Benchmarking Tensor-cell2cell's Decomposition Performance

To help users make informed decisions regarding choices in their computational pipeline, we benchmarked two key factors that can influence Tensor-cell2cell's outputs: batch effects and missing data (which result in missing tensor indices) across samples.

1. Strategies to Address Batch Effects

1.1 Motivation

This analysis aims to determine the impact of batch effects on the performance of the tensor decomposition used by Tensor-cell2cell. Tensor-cell2cell helps CCC analyses of datasets representing > 2 contexts. This means that we will be analyzing multiple samples, each of which contain variation in the data that are due to technical factors. Batch correction [1] removes technical variation while preserving biological variation between samples. Batch correction is an important consideration since Tensor-cell2cell analyzes multiple samples to extract context-dependent patterns, and we want to make sure we are capturing true biological signals rather than sample-specific differences due to technical variability.

Datasets with multiple replicates [2] can shed light on batch effects when using Tensor-cell2cell. Replicates allow us to ensure that the output factors are not simply due to technical effects. We can reasonably assume that the biological variation in samples between different contexts will be greater than that of those within contexts after using appropriate batch correction to remove technical variation. Thus, we expect Tensor-cell2cell to capture overall communication trends differing between contexts and can then assess that output factors are not simply due to technical effects by checking that the sample loading values in the factors' context dimension are similar for biological replicates and do not have high loadings for just one sample in the context dimension.

However, it is not always possible to have replicates, particularly technical replicates. Thus, it is important to understand the extent to which batch effects may change Tensor-cell2cell's results and how to appropriately handle them. Additional considerations:

1. CCC inference methods typically require a gene by cell counts matrix. Many batch correction methods only return a latent embedding, rather than a corrected counts matrix suitable for downstream analyse rather than a corrected counts matrix suitable for downstream analyses (see Table 1 of refs [3,4]).
2. Non-negativity: A non-negative counts matrix is required as input both for CCC inference (2a) and Tensor-cell2cell (2b). Most batch correction methods return corrected counts with negative values. Thus, if using one of these methods, one must typically replace negative values with 0.

a. Most CCC inference methods require non-negative counts as input for their scoring functions. Take the simple case of the product between a ligand and receptor. A negative count for both the ligand and the receptor indicates both are lowly expressed, however their product would be a positive value. If negative counts are retained in the expression matrix, we recommend using a CCC inference method that implements a scoring function that can handle this issue (e.g., an additive function).
b. Even if the CCC inference methods can appropriately handle negative counts, Tensor-cell2cell requires non-negative values for decomposition. Thus, if output communication scores are negative, Tensor-cell2cell will fill these negative counts with 0 or NaN (see the "Missing Indices" section for more details on these fill values). By default, if the tensor contains negative values, Tensor-cell2cell will treat these as 0 during decomposition.

We evaluated two batch correction methods to assess their utility for analyses with LIANA and Tensor-cell2cell: scVI[5] and Scanorama[6]. scVI implements a machine-learning approach and outputs non-negative values, whereas Scanorama implements a mutual nearest neighbors (MNN) approach and does not output non-negative values. Both methods output a corrected counts matrix and benchmarks have shown that they perform well[7].

Our analysis aimed to determine the extent to which batch effects affect Tensor-cell2cell's decomposition output, and whether batch correction methods can mitigate any observed deterimental affects. Moreover, we check whether introducing negative values to the corrected counts matrix decreases decomposition performance.

1.2 Results

Iterating across increasing levels of batch severity, we generated four counts matrices:

1) Gold-standard: a processed counts matrix with no batch effects
2) Log-normalized: a processed counts matrix with batch effects present
3) Scanorama batch-corrected: a processed counts matrix with batch effects corrected for using Scanorama
4) scVI batch-corrected: a processed counts matrix with batch effects corrected for using scVI

We ran the combined LIANA and Tensor-cell2cell pipeline on each of these counts matrices. Finally, we assessed the similarity between each of the decomposition outputs as follows:

- Log-normalized similarity: Similarity between Tensor-cell2cell's decomposition output from the log-normalized counts matrix (2) and that of the gold-standard input (1)
- Scanorama similarity: Similarity between Tensor-cell2cell's decomposition output from the Scanorama batch-corrected counts matrix (3) and that of the gold-standard input (1)
- scVI similarity: Similarity between Tensor-cell2cell's decomposition output from the scVI batch-corrected counts matrix (4) and that of the gold-standard as input (1)

Similarity was measured using the CorrIndex metric[8]. We assess batch severity using three performance metrics: clusterability using k-means clustering[9], clusterability using Louvain clustering [10] and mixability. For details, please see the Methods section.

1.2.1 Batch severity simulations perform as expected

We first ensure that the batch severity simulations are performing as expected. First, we check that the Splatter[11] parameters we iterate across (number of batches, batch scale, and batch location) change batch severity as expected. We assessed this for each batch severity metric and each input counts matrix using a multivariate regression with batch severity as the dependent variable and the Splatter parameters as the independent variables (**Fig. S1**). We expected that the batch severity of our gold-standard matrix is unchanged due to these parameters, since it should not contain batch effects. In contrast, the batch severity of the log-normalized matrix should increase with these parameters, which would be reflected in a positive coefficient estimate in the regression. Conversely, if batch correction performed well, we anticipated higher similarity with the gold-standard (or a lower rate of increase in severity) than that of the uncorrected log-normalized matrix, reflected in a smaller coefficient estimate in the regression.

