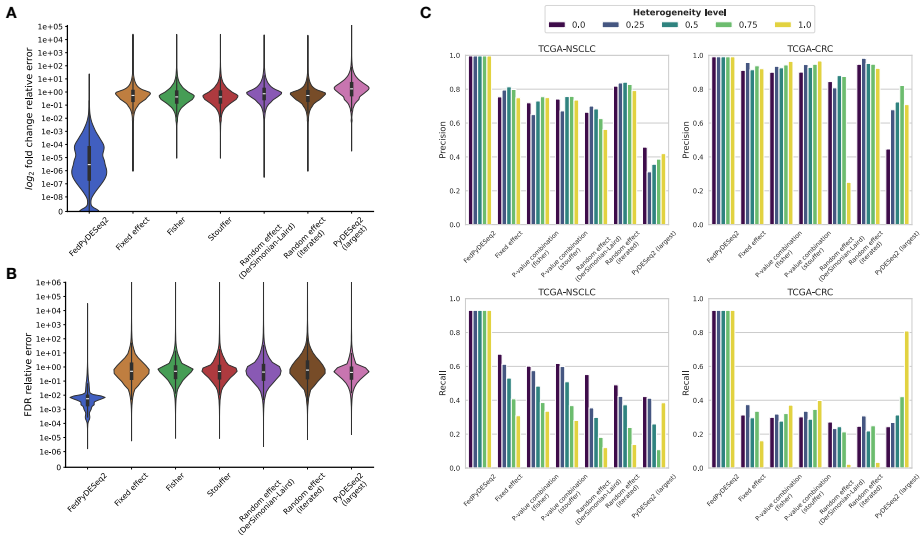


**Fig. 3** Comparison of `FedPyDESeq2` with meta-analysis on siloed data. **(A)** LFC and **(B)** adjusted p-value relative errors of `FedPyDESeq2` and baseline methods applied on geographically split data, compared with `PyDESeq2` on pooled data ($\frac{|\theta_{\texttt{method}} - \theta_{\texttt{PyDESeq2}}|}{|\theta_{\texttt{PyDESeq2}}|}$), evaluated on TCGA-LUAD. **(C)** Precision and recall scores of `FedPyDESeq2` and baselines on siloed TCGA-NSLCL and TCGA-CRC data with increasing heterogeneity, compared with `PyDESeq2` on pooled data.

consists in restricting the analysis to the largest local cohort. For all methods, we use the results of `PyDESeq2` applied to the pooled dataset as reference.

In addition to the geographical split, we introduce a set of synthetic splits to further evaluate the robustness of our methods w.r.t. data heterogeneity. To do so, we group two colorectal cancer (CRC) datasets together, TCGA-COAD and TCGA-READ. We then simulate various scenarios of two centers containing CRC patients data with increasing levels of heterogeneity. At the lowest level of heterogeneity, each of the two datasets contains the same proportion of patients from TCGA-COAD and TCGA-READ, while at the highest level of heterogeneity, one of the dataset only contains TCGA-COAD patients and the other one only TCGA-READ. We apply the same procedure to two non-small cell lung cancer (NSCLC) datasets: TCGA-LUAD and TCGA-LUSC. We refer to the methods section for a detailed description of splitting procedures and meta-analysis baselines.

We perform our benchmark both on heterogeneous splits and on the geographical split introduced previously. The results our experiments are summarized in Fig. 3. While in both CRC and NSCLC, `FedPyDESeq2` retrieves about 93% of `PyDESeq2` DEGs, meta-analysis methods perform much worse all-around, which makes them a poor substitute for federated methods. Overall, the low performance of meta-analysis is mostly due to poor sensitivity (i.e., most DEGs are missed), with the exception of the DerSimonian and Laird random effect model [23], which also introduces many false positives. We refer to the supplementary material for a finer-grain report of each

method's output. Second, whereas FedPyDESeq2 is insensitive to data heterogeneity, the recall of meta-analysis deteriorates sharply as heterogeneity increases, i.e., fewer and fewer DEGs are correctly identified. As a notable exception, the performance of PyDESeq2 restricted to the largest CRC cohort increases with heterogeneity. However, we believed that this phenomenon is an artifact, which can be explained by the unbalanced numbers of replicates between TCGA-COAD and TCGA-READ, resulting in a maximum number of replicates in the largest cohort at the maximum heterogeneity level.

# Discussion

FedPyDESeq2 was designed to address the issue of data silos in bulk RNA-seq DEA, which limits the size of cohorts. FedPyDESeq2 leverages FL to perform DEA without sharing individual patient data, such as RNA transcripts or design covariates, relying instead on data aggregation. Our experiments show that the results of FedPyDESeq2 applied on siloed data are very close to those PyDESeq2 would yield if it were applied on pooled data. Further, FedPyDESeq2 strongly outperforms methods based on the aggregation of local (shareable) PyDESeq2 results, such as fixed or random effects models, or p-value combination methods.

Furthermore, FedPyDESeq2 opens the way for DEA studies which would be completely impossible to perform without pooling the data together. Indeed, FedPyDESeq2 allows comparing cohorts that are perfectly partitioned across data silos. As an example, it could be used to determine whether patients from a hospital A have DEGs compared with patients from a hospital B. This enables comparing the effects of treatment protocols that are performed exclusively in a given hospital. On the other hand, such an analysis would be intractable using meta-analysis, as it may only aggregate local results, in which the comparison of A patients with B patients is impossible.

FedPyDESeq2 is based on federated learning, a privacy-enhancing technology that ensures that raw data stays within each center and is not shared across the network. However, on a cautionary note, various studies [11, 25, 26] have shown that privacy leakage can still occur, as intermediate quantities exchanged over the network may contain private information. In the case of machine learning models trained using FedAverage [10, 11], an attacker could in some conditions reconstruct the local data from the gradients exchanged over the network. In FedPyDESeq2, the quantities shared with the aggregator consist only in aggregated read counts statistics, such as log-means or variances, or Gram or hat matrices built from the design covariates. We provide an exhaustive list of those quantities in the supplementary material. While a detailed study evaluating precisely the risks of data leakage is left to future work, we believe that as long as each center has a sufficient number of samples, an attacker cannot reconstruct initial gene counts from the quantities exchanged over the network.

Further privacy-enhancing technologies can be applied on top of FL, in order to strengthen the privacy guarantees. As an example, differential privacy [27, 28] provides theoretical guarantees on privacy leakage by adding random noise to the initial data. However, in doing so, it alters the final output of the FL algorithm. Another privacy-enhancing layer that can be added is secure aggregation [29] which relies on

cryptography methods to hide center-level shared states from the server. However, while secure aggregation leaves the outputs of FL algorithms unchanged, it induces strong computational overheads.

A critical aspect of federated learning is the heterogeneity that is inherent to different data sources [11]. In this work, our objective is to produce results that are as close as possible to those that would have been obtained from pooled analysis, and to show the robustness of `FedPyDESeq2` with respect to heterogeneous data silos. In practice, more relevant results could be obtained by taking heterogeneity into account. As advocated in the `DESeq2` vignette, this can either be achieved by including batch variables (e.g., hospital A vs hospital B) in the design of `FedPyDESeq2`, or by using dedicated batch effect correction tools before we perform the analysis, such as `RUVSeq` [30], `SVASeq` [31], `ComBat` [32] and `ComBat-Seq` [33]. In that perspective, a federated version of `ComBat`, `Fed-Combat` [34], was recently proposed.

Finally, let us remark that DEA is only one of the many steps involved in most RNA-seq analysis pipelines. Hence, to build a complete pipeline, federated software handling complementary tasks such as clustering is required (see, e.g., [35] and [36] for federated principal component analysis). More importantly, let us highlight the importance of applying unified sample processing across all centers (library preparation, sequencing technology, read files preprocessing and alignment) when using `FedPyDESeq2`, to ensure that the transcripts which are being compared correspond to the same biological entities.