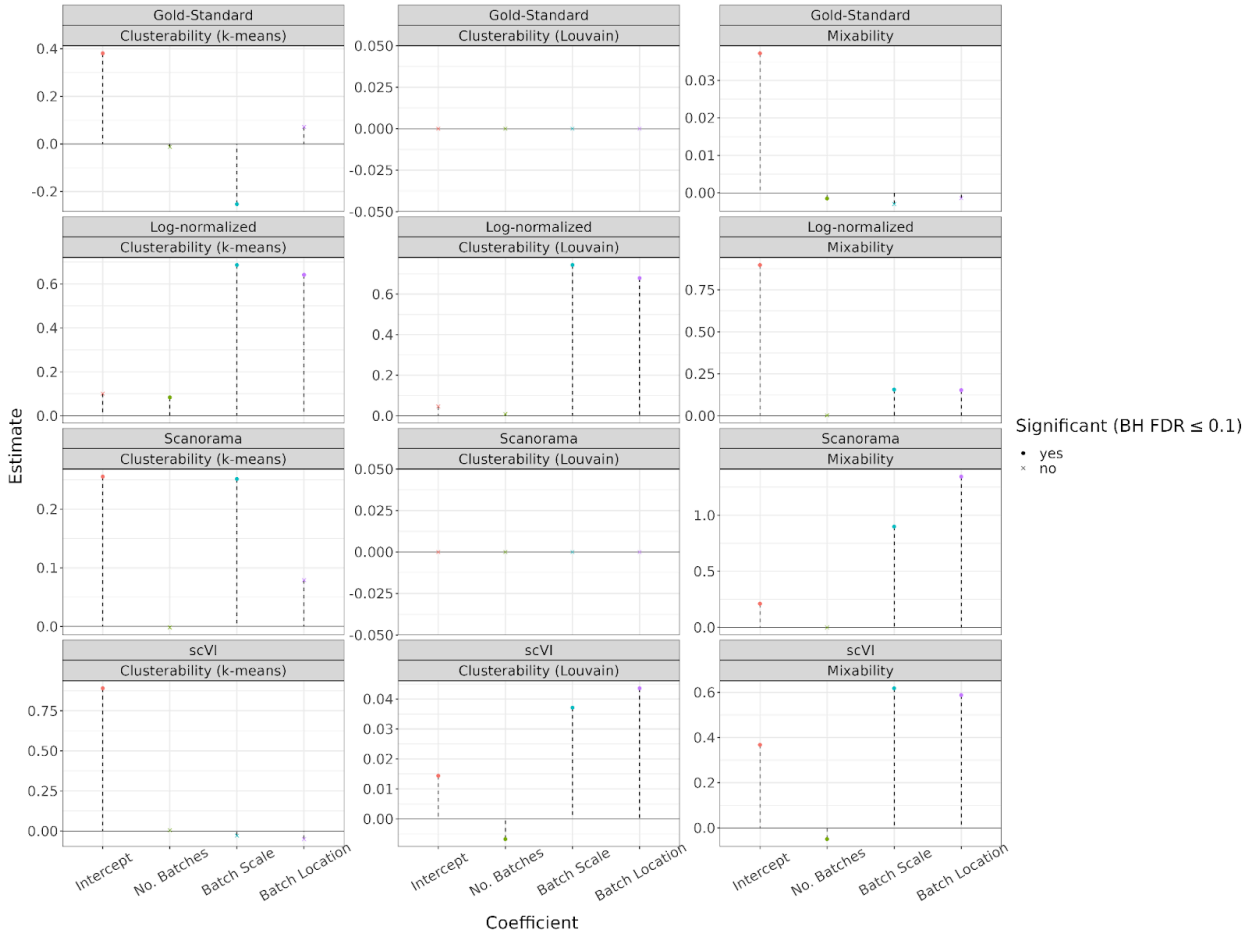

Figure S1. ***Effect of Splatter parameters on batch severity.*** *Coefficient estimates of linear regressions estimating batch severity (dependent variable) from the Splatter parameters used to introduce batch effects (independent variables). Panels represent distinct processed counts matrix inputs (rows) and batch severity metrics (columns). Coefficient estimates are on the y-axis, coefficient labels are on the x-axis, and the horizontal solid black line represents 0 (no effect). Significant estimates (Wald test, Benjamini-Hochberg FDR correction, q-value ≤ 0.1) are marked by a circle and insignificant estimates are*

marked by a cross. Note that the gold-standard and Scanorama batch-corrected matrices do not have significant estimates for the Louvain clusterability metric because clusterability values were 0 across all iterations.

Overall, we saw that the gold-standard matrix performed as expected, showing clear Louvain (**Fig. S3**) clusterability and little-to-no mixability (**Fig. S4**). For a discussion of the gold-standard k-means clusterability (**Fig. S2**), please refer to the proceeding section. The log-normalized matrix also performed as expected across all batch severity metrics. While the batch-corrected counts matrices increased along with the Splatter parameters on occasion, the increases were overall less severe than that of the log-normalized matrix (**Fig. S1-S4**).