# References


[1] Stark, R., Grzelak, M., Hadfield, J.: RNA sequencing: the teenage years. Nature Reviews Genetics **20**(11), 631–656 (2019)

[2] Robinson, M.D., McCarthy, D.J., Smyth, G.K.: edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. bioinformatics **26**(1), 139–140 (2010)

[3] Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., Smyth, G.K.: limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic acids research **43**(7), 47–47 (2015)

[4] Love, M.I., Huber, W., Anders, S.: Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome biology **15**, 1–21 (2014)

[5] Muzellec, B., Telenczuk, M., Cabeli, V., Andreux, M.: PyDESeq2: a python package for bulk RNA-seq differential expression analysis. Bioinformatics (2023)

[6] Schurch, N.J., Schofield, P., Gierliński, M., Cole, C., Sherstnev, A., Singh, V., Wrobel, N., Gharbi, K., Simpson, G.G., Owen-Hughes, T., *et al.*: How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use? Rna **22**(6), 839–851 (2016)

[7] Lamarre, S., Frasse, P., Zouine, M., Labourdette, D., Sainderichin, E., Hu, G., Le Berre-Anton, V., Bouzayen, M., Maza, E.: Optimization of an RNA-Seq differential gene expression analysis depending on biological replicate number and library size. Frontiers in plant science **9**, 108 (2018)

[8] Borenstein, M., Hedges, L.V., Higgins, J.P., Rothstein, H.R.: Introduction to Meta-analysis. John Wiley & Sons, ??? (2021)

[9] Higgins, J.P., Thompson, S.G.: Quantifying heterogeneity in a meta-analysis. Statistics in medicine **21**(11), 1539–1558 (2002)

[10] McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017). PMLR

[11] Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., *et al.*: Advances and open problems in federated learning. Foundations and trends® in machine learning **14**(1–2), 1–210 (2021)

[12] Terrail, J.O.d., Klopfenstein, Q., Li, H., Mayer, I., Loiseau, N., Hallal, M., Balazard, F., Andreux, M.: FedECA: A federated external control arm method for causal inference with time-to-event data in distributed settings. arXiv preprint arXiv:2311.16984 (2023)

[13] Heyndrickx, W., Mervin, L., Morawietz, T., Sturm, N., Friedrich, L., Zalewski, A., Pentina, A., Humbeck, L., Oldenhof, M., Niwayama, R., Schmidtke, P., Fechner, N., Simm, J., Arany, A., Drizard, N., Jabal, R., Afanasyeva, A., Loeb, R., Verma, S., Harnqvist, S., Holmes, M., Pejo, B., Telenczuk, M., Holway, N., Dieckmann, A., Rieke, N., Zumsande, F., Clevert, D.-A., Krug, M., Luscombe, C., Green, D., Ertl, P., Antal, P., Marcus, D., Do Huu, N., Fuji, H., Pickett, S., Acs, G., Boniface, E., Beck, B., Sun, Y., Gohier, A., Rippmann, F., Engkvist, O., Göller, A.H., Moreau, Y., Galtier, M.N., Schuffenhauer, A., Ceulemans, H.: MELLODDY: Cross-pharma federated learning at unprecedented scale unlocks benefits in qsar without compromising proprietary information. Journal of Chemical Information and Modeling **64**(7), 2331–2344 (2024)

[14] Terrail, J., Leopold, A., Joly, C., Béguier, C., Andreux, M., Maussion, C., Schmauch, B., Tramel, E.W., Bendjebbar, E., Zaslavskiy, M., *et al.*: Federated learning for predicting histological response to neoadjuvant chemotherapy in triple-negative breast cancer. Nature medicine **29**(1), 135–146 (2023)

[15] Zolotareva, O., Nasirigerdeh, R., Matschinske, J., Torkzadehmahani, R., Bakhtiari, M., Frisch, T., Späth, J., Blumenthal, D.B., Abbasinejad, A., Tieri, P., *et al.*: Flimma: a federated and privacy-aware tool for differential gene expression analysis. Genome biology **22**(1), 338 (2021)

[16] Galtier, M.N., Marini, C.: Substra: a framework for privacy-preserving, traceable and collaborative machine learning. arXiv preprint arXiv:1910.11567 (2019)

[17] Terrail, J., Ayed, S.-S., Cyffers, E., Grimberg, F., He, C., Loeb, R., Mangold, P., Marchand, T., Marfoq, O., Mushtaq, E., Muzellec, B., Philippenko, C., Silva, S., Teleńczuk, M., Albarqouni, S., Avestimehr, S., Bellet, A., Dieuleveut, A., Jaggi, M., Karimireddy, S.P., Lorenzi, M., Neglia, G., Tommasi, M., Andreux, M.: FLamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems, vol. 35, pp. 5315–5334 (2022)

[18] Weisberg, S., Cook, R.D.: Residuals and influence in regression (1982)

[19] Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical programming **45**(1), 503–528 (1989)

[20] Friedman, J.H., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of statistical software **33**, 1–22 (2010)

[21] Sergushichev, A.A.: An algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation. bioRxiv (2016)

[22] Paule, R.C., Mandel, J.: Consensus values and weighting factors. Journal of research of the National Bureau of Standards **87**(5), 377 (1982)

[23] DerSimonian, R., Laird, N.: Meta-analysis in clinical trials. Controlled clinical trials **7**(3), 177–188 (1986)

[24] Rau, A., Marot, G., Jaffrézic, F.: Differential meta-analysis of RNA-seq data from multiple studies. BMC bioinformatics **15**, 1–10 (2014)

[25] Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients-how easy is it to break privacy in federated learning? Advances in neural information processing systems **33**, 16937–16947 (2020)

[26] Huang, Y., Gupta, S., Song, Z., Li, K., Arora, S.: Evaluating gradient inversion attacks and defenses in federated learning. Advances in neural information processing systems **34**, 7232–7241 (2021)

[27] Dwork, C.: Differential privacy: A survey of results. In: International Conference on Theory and Applications of Models of Computation, pp. 1–19 (2008). Springer

[28] Dwork, C., Roth, A., *et al.*: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)

[29] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)

[30] Risso, D., Ngai, J., Speed, T.P., Dudoit, S.: Normalization of RNA-seq data using factor analysis of control genes or samples. Nature biotechnology **32**(9), 896–902 (2014)

[31] Leek, J.T.: Svaseq: removing batch effects and other unwanted noise from sequencing data. Nucleic acids research **42**(21), 161–161 (2014)

[32] Johnson, W.E., Li, C., Rabinovic, A.: Adjusting batch effects in microarray expression data using empirical bayes methods. Biostatistics **8**(1), 118–127 (2007)

[33] Zhang, Y., Parmigiani, G., Johnson, W.E.: ComBat-seq: batch effect adjustment for RNA-seq count data. NAR genomics and bioinformatics **2**(3), 078 (2020)

[34] Silva, S., Oxtoby, N., Altmann, A., Lorenzi, M.: Fed-ComBat: A generalized federated framework for batch effect harmonization in collaborative studies. bioRxiv, 2023–05 (2023)

[35] Grammenos, A., Mendoza Smith, R., Crowcroft, J., Mascolo, C.: Federated principal component analysis. Advances in neural information processing systems **33**, 6453–6464 (2020)

[36] Hartebrodt, A., Röttger, R.: Federated horizontally partitioned principal component analysis for biomedical applications. Bioinformatics Advances **2**(1), 026 (2022)

[37] Cox, D.R., Reid, N.: Parameter orthogonality and approximate conditional inference. Journal of the Royal Statistical Society: Series B (Methodological) **49**(1), 1–18 (1987)

[38] Cule, E., Vineis, P., De Iorio, M.: Significance testing in ridge regression for genetic data. BMC bioinformatics **12**, 1–15 (2011)

[39] Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal statistical society: series B (Methodological) **57**(1), 289–300 (1995)

[40] Kim, D., Sra, S., Dhillon, I.S.: Tackling box-constrained optimization via a new projected quasi-newton approach. SIAM Journal on Scientific Computing **32**(6), 3548–3563 (2010)

[41] Armijo, L.: Minimization of functions having lipschitz continuous first partial derivatives. Pacific Journal of mathematics **16**(1), 1–3 (1966)

[42] Veroniki, A.A., Jackson, D., Viechtbauer, W., Bender, R., Bowden, J., Knapp, G., Kuss, O., Higgins, J.P., Langan, D., Salanti, G.: Methods to estimate the between-study variance and its uncertainty in meta-analysis. Research synthesis methods **7**(1), 55–79 (2016)