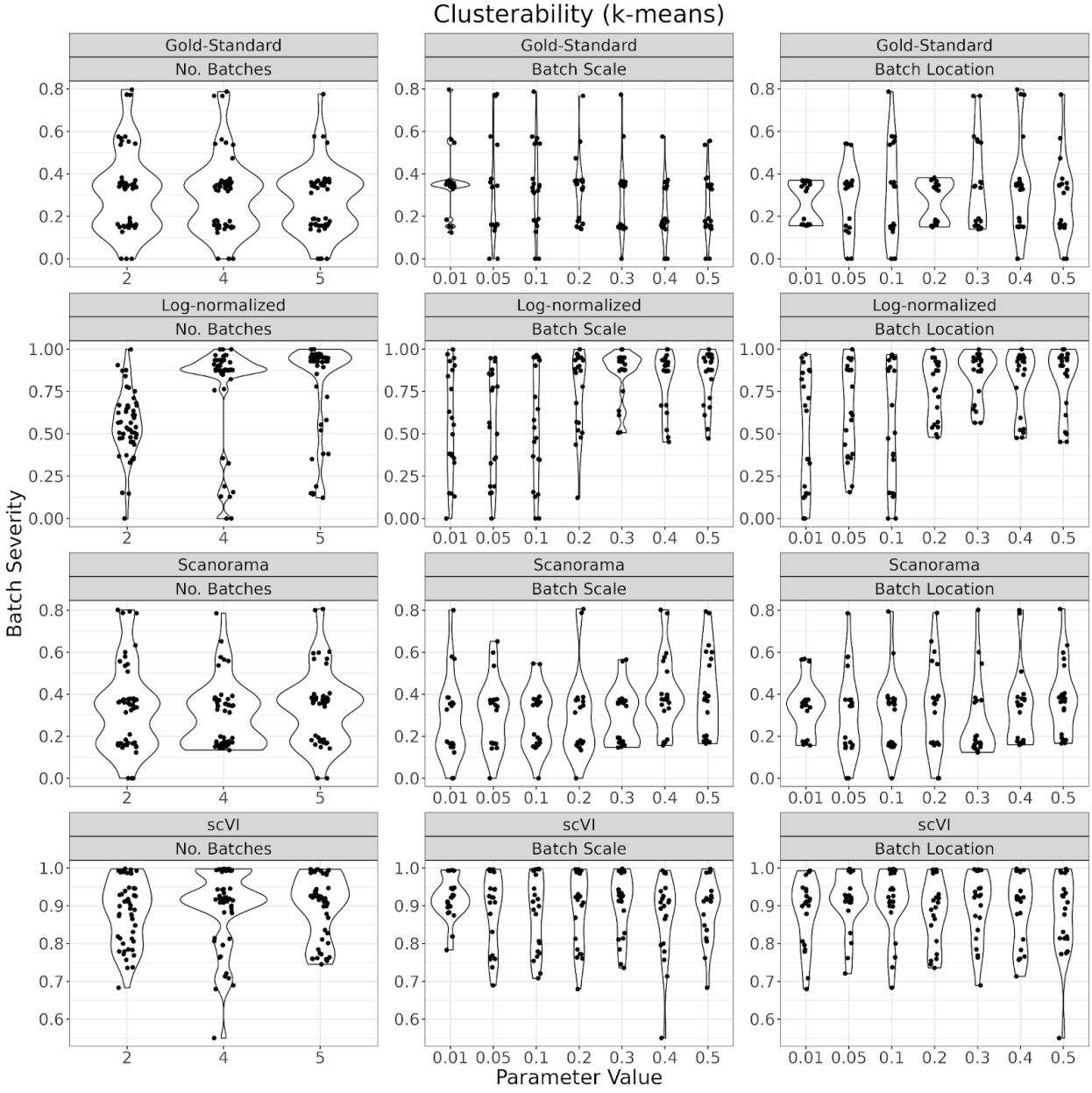

Figure S2. ***Effect of Splatter parameters on k-means clusterability.*** *Violin plots of batch severity (y-axis) as measured by k-means clusterability across iterations. Panels represent distinct processed counts matrix inputs (rows) and Splatter parameters (columns).*