[43] Wilks, C., Zheng, S.C., Chen, F.Y., Charles, R., Solomon, B., Ling, J.P., Imada, E.L., Zhang, D., Joseph, L., Leek, J.T., *et al.*: recount3: summaries and queries for large-scale rna-seq expression and splicing. Genome biology **22**, 1–40 (2021)

[44] Aran, D., Sirota, M., Butte, A.J.: Systematic pan-cancer analysis of tumour purity. Nature communications **6**(1), 8971 (2015)

[45] Liberzon, A., Subramanian, A., Pinchback, R., Thorvaldsdóttir, H., Tamayo, P., Mesirov, J.P.: Molecular signatures database (MSigDB) 3.0. Bioinformatics **27**(12), 1739–1740 (2011)

[46] Jassal, B., Matthews, L., Viteri, G., Gong, C., Lorente, P., Fabregat, A., Sidiropoulos, K., Cook, J., Gillespie, M., Haw, R., *et al.*: The reactome pathway knowledgebase. Nucleic acids research **48**(D1), 498–503 (2020)

# Methods

## The `DESeq2` model

In `DESeq2` [4], the user provides a gene count matrix $Y \in \mathbb{R}^{n \times G}$ and a design matrix $X \in \mathbb{R}^{n \times p}$ corresponding to $n$ input samples and $G$ genes.

For each gene $g$, `DESeq2` models gene counts $(Y_{ig})_{1 \leq i \leq n}$ as **independent** samples from negative binomials:

$$Y_{ig} \sim \mathrm{NB}(\mu_{ig}, \alpha_g),$$

where

- the dispersion parameter $\alpha_g$ depends only on gene $g$;
- the mean parameter $\mu_{ig}$ depends on the covariate vector $X^{(k)}\mathbb{R}^p$, the size factor $\gamma_i$ of sample $i$, and the log fold change (LFC) vector $\beta_g \in \mathbb{R}^p$ of gene $g$, as a generalized linear model (GLM):
$$\mu_{ig} := \gamma_i \exp(\langle X^{(k)}, \beta_g \rangle).$$

Let $\ell(Y_{ig}; X_i, \gamma_i | \alpha_g, \beta_g)$ be the negative log likelihood (NLL) of the negative binomial distribution of a sample characterized by its count, design and size factor, with parameters $\alpha_g, \beta_g$. The high-level objective of `DESeq2` is to estimate the dispersion and log fold change parameters $\alpha_g$ and $\beta_g$ for all genes, by successively using method of moments techniques and alternatively minimizing the NLL (sometimes with prior regularization) w.r.t. $\alpha_g$ and $\beta_g$.

## The DESeq2 workflow

We now briefly describe the main steps of the DESeq2 workflow, to serve as a reference when we describe later how they are implemented in FedPyDESeq2. We refer to the original DESeq2 paper [4] and its implementation (https://github.com/thelovelab/DESeq2) for further details.

### Size factors

DESeq2 estimates size factors using the median of ratios method. Discarding genes which have at least one zero count value, we first compute a pseudo reference $R_g$ for each gene $g$ as the geometric mean of its counts across samples:

$$R_g = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log y_{ig}\right). \tag{1}$$

We then define the **size factor** $\gamma_i$ of sample $i$ as

$$\gamma_i = \text{median}_g\left(\frac{y_{ig}}{R_g}\right).$$

### Initial dispersion estimates

Initial values for dispersions are first estimated using a combination of a method of moments and a rough estimate from a linear regression.

#### *Method of moments*

Let $\widehat{\gamma^{-1}} := \frac{1}{n}\sum_{i=1}^{n}\gamma_i^{-1}$, $\bar{\mu}_g := \frac{1}{n}\sum_{i=1}^{n}\frac{Y_{ig}}{\gamma_i}$ and $\bar{\sigma}_g^2 := \frac{1}{n-1}\sum_{i=1}^{n}(\frac{Y_{ig}}{\gamma_i} - \bar{\mu}_g)^2$. The method-of-moments estimate is defined as

$$\hat{\alpha}_g^{\text{MoM}} = \frac{\bar{\sigma}_g^2 - \widehat{\gamma^{-1}}\bar{\mu}_g}{\bar{\mu}_g^2}. \tag{2}$$

#### *Rough dispersions estimates*

We first compute $\hat{\beta}_g \in \mathbb{R}^p$ by fitting a linear regression predicting the normalized counts from the design factors:

$$\hat{\beta}_g = \arg\min_{\beta}\sum_{i=1}^{n}\left(\frac{Y_{ig}}{\gamma_i} - \langle X^{(k)}, \beta\rangle\right)^2. \tag{3}$$

Let $\hat{Z}_{ig} := \max(\langle X^{(k)}, \hat{\beta}_g\rangle, 1)$.
The "rough dispersion estimates" (RDE) are defined as:

$$\hat{\alpha}_g^{\text{RDE}} = \frac{1}{n-p}\sum_{i=1}^{n}\frac{(\frac{Y_{ig}}{\gamma_i} - \hat{Z}_{ig})^2 - \hat{Z}_{ig}}{\hat{Z}_{ig}^2} \tag{4}$$

***Initial estimates***

Finally, the initial dispersions estimates are set as

$$\hat{\alpha}_g^{\mathrm{INI}} = \min(\hat{\alpha}_g^{\mathrm{RDE}}, \hat{\alpha}_g^{\mathrm{MoM}}).$$

## Initial $\mu$ estimates

First estimates for means $\mu_{ig}$ are obtained by optimizing the log likelihood of the negative binomial w.r.t. LFCs, using iteratively re-weighted least squares (IRLS, see hereafter). Let

$$\hat{\beta}_g^{\mathrm{INI}} := \arg\min_{\beta} \ell_{NB}(Y_{ig}|\beta, \hat{\alpha}_g^{\mathrm{INI}}).$$

We define

$$\hat{\mu}_{ig}^{\mathrm{INI}} := \gamma_i \exp(\langle X_i, \hat{\beta}_g^{\mathrm{INI}} \rangle).$$

At this stage, only the $\hat{\mu}_{ig}^{\mathrm{INI}}$ estimate is relevant, and log-fold changes $\hat{\beta}_g^{\mathrm{INI}}$ are discarded.

## Estimating dispersions

***Gene-wise maximum likelihood estimates***

Dispersions are first estimated independently for each gene using MLE, i.e., by solving, for each gene, the following optimization problem:

$$\alpha_g^{\mathrm{GW}} := \arg\min_{\alpha} \sum_{i=1}^{n} \ell_{NB}(Y_{ig}|\hat{\mu}_{ig}^{\mathrm{INI}}, \alpha) + R_g(\alpha), \tag{5}$$

where $R_g(\alpha_g)$ is a Cox-Reid regularization term [37]:

$$R_g(\alpha) := \frac{1}{2} \log\det\left(X^T W_g(\alpha) X\right) \tag{6}$$

and

$$W_g(\alpha) = \mathrm{diag}\left(\frac{\hat{\mu}_g}{1 + \alpha\hat{\mu}_g}\right) \in \mathbb{R}^{n \times n}.$$

In practice, box constraints are applied to restrict $\alpha_g$ to lie in

$$[\alpha_{\min}, \alpha_{\max}] := [10^{-8}, \max(n, 10)].$$

Numerically, the optimization is carried out on a $\log\alpha$ scale, using line search (`DESeq2`) or L-BFGS-B [19] (`PyDESeq2`).

***Fitting a trend curve to gene-wise dispersion estimates***

Next, information is pooled across genes by fitting a curve relating the gene-wise dispersions $\alpha^{\mathrm{GW}}$ to the inverse of the empirical normalized means $\bar{\mu}_g = \frac{1}{n}\sum_{i=1}^{n} \frac{Y_{ig}}{\gamma_i}$:

$$f(\bar{\mu}) = a_0 + \frac{a_1}{\bar{\mu}}.$$

The parameters $a_0$ and $a_1$ are fitted with a Gamma-distributed generalized linear model, minimizing the following loss function:

$$\min_{a_0,a_1} \sum_g \frac{\alpha_g^{\mathrm{GW}}}{a_0 + a_1/\bar{\mu}_g} + \log\left(a_0 + \frac{a_1}{\bar{\mu}_g}\right). \tag{7}$$

Equation 7 is fit iteratively, excluding genes for which the ratio between the actual value and the fitted curve value is outside of $[10^{-4}, 15]$ at each iteration.