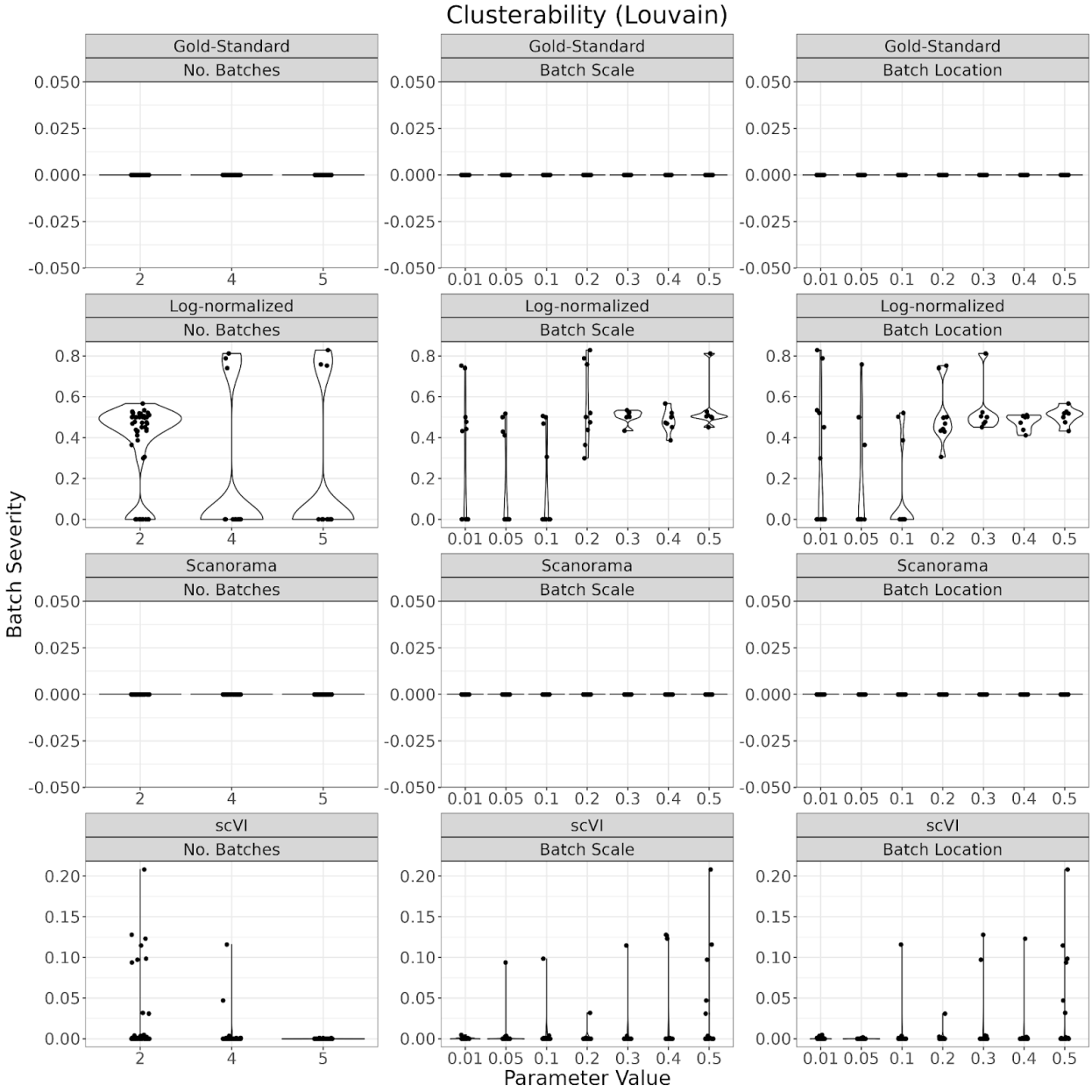

Figure S3. ***Effect of Splatter parameters on Louvain clusterability.*** *Violin plots of batch severity (y-axis) as measured by Louvain clusterability across iterations. Panels represent distinct processed counts matrix inputs (rows) and Splatter parameters (columns).*