In case the optimization does not converge, a constant (flat) curve is used instead, taking the means of $\alpha^{\mathrm{GW}}$ over all genes, trimmed to remove the top and bottom $10^{-3}$ quantiles of dispersions, as value.

#### *Dispersions variance prior*

A dispersion variance prior is then estimated based on the squared mean absolute deviation (MAD) of the terms $\log\left(\alpha_g^{\mathrm{GW}}/\alpha_g^{\mathrm{tr}}\right)$ (where $\alpha_g^{\mathrm{tr}} := f(\bar{\mu}_g)$), as

$$\nu^2 = \max\left((\mathrm{MAD}\{\log\hat{\alpha}_g - \log\alpha_g^{\mathrm{tr}}\}_g)^2 - \psi\left(\frac{n-p}{2}\right), 1/4\right),$$

where $\psi$ denotes the digamma function.

#### *Maximum a posteriori (MAP) dispersions*

MAP dispersion estimates are then fitted for each gene. The loss is similar to (5) with an additional regularization term to control the log-deviation from the fitted curve:

$$\alpha_g^{\mathrm{MAP}} := \underset{\alpha\in[\alpha_{\min},\alpha_{\max}]}{\arg\min} \sum_{i=1}^{n} \ell_{NB}(Y_{ig}|\hat{\mu}_{ig}, \alpha) + R_g(\alpha) + \frac{(\log\alpha - \log\alpha_g^{\mathrm{tr}})^2}{2\nu^2}, \tag{8}$$

with $\alpha_g^{\mathrm{tr}} = f(\bar{\mu}_g)$. Again, the optimization is performed using line search in log domain, and dispersions are constrained to lie in $[\alpha_{\min}, \alpha_{\max}]$.

#### *Filtering outlier dispersions*

Finally, when $\alpha_g^{\mathrm{GW}}$ is too far above the trend function, i.e. when

$$\log\alpha_i^{GW} > \log\alpha_g^{\mathrm{tr}} + 2(\mathrm{MAD}\{\log\hat{\alpha}_g - \log\alpha_g^{\mathrm{tr}}\}_g)^2, \tag{9}$$

the $\alpha_g^{\mathrm{MAP}}$ is not computed and $\alpha_g^{\mathrm{GW}}$ is used instead.

### Log fold change estimation

Based on the final dispersion estimates $\alpha_g$ obtained in the previous step, we can proceed to the log-fold change estimation. Essentially, `PyDESeq2` assumes that the

raw counts $Y_{ig}$ are sampled from a negative-binomial distribution $Y_{ig} \sim \mathrm{NB}(\mu_{ig}, \alpha_g)$, where $\mu_{ig}$ satisfies:

$$\mu_{ig} = \gamma_i \exp(\langle X^{(k)}, \beta_g \rangle) \tag{10}$$

for some gene-dependent vector $\beta_g$. Let

$$L(\beta_g, \alpha_g) := \sum_{i=1}^{n} \ell_{NB}\left(Y_{ig} | \gamma_i \exp(\langle X^{(k)}, \beta_g \rangle), \alpha_g\right)$$

be the log-likelihood associated with gene $g$. In order to estimate the log-fold change $\beta_g$, `(Py)DESeq2` minimizes $L(\beta, \alpha_g)$ independently for each gene:

$$\min_{\beta_g} \sum_{i=1}^{n} \ell_{NB}\left(y_{ig} | \gamma_i \exp(\langle X^{(k)}, \beta_g \rangle), \alpha_g\right)$$

This optimization is carried out with the iteratively reweighted least-squares (IRLS) algorithm.

From a bird's eye view, IRLS boils down to iterating the following updates, starting from an initial LFC estimate $\beta_0$:

$$\beta_{t+1} = \beta_t - F(\beta_t)^{-1} \nabla_\beta L(\beta, \alpha), \tag{11}$$

where $F(\beta)$ is the Fischer information, defined as

$$\forall \beta,\ F(\beta) := \mathbb{E}[\nabla_\beta L(\beta, \alpha) \otimes \nabla_\beta L(\beta, \alpha)].$$

I.e., at each iteration, we need to solve

$$F(\beta_t) d_t = \nabla_\beta L(\beta, \alpha), \tag{12}$$

and apply

$$\beta_{t+1} = \beta_t - d_t.$$

In the present setting, we have

$$F(\beta) = X^\top W(\beta) X,$$

with

$$W(\beta) = \mathrm{diag}\left(\frac{\mu(\beta)}{1 + \alpha\mu(\beta)}\right),$$
$$\mu(\beta) = \gamma \odot \exp(X\beta).$$

Whenever IRLS fails to converge, L-BFGS-B [19] is used as an alternative.

## Detecting and replacing outliers

Cook's distances [18] are used to assess whether any single read count value has had a disproportionately strong influence on the final dispersion and LFC values. In that case, this read count is replaced with a trimmed mean imputation, and dispersions and LFCs are refit.

***Computing Cook's distances***

A method of trimmed moments is applied to estimate dispersions based on counts from which extreme values where removed :

$$\tilde{\alpha} := \max((\bar{\mu}) - \tilde{v})/\bar{\mu}^2, 0),$$

where $\bar{\mu}$ is the empirical mean (i.e., the average of normalized counts), and $\tilde{v}$ is a trimmed variance, i.e., the empirical variance after a fixed fraction of the largest and smallest counts were removed.

For a gene $g$ and sample $i$, Cook's distance for GLMs is then given by [see 4, 18]:

$$D_{ig} = \frac{R_{ig}^2}{p} \frac{h_{ii}}{(1 - h_{ii})^2},$$

where

$$R_{ig} = \frac{Y_{ig} - \mu_{ig}(\beta)}{\sqrt{\mu_{ig}(\beta) + \tilde{\alpha}\mu_{ig}(\beta)^2}}$$

is the Pearson residual of sample j and $\mu_{ig}(\beta) = \gamma_i \exp(\langle X^{(k)}, \beta_g \rangle)$, and $h_{ii}$ is the $i$-th diagonal element of the hat matrix $H$:

$$H = W^{1/2} X (X^\top W X)^{-1} X^\top W^{1/2} \in \mathbb{R}^{n \times n}.$$

***Replacing outliers***

Then, for all genes $g$ and samples $i$ for which $D_{ig}$ is greater than a given threshold (which depends only on the number of samples $n$ and the number of parameters $p$), $Y_{ig}$ is replaced with the trimmed mean of $g$'s counts over all samples, multiplied by $\gamma_i$. Dispersions and LFCs for all such genes are then refit, and the new values replace the old ones.

This is only applied to samples whose condition (i.e. whose covariates value combination) meet a minimal number of replicates (by default, 7).

## Computing and adjusting p-values

***Wald tests***

A p-value is computed for each gene using a Wald test, which consists in comparing the LFC coefficient of interest $\beta_{gr}$, divided by its standard error $\sqrt{\Sigma_{rr}^{(g)}}$, to a normal distribution (mulitplied by 2 to achieve a two-tailed test). The following covariance matrix estimator is used (see [4, 38] for details):

$$\Sigma^{(g)} = \mathrm{Cov}(\beta_g) = (X^\top W X + \lambda I)^{-1} (X^\top W X)(X^\top W X + \lambda I)^{-1} \in \mathbb{R}^{p \times p}. \tag{13}$$

***P-value Cook filtering***

Cook's distances are then used again to filter out p-values for genes which have remaining outlier count values. More precisely, genes which still have at least one sample with a Cook's distance above the threshold have their p-value set to `NaN`, unless there are 3

samples or more whose read counts are above those of the maximum Cook's distance sample.

#### *P-value adjustment*

Finally, p-values are adjusted by applying the Benjamini-Hochberg procedure [39] and independent filtering (see [4]). We do not detail these steps here: in a federated setting, they can be performed by the server using quantities from previous steps. Hence, they do not require any federation effort.