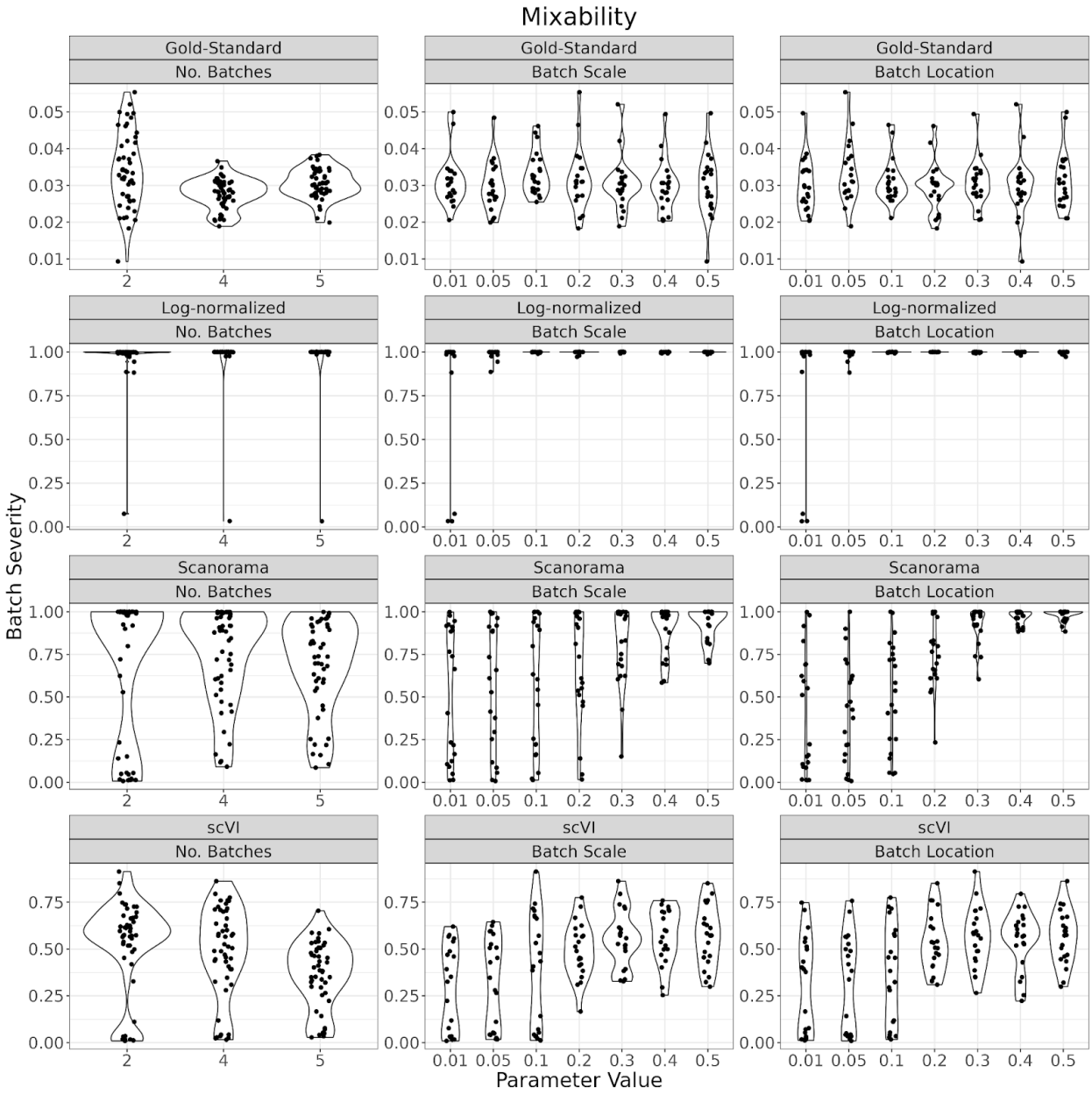

Figure S4. ***Effect of Splatter parameters on mixability.*** *Violin plots of batch severity (y-axis) as measured by mixability across iterations. Panels represent distinct processed counts matrix inputs (rows) and Splatter parameters (columns).*

Next, we check that the gold-standard counts matrices demonstrate comparably low batch severity across all iterations (**Fig. S5**). Louvain clusterability is 0 across all iterations (data not shown), and mixability also remains low across iterations (**Fig. S5b**). While k-means clusterability does demonstrate a wide range of values (**Fig. S5a**), the distribution skews towards the lower end of the range, which is similarly shown in the distribution of the k-means batch severity values across the Splatter parameters that are intended to increase it (**Fig. S2**). In contrast, the log-normalized matrix shows distributions weighted towards the higher end of

severity values across these parameters (**Fig. S2**). Statistically, this is reflected in the fact that, unlike the log-normalized matrix, the batch severity does not increase with these parameters (**Fig. S1**). Furthermore, across all iterations, the k-means batch severity of the gold-standard matrix is statistically significantly lower (Mann-Whitney U test, p-value << 1e-10) than the log-normalized matrix. Given all this evidence, we conclude that the k-means clusterability metric can be used in downstream analyses despite its wide range of values across iterations in the gold-standard counts matrix.

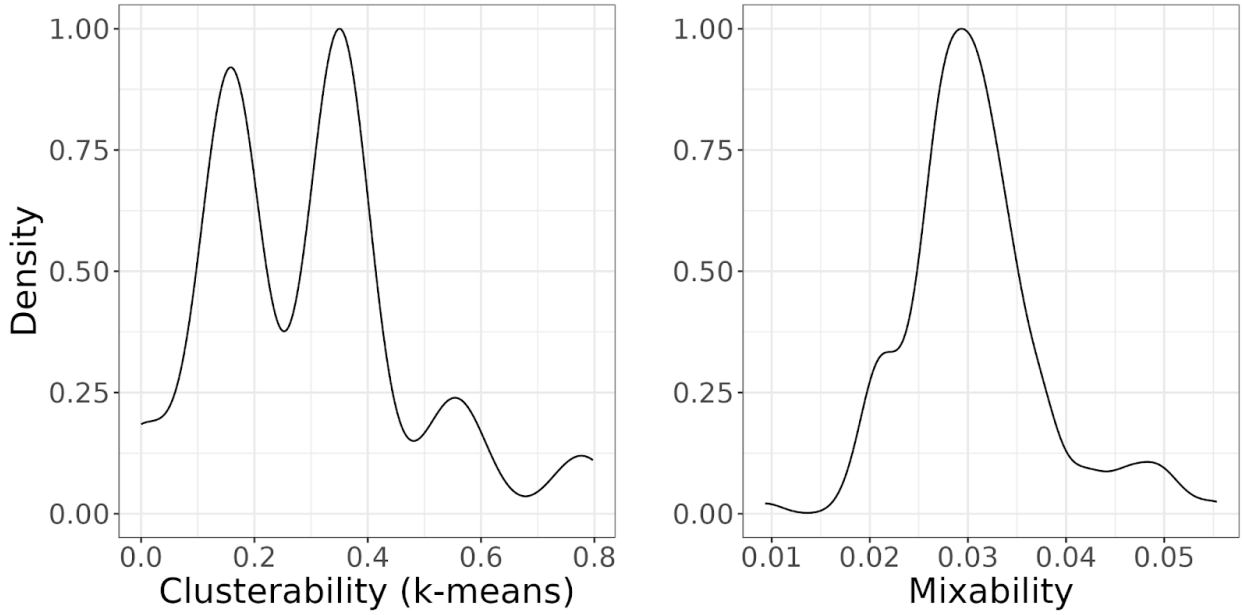

Figure S5. ***Batch-severity of the gold-standard counts matrix.*** *Scaled kernel density estimates (y-axis) visualizing distributions of batch-severity (x-axis) introduced to the gold-standard counts matrix across all iterations. Batch severity is measured by k-means clusterability (left panel) and mixability (right panel).*

1.2.2 Tensor-cell2cell Performance across Batch Severity without Correction

To answer this question, we assessed how similarity of decompositions (CorrIndex) across different batch severities changes with respect to the gold-standard log-normalized counts. At low levels of batch severity, we expect that the similarity to the log-normalized counts should be high, since the input counts matrices are similar. If Tensor-cell2cell is robust to batch effects, the similarity should not be substantially affected by increasing batch severity. In contrast, if batch effects distort Tensor-cell2cell's performance, we should see that similarity decreases with increasing batch severity.

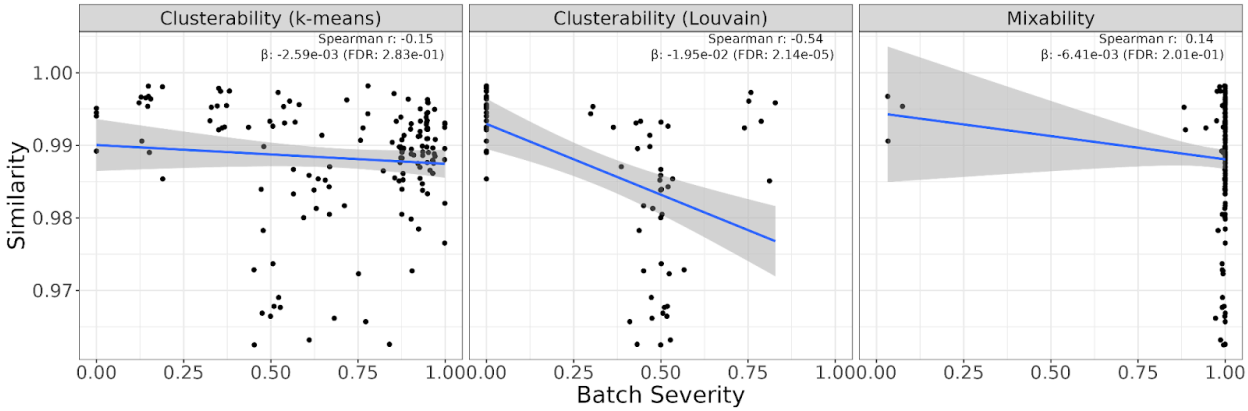

Figure S6. ***Tensor-cell2cell is robust to batch effects.*** *Scatterplots of log-normalized similarity (y-axis) across batch severity (x-axis) assessed by three metrics (panels). Blue lines show the regression fit with gray shaded regions showing the 95% confidence interval for this fit. Spearman correlations and regression coefficient estimates (as well as their Wald test FDRs) are annotated in the upper right corner of each panel.*

We saw no significant association of batch severity when considering clusterability (k-means) and mixability. Yet, we observed a correlation when considering clusterability (with Louvian). Nevertheless, we saw that across all batch severity metrics, similarity does not decrease beyond 0.963, indicating that Tensor-cell2cell is robust to batch effects (**Fig. S6**). While this is reassuring and tells us that batch correction may not be necessary for recovering biological signals using Tensor-cell2cell, in the next section, we proceed with assessing batch correction for the instances that users feel that it is necessary.

1.2.3 Tensor-cell2cell Performance with Replaced Negative Values

Of the two batch correction methods we used, Scanorama introduces negative values to the counts matrix. In our analysis pipeline, we replaced these values with 0 (see the “Motivation” section here as well as in the “Missing Indices” section to refer to discussions regarding this). We want to understand how the extent to which introducing these negative values affects Tensor-cell2cell for the following reasons: First, it allows us to assess whether these negative values act as a confounder when assessing Question 3B in the proceeding section. Second, most batch correction methods that can return corrected counts matrices introduce negative values; thus, knowing the consequences will help users understand how their batch correction method choice can affect results.

To determine whether it will be a confounder, we check whether the fraction of negative counts correlates with batch severity. The limited correlation between the two variables (**Fig. S7a-b**), as well as the insignificant regression coefficient estimates (**Fig. S7c**) suggests that the fraction of negative values introduced to the counts matrix does not change with batch severity levels and will not confound our downstream analyses of batch effect correction. Note, a scatter plot for

clusterability using Louvain is not visualized because all values are 0.