## Federating `DESeq2`

Let us now assume that the data is partitioned into $K$ different centers $\{1, ..., K\}$ which cannot share their data. Let $n_1$, ... ,$n_K$ be the number of data samples in each center, $X^{(1)}, ..., X^{(K)}$ the corresponding design matrices, and $Y^{(1)}, ..., Y^{(K)}$ the corresponding count matrices. In particular, for $1 \leq k \leq K$, we have $X^{(k)} \in \mathbb{R}^{n_k \times p}$ and $Y_k \in \mathbb{R}^{n_k \times G}$.

The objective remains the same: from $(X^{(k)}, Y^{(k)}), 1 \leq k \leq K$, we wish to estimate, for each gene $g$, a dispersion $\alpha_g \in \mathbb{R}_+$, a log fold change vector $\beta_g \in \mathbb{R}^p$ and a p-value in $(0, 1)$.

However, we now have the additional constraint that centers $\{1, ..., K\}$ are not allowed to share individual data between themselves or with the server. In particular, this implies that neither the full design matrix nor the full counts matrix may be formed.

As in most FL settings, we will overcome this obstacle by leveraging the fact the negative likelihood function (NLL), which we minimize to fit dispersions and LFCs, is *separable*. That is, the total NLL is the sum of each center's local NLL: given a gene $g$, it holds

$$L(\beta_g, \alpha_g) = \sum_{k=1}^{K} L_k(\beta_g, \alpha_g), \tag{14}$$

where $L_k(\beta_g, \alpha_g) := \sum_{i=1}^{n_k} \ell(Y_{ig}^{(k)}; X_i^{(k)}, \gamma_i | \alpha_g, \beta_g), 1 \leq k \leq K$ are the local NLLs. From there, we can compute the gradient of the NLL (w.r.t either dispersions or LFCs) from local gradients, and perform (stochastic) gradient descent:

$$\nabla L(\beta_g, \alpha_g) = \sum_{k=1}^{K} \nabla L_k(\beta_g, \alpha_g). \tag{15}$$

Note, however, that for communication efficiency reasons, in most FL optimization techniques such as FedAvg [10] and its variants, the full gradient (Equation 15) is never formed. Instead, several (stochastic) gradient descent steps are performed locally in each center, before the server averages the local values [see 11].

### Federating size factor estimation

We obtain the pseudo reference sample (Equation 1) by computing a federated mean of log-counts: each center communicates $S_k := \sum_{i=1}^{n_k} \log(Y_{ig}^{(k)})$ to the server, which then forms

$$R_g = \exp\left(\frac{1}{n}\sum_{k=1}^{K} S_k\right).$$

Finally, the server shares back $R_g$ to the center, which can then compute their size factors $\gamma_i := \text{median}_g\left(\frac{y_{ig}}{R_g}\right)$ locally.

### Federating initial dispersions estimates

Method-of-moments estimates (Equation 2) are derived by computing global means using the same strategy as for size factors (c.f. above).

To compute "rough dispersion estimates", the linear system in Equation 3 is first solved by forming the global Gram matrix $X^\top X \in \mathbb{R}^{p\times p}$ and the global features $X^\top(Y \oslash \gamma) \in \mathbb{R}^p$ from local Gram matrices $(X^{(k)})^\top X^{(k)}, 1 \leq k \leq K$ and local features $(X^{(k)})^\top(Y^{(k)} \oslash \gamma^{(k)}), 1 \leq k \leq K$ :

$$X^\top X = \sum_{k=1}^{K} (X^{(k)})^\top X^{(k)}$$
$$X^\top(Y \oslash \gamma) = \sum_{k=1}^{K} (X^{(k)})^\top (Y^{(k)} \oslash \gamma^{(k)}).$$

$\hat{\beta}_g$ may then be derived as the solution of $X^\top X\beta = X^\top(Y \oslash \gamma)$. From there, rough dispersion estimates are obtained by computing federated means (c.f. Equation 4).

### Federating initial $\mu$ estimates

As mentioned in the `DESeq2` workflow section, initial $\mu$ estimates are obtained using IRLS. We refer to the subsection on federating LFC computation for a federated version of IRLS.

### Fitting dispersions in a federated setting

In `DESeq2`, the dispersions are fitted by minimizing the NLL with fixed log fold change twice: in the genewise dispersions step, and in the MAP dispersions step. In `DESeq2`, these are fitted by minimizing the NLL using line search. As an implementation of line search in a federated setting is challenging, we instead resort to a simple grid search.

The choice of a grid search is motivated by three main facts:

- the underlying optimization problem is one-dimensional;
- although its objective function is not convex (because of the Cox-Reid term, equation 6), we empirically observe that it has a single optimum;
- in a federated setting, computational costs are dominated by communication costs.

Indeed, since the optimization problem is one dimensional, performing a grid search is a relatively effective way of finding the minimum.

To apply a grid search in a federated setting, we must compute the global adjusted NLL (i.e. the NLL plus the Cox-Reid adjustment term, see equation 5) evaluated on the whole federated data at each point of the grid.

The NLL term can be straightforwardly evaluated leveraging its data separability. Indeed, if each center computes its local $L_k(\alpha, \beta)$ for all $\alpha$'s on the grid and communicates it to the server, the server may in turn derive the global $L(\alpha, \beta)$ values by summation (see equation 14).

The Cox-Reid adjustment term (equation 6), however, cannot be obtained directly from a simple summation. Instead, the server computes the global argument of the log-determinant, $F(\alpha) := X^\top W(\alpha) X$ as

$$F(\alpha) = \sum_{k=1}^{K} F_k(\alpha),$$

where

$$F_k(\alpha) := (X^{(k)})^\top W_k(\alpha) X^{(k)},$$

by having each center compute and communicate local $F_k(\alpha)$ matrices at each point of the grid, and then applies the log-determinant.

The aggregator then sums all the terms and computes the global loss on all grid points in $[\alpha_{\min}, \alpha_{\max}]$, and selects the dispersion on the grid point $\alpha^\star_{\text{grid}}$ which minimizes the loss.

This value is then refined by re-iterating the process: a new grid is defined on $[\alpha^\star_{\text{grid}} - \delta, \alpha^\star_{\text{grid}} - \delta]$, on which a new optimum is sought.

By default, `FedPyDESeq2` repeats this process three times on grids of size 100, in logarithmic space.

From a computational point of view, this is very cost-effective: indeed, the cost of locally computing 100 adjusted NLL for a batch of genes is negligible compared with the cost of transferring data back and from the centers to the server. By limiting these exchanges to three rounds, iterative grid search achieves much faster convergence than more classic gradient-based FL methods such as FedAvg [10] would.

#### *Gene-wise estimates*

Gene-wise estimates $\alpha^{\text{GW}}$ are derived using the above strategy.

#### *Fitting a trend curve to gene-wise dispersion estimates and computing the variance prior*

This step does not require any particular federation effort: it can be performed on the server from the gene-wise estimates and means of normalized counts.

#### *MAP estimates*

As for gene-wise estimates, MAP estimation is performed using grid search, with an additional $\frac{(\log \alpha - \log \alpha_g^{\text{tr}})^2}{2\nu^2}$ shrinkage term. This term is computed directly on the server from the trend curve dispersion estimates $\alpha_g^{\text{tr}}$ and the prior variance $\nu$.

## Fitting log-fold changes in a federated setting

In `DESeq2` [4], two steps require the log fold changes $\beta_g$ to be fitted with negative log-likelihood minimization:

- to obtain initial $\hat{\mu}$ estimates;
- to compute the final log fold changes, estimated from the final dispersions.

`DESeq2` performs the optimization using IRLS, and switches to LBFGS in case of failure.

In the federated setting, we adopt the same approach. We first try to apply the federated IRLS algorithm to all genes. This algorithm is pooled-equivalent: it outputs exactly the same results as in the pooled setting (see the next paragraph). However, for genes for which IRLS has not converged, we do not use LBFGS or LBFGS-B, which are harder to federate and are more adapted to a setting where communication bottlenecks do not exist. Instead, we use a Proximal Quasi Newton method [40] with a line search in order to minimize the number of communication rounds, which is the main bottleneck for cross-silo FL.