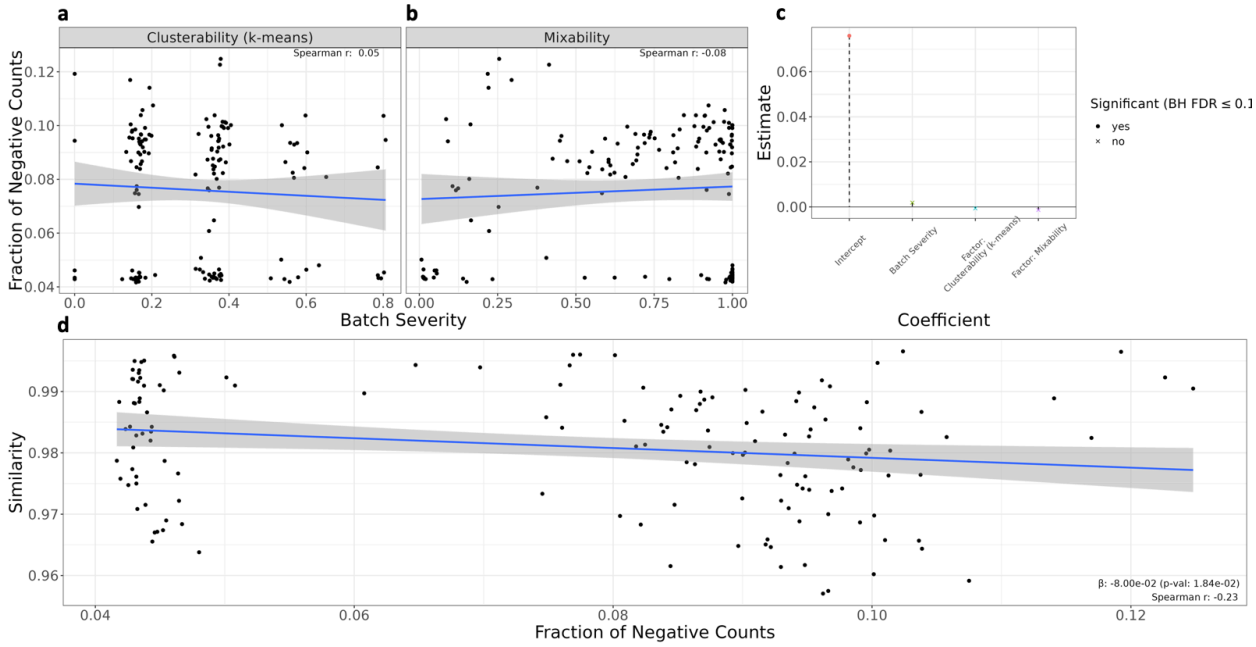

Figure S7. ***Introduction of negative counts by batch correction has negligible impact on the Tensor-cell2cell pipeline.*** *Scatterplots and regression fits of the fraction of negative values (y-axis) introduced by Scanorama to the batch-corrected counts matrix across batch severity (x-axis) assessed by* ***(a)*** *k-means clusterability and* ***(b)*** *mixability.* ***(c)*** *Coefficient estimates of a linear regression estimating the fraction of negative values (dependent variable) from batch severity while controlling for the batch severity metric (independent variables). Since Louvain clusterability is constantly 0 for all fractions of negative values, it is set as the reference. Coefficient estimates are on the y-axis, coefficient labels are on the x-axis, and the horizontal solid black line represents 0 (no effect). Significant estimates (Wald test, Benjamini-Hochberg FDR correction, q-value ≤ 0.1) are marked by a circle and insignificant estimates are marked by a cross.* **(d)** *Scatterplots and regression fit the Scanorama similarity (y-axis) across the fraction of negative values (x-axis). The regression coefficient estimate (as well as its Wald test p-value) are annotated in the bottom right corner. For all scatter plots and regression fits, Spearman correlations are annotated and blue lines show the regression fit with gray shaded regions showing the 95% confidence interval for this fit.*

Next, to determine whether this changes Tensor-cell2cell's output, we assess how Scanorama similarity is affected by the fraction of negative values introduced to the counts matrix. While there is a moderate negative correlation between these two values, it appears that the fraction of negative counts does not substantially affect the Scanorama similarity as indicated by the small regression coefficient estimate and insignificant p-value (**Fig. S7d**). This tells us that using batch correction methods that introduce negative values and simply replacing those with 0 prior to running communication scoring can be appropriate for recovering biological signals from Tensor-cell2cell. We note that one caveat to this analysis is that the maximum fraction of negative counts introduced by Scanorama is limited to 0.125.