### *Federating the IRLS algorithm*

In a nutshell (c.f. the `DESeq2` workflow section), IRLS consists in iteratively solving

$$F(\beta_t)d_t = \nabla_\beta L(\beta, \alpha), \tag{16}$$

for $d_t$, and applying

$$\beta_{t+1} = \beta_t - d_t.$$

In a federated setting, we cannot obtain the update $d_t$ as the sum of local values. Indeed, $F(\beta_t)^{-1}$ is not separable (i.e. linearly decomposable w.r.t centers). Instead, at each iteration, we form $F(\beta)$ and $\nabla_\beta L(\beta, \alpha)$ on the server using local quantities shared by the centers, and solve equation 12.

Because of the separability of NLL (equation 15), it holds

$$\nabla_\beta L(\beta, \alpha) = \sum_{k=1}^{K} \nabla_\beta L_k(\beta, \alpha).$$

Hence, the global gradient can be computed by the server from local (partial) gradients sent by the centers.

Next, define local, per-center $W_k \in \mathbb{R}^{n_k \times p}$ and $\mu_k \in \mathbb{R}_k^n$ as $W_k(\beta) = \text{diag}\left(\frac{\mu_k(\beta)}{\mu_k(\beta)+\alpha^{-1}}\right)$ and $\mu_k(\beta) = \gamma \odot \exp(X^{(k)}\beta)$, and let

$$F_k(\beta) = (X^{(k)})^\top W_k(\beta) X^{(k)}.$$

We observe that

$$\sum_k F_k(\beta) = \sum_k (X^{(k)})^\top W_k(\beta) X^{(k)} = X^\top W(\beta) X = F(\beta).$$

Hence, to federate Equation 11, it suffices to

1. have centers compute local $F_k(\beta)$ and $\nabla_\beta L_k(\beta, \alpha)$ and send them to the server;
2. make the server
   (a) form $F(\beta) = \sum_k F_k(\beta)$ and $\nabla_\beta L(\beta, \alpha) = \sum_{k=1}^{K} \nabla_\beta L_k(\beta, \alpha)$,
   (b) compute $d_t$ as the solution of $F(\beta)d = \nabla_\beta L(\beta, \alpha)$,
   (c) and output $\beta_{t+1} = \beta_t - d_t$.

***The federated Prox Quasi Newton method***

When IRLS iterations start diverging, we use instead [40], Algorithm 1, to fit LFCs. Federating this algorithm simply requires to federate the computation of the gradient scaling matrix. In our case, the inverse gradient scaling matrix is the Fisher information matrix $F(\beta)$, with an additional $\tau I$ regularization term, which varies along the iterations: $\tau$ starts at 1 and ends at $10^{-6}$ at the final iteration. Indeed, using the inverse Fisher information as a gradient scaling matrix is only relevant close to the optimum. Using large $\tau$ at the first iterations makes the inverse scaling close the identity matrix, and the iterations behave like gradient descent. However, in order to account for the specificity of the federated setting, we introduce a slight modification compared to [40], Algorithm 1. Instead of performing a line search, we actually compute the gradient and gradient scaling matrices at all potential next steps, so that we do not need to iterate. Instead, we simply select the best step size, that is, the largest step size for which the Armijo criterion [41] is satisfied. This reproduces what would have happened if we had actually run the line search, while cutting center to server communication time.

Thus, at each step, each center computes all next potential gradients, all next potential local Fisher information, and all next potential local NLLs. The server then forms all possible gradients, Fisher information, and NLLs. Finally, it selects the largest step size satisfying the Armijo condition, and returns the corresponding ascent direction (computed using 2.4 and 2.5 of [40]).

## Detecting and replacing outliers in a federated setting

As seen in the description of the `DESeq2` pipeline, computing Cook's distances to detect outliers and replacing those outliers with imputed values requires computing trimmed means and variances, i.e. means and variances of counts data from which fixed upper and lower quantiles were removed. This implies designing a federated algorithm to compute data quantiles.

To approximate $r$ and $1 - r$ quantiles, we adopt a dichotomy method. Here, we assume the gene and quantile to be fixed: in the code, we perform this dichotomy for all genes and the 2 quantiles in parallel.

Each center first shares its minimum and maximum normalized counts with the server, which in turn computes the global minimum and maximum. This provides an initial upper and lower bound on the quantile values.

Then, at each iteration, we replace one of the upper and lower bounds by the average of the two, thus dividing the uncertainty in half.

To do so, each center counts the number of samples that is above this mean value. The aggregator then sums these counts and assesses wether the total is above or below the quantile target value ($r \times n$, where $n$ is the total number of samples). If it is above,

it replaces the lower bound by the average, if is below, it replaces the upper bound instead. This procedure is then repeated.

The number of dichotomy iterations thus performed is fixed in advance, with a default number of 40. In particular, this implies that the true quantile value would be theoretically found up to a precision of $(\max - \min)/2^{40}$. Hence, in most cases, machine precision is actually the limiting factor.

## Computing and adjusting p-values in a federated setting

### *Wald tests*

For a given gene, Wald tests require two quantities: the LFC coefficient $\beta_r^{(g)}$, and its standard error $\sqrt{\Sigma_{rr}^{(g)}}$. $\beta_r^{(g)}$ was computed during the IRLS step, and $\sqrt{\Sigma_{rr}^{(g)}}$ can be derived from $X^\top W X$ (see equation 13), which is computed from local $(X^{(k)})^\top W_k X^{(k)}, 1 \leq k \leq K$.

### *P-value Cook filtering*

Following the procedure described in the `DESeq2` methodology, we must identify genes which have

- (i) at least one sample with a Cook's distance above the threshold;
- (ii) fewer than 3 samples whose read counts are above the maximum Cook's distance sample;

and set their p-value to `NaN`.

To achieve (i), centers share whether each gene has remaining Cook outliers, and the server then compiles the total list.

Achieving (ii) requires several steps. First, for each of the genes from (i), we must find the sample with the maximum Cook's distance, and the corresponding read count value. To do so, centers share (for each gene) the maximum Cook's distance over their local samples, and the read count of the corresponding sample. The server then derives the global maximum for each gene, and its corresponding counts value threshold, which it shares back to centers.

Second, for each gene, we must count the number of samples above the threshold. This is first performed locally. Centers then share their local numbers of samples above the threshold, which the server sums.

### *P-value adjustment and independent filtering*

P-value adjustment and independent filtering can be performed from p-values and (global) means of normalized counts. As those quantities are computed as part of previous steps, no particular federation effort is required.

## Meta-analysis baselines

Meta-analysis [8] is a set of methods which aim to combine the effects measured by studies with identical or similar designs effect, and derive global effect sizes and variances which can, in turn, be employed to compute a global p-value.

In particular, in the presence of data silos, meta-analysis can be used as a surrogate for federated methods. Indeed, centers need only to share the LFCs and p-values they obtained by running DEA on their local data, and do not have to share any individual RNA transcript or covariate.

We now quickly review the meta-analysis baselines to which we compared the output of `FedPyDESeq2`. Each of the methods described below is first applied independently to each gene, and we then perform the Benjamini-Hochberg procedure [39] to control the false discovery rate.

## P-value combination

P-value combination consists in building a meta-statistic from the set of study (center) p-values, on which a global test is then performed. Note however that these methods only combine p-values, and not the effects (in our case, LFCs). As LFC estimation is a crucial aspect of DEA, we use the average of LFCs weighted by sample size, for lack of a better estimator. P-value combination methods differ in the manner in which the meta-statistic is built. In the following two paragraphs, we describe Stouffer's and Fisher's methods (see., e.g., [24] for a description in the context of DEA).

In both cases, we relied on `scipy`'s `combine_pvalues` implementation for our experiments.

### *Stouffer's method*

Let $Z_k^{(g)} := \phi^{-1}(1 - p_k^{(g)})$ where $\phi^{-1}$ is the inverse cumulative distribution function of the standard normal distribution. Then, under the null hypothesis, it holds

$$N^{(g)} := \sum_{k=1}^{K} \sqrt{\tfrac{n_k}{n}} Z_k^{(g)} \sim \mathcal{N}(0, 1).$$

### *Fisher's method*

Under the null hypothesis, it holds

$$F^{(g)} := -2 \sum_{k=1}^{K} \ln p_k^{(g)} \sim \chi^2(2K).$$

## Fixed-effects model

In a fixed-effects model, we make the assumption that the observed LFCs are distributed around one common true LFC, with a measurement error which is informed entirely by the within-study variance (see, e.g., [42]):

$$\tilde{\beta}_{ig} = \beta_g + \varepsilon_{ig}, \quad \varepsilon_{ig} \sim \mathcal{N}(0, \sigma_{ig}^2). \tag{17}$$

where

- $\tilde{\beta}_{ig}$ is the LFC estimated for gene g in $i$-th study, i = 1,. . . ,k;

- $\sigma_{ig}^2$ is the within-study variability representing sampling errors conditional on the $i$-th study;
- $\beta_g$ is the true global LFC across datasets for each gene;

The common LFC is estimated as $\widehat{\beta} = \frac{\sum_{i=1}^{K} \tilde{w}_i \tilde{\beta}_i}{\sum_{i=1}^{K} \tilde{w}_i}$, where $\tilde{w}_i = \frac{1}{\sigma_i^2}, 1 \leq i \leq K$. P-values are then derived thanks to a Wald test, using $\left(\sum_{i=1}^{K} \tilde{w}_i{}^2\right)^{-\frac{1}{2}}$ as standard error estimate.

### Random-effects model

Compared with fixed-effects models, random-effects models incorporate a term representing between-study variability (see, e.g., [42]):

$$\tilde{\beta}_{ig} = \theta_{ig} + \varepsilon_{ig}, \quad \varepsilon_{ig} \sim \mathcal{N}(0, \sigma_{ig}^2), \tag{18}$$

$$\theta_{ig} = \beta_g + \delta_{ig}, \quad \delta_{ig} \sim \mathcal{N}(0, \tau_g^2), \tag{19}$$

where

- $\tilde{\beta}_{ig}$ is the LFC estimated for gene g in $i$-th study, i = 1,. . . ,k;
- $\theta_{ig}$ is the true LFC in the $i$-th study;
- $\sigma_{ig}^2$ is the within-study variability representing sampling errors conditional on the $i$-th study;
- $\beta_g$ is the true global LFC across datasets for each gene;
- $\delta_{ig}$ is the random effect;
- and $\tau_g$ is the between-study variability.

We use $\beta(\tau^2) = \frac{\sum_{i=1}^{K} w_i(\tau^2)\tilde{\beta}_i}{\sum_{i=1}^{K} w_i(\tau^2)}$, where $w_i(\tau^2) = \frac{1}{\sigma_i^2+\tau^2}, 1 \leq i \leq K$, as an estimate of the true LFC $\beta_g$. P-values are computed thanks to a Wald test, using $\left(\sum_{i=1}^{K} w_i(\tau^2)^2\right)^{-\frac{1}{2}}$ as standard error estimate.

The two random effects methods that we employ here differ in the manner in which they estimate the between-study variability $\tau_g^2$. In both cases, we rely on `statsmodels`'s `combine_effects` implementation.

#### *DerSimonian and Laird (DL) one-step estimator [23]*

The DL estimator estimates the between-study variability as

$$\widehat{\tau}_{DSL(g)}^2 := \max\left\{0, \frac{Q_g - (K-1)}{S_{1g} - (S_{2g}/S_{1g})}\right\},$$

where $Q_g = \sum_{i=1}^{k} w_{ig}\left(\tilde{\beta}_{ig} - \widehat{\beta}_g\right)^2$, $\tilde{w}_{ig} := \sigma_{ig}^{-2}$, $\widehat{\beta}_g = \frac{\sum_{i=1}^{k} \tilde{w}_{ig}\tilde{\beta}_{ig}}{\sum_{i=1}^{k} \tilde{w}_{ig}}, S_{1g} = \sum_{i=1}^{k} \tilde{w}_{ig}$ and $S_{2g} = \sum_{i=1}^{k} \tilde{w}_{ig}^2$.

#### *Paule-Mandel iterative estimator [22]*

The between-study variability is estimated by iteratively solving for $\tau^2$ the following equation:

$$F(\tau^2) := \sum_{i=1}^{K} w_i(\tau^2)[\tilde{\beta}_i - \beta(\tau^2)]^2 - (K-1) = 0.$$

To do so, the following update is applied until convergence:

$$\tau^2 \leftarrow \tau^2 + F(\tau^2) \sum_{i=1}^{K} \frac{(\tilde{\beta}_i - \beta(\tau^2))^2}{(w_i(\tau^2))^2}.$$

### Largest cohort

We include a final baseline, which consists in using the results output by `PyDESeq2` on the center with the largest cohort. This baseline serves as a sanity check, to confirm that employing federated methods or meta-analysis does bring a benefit compared to individual local results.

## Datasets and preprocessing

Let us now describe the datasets that were used for experiments, and the preprocessing steps they underwent. We used data from The Cancer Genome Atlas (https://www.cancer.gov/tcga) corresponding to 8 different indications:

- TCGA-BRCA: Breast invasive carcinoma
- TCGA-COAD: Colon adenocarcinoma
- TCGA-LUAD: Lung adenocarcinoma
- TCGA-LUSC: Lung squamous cell carcinoma
- TCGA-PAAD: Pancreatic adenocarcinoma
- TCGA-PRAD: Prostate adenocarcinoma
- TCGA-READ: Rectum adenocarcinoma
- TCGA-SKCM: Skin cutaneous melanoma

For each indication, we collected two types of data: (i) bulk RNA-seq raw counts (not restricted to protein-coding genes, for a total of 60k+ genes) and (ii) sample metadata, to be used in the design of our DEA experiments. Both were downloaded from the `recount3` database [43].

#### *Read count preprocessing*

Counts are indexed by TCGA sample barcode, with columns corresponding to the gene_id in ENSEMBL convention. PAR_Y genes are filtered out, as they are not common to all samples.

#### *Metadata preprocessing*

From the metadata, we extract the following factors:

- Patient `gender`;
- Tumor `stage`. Originally from I to IV, we transform `stage` into a binary covariate by grouping stages I-II-III (`Non-advanced`) against IV (`Advanced`);
- `CPE` (for "Consensus measurement of purity estimation", an aggregate of different purity estimations obtained from [44], taking values between 0 and 1;

We also extract tissue source sites (TSS) from the TCGA barcodes: see "Geographical split" below.

Samples corresponding to normal (non-tumoral) tissues are removed from our experiments. Likewise, any sample with missing design covariate information is dropped from the corresponding experiment (c.f. "Experimental designs" below).

Scripts to obtain the data, including download and preprocessing, are available in the following repository: https://github.com/owkin/fedpydeseq2-datasets.

### *Experimental designs*

In the geographical split setting, we perform experiments with three different designs:

- A single-factor design, with tumoral `stage` as sole design factor;
- A multi-factor design with categorical covariates, including tumoral `stage` and `gender` as factors. Because of the gender covariate, TCGA-BRCA and TCGA-PRAD are excluded from this experiment;
- A multi-factor design with a numerical covariate, including tumoral `stage` and `CPE` as factors. TCGA-PAAD is excluded from this experiment, as CPE scores are unavailable.

In the case of the synthetic heterogeneous split, only the single-factor design was used. For all three designs, we tested `Advanced` compared to `Non-Advanced` `stage` samples, with a LFC=0 null hypothesis.

| | stage | | gender | |
|---|---|---|---|---|
| | Non-advanced | Advanced | female | male |
| TCGA-BRCA | 1074 | 20 | 1082 | 12 |
| TCGA-COAD | 403 | 67 | 225 | 245 |
| TCGA-LUAD | 507 | 26 | 288 | 245 |
| TCGA-LUSC | 491 | 7 | 131 | 367 |
| TCGA-PAAD | 171 | 5 | 79 | 97 |
| TCGA-PRAD | 420 | 82 | 0 | 502 |
| TCGA-READ | 132 | 24 | 71 | 85 |
| TCGA-SKCM | 389 | 24 | 155 | 258 |

**Table 1** The (`stage`, `gender`) multi-factor metadata.

### *Geographical split*

Following the methodology introduced in [17], we simulate a realistic federated setting by separating samples according to the locations they were gathered from. More precisely, we extract TSS from TCGA barcodes (c.f. https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/). We then map each TSS

| | | Europe | Northeast (USA) | Midwest (USA) | South (USA) | West (USA) | Other | Total |
|---|---|---|---|---|---|---|---|---|
| TCGA-BRCA | 50 | 152 | 161 | 315 | 199 | 209 | 8 | 1094 |
| TCGA-COAD | 10 | 199 | 43 | 99 | 104 | 12 | 3 | 470 |
| TCGA-LUAD | 21 | 50 | 99 | 85 | 150 | 95 | 33 | 533 |
| TCGA-LUSC | 45 | 67 | 114 | 87 | 87 | 54 | 44 | 498 |
| TCGA-PAAD | 40 | 6 | 39 | 5 | 32 | 37 | 17 | 176 |
| TCGA-PRAD | 27 | 57 | 18 | 195 | 79 | 113 | 13 | 502 |
| TCGA-READ | 1 | 92 | 18 | 23 | 5 | 0 | 17 | 156 |
| TCGA-SKCM | 1 | 66 | 59 | 65 | 91 | 41 | 90 | 413 |

**Table 2** The single-factor and (`stage`, `gender`) multi-factor splits.

| | Canada | Europe | Northeast (USA) | Midwest (USA) | South (USA) | West (USA) | Other | Total |
|---|---|---|---|---|---|---|---|---|
| TCGA-BRCA | 50 | 152 | 160 | 309 | 191 | 205 | 8 | 1075 |
| TCGA-COAD | 10 | 199 | 43 | 98 | 99 | 12 | 3 | 464 |
| TCGA-LUAD | 21 | 50 | 99 | 83 | 149 | 95 | 33 | 530 |
| TCGA-LUSC | 45 | 67 | 114 | 87 | 87 | 54 | 44 | 498 |
| TCGA-PRAD | 27 | 57 | 18 | 195 | 79 | 108 | 13 | 497 |
| TCGA-READ | 1 | 92 | 18 | 23 | 5 | 0 | 17 | 156 |
| TCGA-SKCM | 1 | 66 | 59 | 63 | 89 | 41 | 90 | 409 |

**Table 3** The (`stage`, `CPE`) multi-factor split.

code to its source (https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/tissue-source-site-codes), which we group in 7 regions: Canada, Europe, Northeast (USA), Midwest (USA), South (USA), West (USA) and Other. The TSS to region correspondence file is available in the following repository: https://github.com/owkin/fedpydeseq2-datasets. Each region then acts as a federated data center in our experiments.

The corresponding splits are summarized in Tables 2 and 3. The differences between these two tables are due to the fact that we drop samples with no available `CPE` score.

Finally, let us note that in some centers, some conditions may have 0 replicates (e.g., they only contain samples with `Non-advanced` tumoral `stages`). In that case, as mentioned in the Discussion, it is not possible to run `PyDESeq2` locally on those centers. Hence, they are not included in meta-analysis baselines. However, `FedPyDESeq2` is still capable to take them into account.

### *Heterogeneous splits*

To better measure the effects of data heterogeneity between centers on our method, we generate two sets of 2-centric synthetic splits: TCGA-CRC, which is based on the union of TCGA-COAD and TCGA-READ, and TCGA-NSCLC, based on TCGA-LUAD and TCGA-LUSC.

For TCGA-CRC, we create two centers, and assign each TCGA-COAD and TCGA-READ sample to either according a binomial distribution with parameter $h/2$, $h \in \{0, 0.25, 0.5, 0.75, 1\}$. In particular, when $h = 1.0$, each center receives half of TCGA-COAD and half of TCGA-READ in expectation, whereas when $h = 0.0$, one of the centers contains all TCGA-COAD samples, and the other all TCGA-READ samples. The same procedure is applied for TCGA-NSCLC. To ensure that our results are reproducible, we fix a random seed (`np.random.seed(42)`). The corresponding splits are summarized in Table 4.

| Heterogeneity level | TCGA-CRC | | | | TCGA-NSCLC | | | |
|---|---|---|---|---|---|---|---|---|
| | Center 0 | | Center 1 | | Center 0 | | Center 1 | |
| | TCGA-COAD | TCGA-READ | TCGA-COAD | TCGA-READ | TCGA-LUAD | TCGA-LUSC | TCGA-LUAD | TCGA-LUSC |
| 0.0 | 470 | 0 | 0 | 156 | 533 | 0 | 0 | 498 |
| 0.25 | 407 | 25 | 63 | 131 | 454 | 58 | 79 | 440 |
| 0.5 | 358 | 43 | 122 | 113 | 387 | 110 | 146 | 388 |
| 0.75 | 287 | 58 | 183 | 98 | 320 | 165 | 213 | 327 |
| 1.0 | 229 | 76 | 241 | 80 | 253 | 261 | 280 | 237 |

**Table 4** The heterogeneous single-factor split.

## Evaluation

As the goal of `FedPyDESeq2` is to reproduce, from siloed data, the results of `PyDESeq2` on pooled data, we use the output of `PyDESeq2` on pooled data as a ground truth for our benchmarks, and perform evaluations both on `FedPyDESeq2` and our baselines.

### *Comparison of DEGs*

For each combination of dataset, split and experimental design, we compare the set of `FedPyDESeq2` and baslines DEGs to those of `PyDESeq2`, setting $|\text{LFC}| \geq 2$ and $\text{padj} < 0.05$ thresholds for differential expression.

The results for `FedPyDESeq2` on the geographical split with the single-factor design are summarized in Fig. 2 A in the form of a confusion matrix, where percentages are expressed w.r.t. `PyDESeq2` DEGs. Results for other experiments are included in the supplementary material.

### *Gene set enrichment analysis*

To further validate the output of `FedPyDESeq2`, we perform gene set enrichment analysis (GSEA) on DEGs identified by `FedPyDESeq2` and baselines on siloed RNA-seq, and compare the results to those obtained from pooled `PyDESeq2` DEGs.

We perform GSEA on MSigDB human reactome gene sets [45, 46] using `fgsea` [21] and DEA test statistics as gene ranking criterion.

### *LFC and adjusted p-value relative errors*

Finally, we report `FedPyDESeq2` and baseline LFC and adjusted p-value errors relative to pooled `PyDESeq2` LFC and adjusted p-values, that is:

$$\frac{|\beta_{\texttt{method}} - \beta_{\texttt{PyDESeq2}}|}{|\beta_{\texttt{PyDESeq2}}|},$$

and

$$\frac{|\text{padj}_{\texttt{method}} - \text{padj}_{\texttt{PyDESeq2}}|}{|\text{padj}_{\texttt{PyDESeq2}}|}.$$

## Deployment and runtime

Our experiments were performed by deploying `Substra` on a set on independent Google Cloud Platform (GCP) instances. For the geographical split, 8 machines were deployed (one for each of the 7 centers, plus one to act as central server). Likewise, 3 machines were used in the synthetic split experiments. Both the centers and the server were allocated `n1-standard-8` GCP instances with 8 virtual processors and 30 Gb of RAM

each. With this configuration, each DEA experiment ran in times ranging from 8 to 12 hours.

Note that for demonstration or debugging purposes it is also possible to run `FedPyDESeq2` in simulated mode on a single machine. On a modern laptop, the simulated mode of each experiment runs in about an hour.

# Declarations

## Software availability

The code for `FedPyDESeq2` is available at https://github.com/owkin/fedpydeseq2 under an MIT license. The code to download and preprocess TCGA datasets and to reproduce our experiments is available at https://github.com/owkin/fedpydeseq2-datasets and https://github.com/owkin/fedpydeseq2-paper-experiments.

## Data availability

The data used in this study can be downloaded from The Cancer Genome Atlas (https://www.cancer.gov/tcga) and processed using the following code: https://github.com/owkin/fedpydeseq2-datasets.

## Acknowledgment

The authors would like to thank Sarah Diot-Girard, Thibault Fouqueray and Thibault Camalon for their invaluable insights and support on how to use `Substra` for federated transcriptomics, Vincent Cabeli for his careful proofreading, and Jade Lum and Davide Mantiero for their help with the FL figure.

The results published here are in whole or part based upon data generated by the TCGA Research Network: https://www.cancer.gov/tcga.