1.2.4 Tensor-cell2cell Performance across Batch Severity with Correction

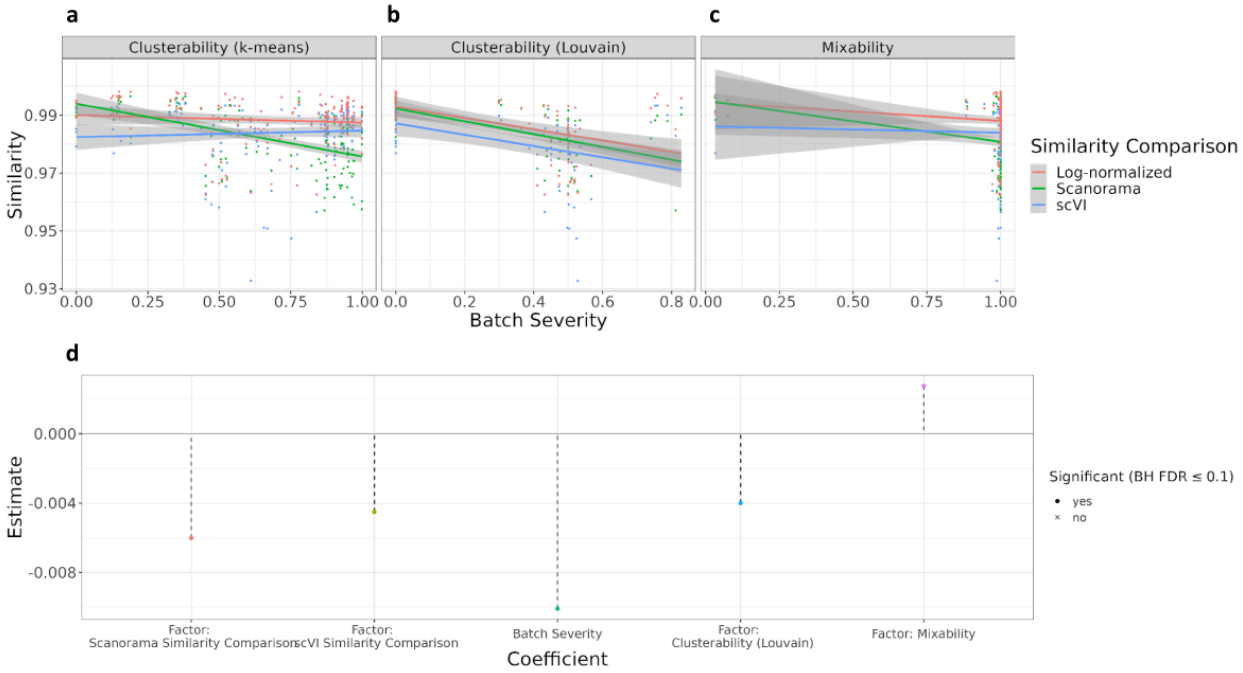

Figure S8. ***Batch effect correction has negligible impact on the Tensor-cell2cell pipeline.*** *Scatterplots and regression fits of the fraction of decomposition similarity (y-axis) between log-normalized (red), Scanorama (green), and scVI (blue) and the gold-standard across batch severity (x-axis) assessed by* ***(a)*** *k-means clusterability,* ***(b)*** *Louvain clusterability, and* ***(c)*** *mixability. Solid lines show the regression fit with gray shaded regions showing the 95% confidence interval for this fit.* ***(d)*** *Coefficient estimates of a linear regression estimating the decomposition similarity (dependent variable) according to the similarity comparison type (independent variable) (i.e., log, Scanorama, or scVI) while controlling for batch severity and the batch severity metrics (independent variables). The log similarity comparison type is set as the reference to identify how batch correction compares. Coefficient estimates are on the y-axis, coefficient labels are on the x-axis, and the horizontal solid black line represents 0 (no effect). Significant estimates (Wald test, Benjamini-Hochberg FDR correction, q-value ≤ 0.1) are marked by a circle and insignificant estimates are marked by a cross. The regression intercept is 0.995 and significant (q << 1e-10) (data not shown).*

To see whether batch correction improves decomposition, we can compare how each similarity score changes across batch severity. If batch correction improves decomposition, we would expect batch-corrected similarity (Scanorama and scVI) to a) score higher than log-normalized similarity across batch similarity metrics and b) decrease at a lower rate with increasing batch severity than log-normalized similarity. Across batch severity metrics, we see that this tends not to be the case, though all similarity types maintain a high similarity score across batch severity levels (**Fig. S8a-c**). We confirm this using a regression with the similarity score as the dependent variable and the similarity score type as the independent variable, setting the reference to be the log-normalized similarity score type. From Question 3A, we know that we can include both the Scanorama similarity and scVI similarity score types in the same model without having confounding effects due to Scanorama introducing negative counts. In this regression, we also control for batch severity and the batch severity metric by including them as

covariates. We can see that, while both scVI and Scanorama similarity are significantly smaller than log-normalized similarity after controlling for batch severity, the effect sizes (coefficient estimates) are very small such that similarity is always high regardless of the pre-processing choice (**Fig. S8d**). Overall, these results tell us that while batch effect correction may not be necessary to recover biological signals using Tensor-cell2cell, if the user feels it is important, they can be comfortable in implementing the batch correction method of choice.

2. Strategies to Address Missing Data and Tensor Indices

2.1 Motivation

The purpose of these analyses is to determine how the tensor decomposition used by Tensor-cell2cell handles missing values. Cell-cell communication (CCC) inference tools output the following for each sample: communication scores associated with interactions (i.e., sender and receiver cell type pairs and ligand-receptor (LR) pairs) [12].

Tensor-cell2cell constructs the tensor by concatenating the output of these CCC tools across the context dimension. Consequently, cell types or LR pairs that are not present across all samples will result in missing values (at the "sample - sender cell type - receiver cell type - LR pair" tensor coordinate or index) in the samples they were absent from. Sample-specific "missing" data may be the result of any of the following:

- Technical limitations in measuring a gene or cell.
- Computational pipelines (data processing, negative expression counts or communication scores, thresholding parameters of CCC tools, etc.) that result in the exclusion of certain measurements.
- The cell type or LR pair is absent from that sample due to biological reasons (a "true biological zero").

Tensor-cell2cell's decomposition will handle these missing values differently depending on how they are filled in during tensor construction. Within the tool, this is handled by the “how”, “cell_fill”, “lr_fill”, and “outer_fraction” parameters discussed in the protocols and tutorials. Also note that using LIANA's “return_all” parameter can mitigate the number of missing values due to thresholding parameters in point #2 above.

If the missing values are filled with NaN/None (the default option in Tensor-cell2cell), Tensor-cell2cell's non-negative canonical polyadic decomposition will “mask” these values during decomposition. Technically, this means that during iteration of the Alternating Least Squares algorithm, masked indices are randomly initialized then updated in each iteration. The masked indices in the full tensor are updated with those imputed from the previous iteration, leading to a new optimization problem and new output set of masked values in each iteration. Conceptually, this is essentially an imputation of missing values. If missing values are filled with a floating point value, they will not be masked and be considered as the actual value they were filled in with (rather than imputed) during the decomposition. For example, filling missing indices with 0 will cause these indices to be treated as "true biological zeroes".

Thus, our goal is to determine the effect that the fraction of missing indices and the value with which they are filled, has on decomposition results. To do so, we simulate CCC across multiple samples and construct a gold-standard tensor with no missing indices. Next, we generate missing interactions in the dataset, and fill these values in the tensor with either NaN or 0. Finally, we compared the similarity of decomposition outputs between the tensor with missing indices and the gold-standard using the CorrIndex metric[8].

Our expectations are as follows:

- If Tensor-cell2cell is appropriately robust to missing indices due to technical limitations or computational pipelines that exclude measurements (Points #1 and 2 above), similarity between the gold-standard tensor decomposition output and that of the tensor with missing indices should be high even at a high fraction of missing indices (i.e., when filled with NaN and masked, Tensor-cell2cell should be able to accurately impute the data).
- If Tensor-cell2cell is appropriately sensitive to missing indices because those are truly absent in the sample, similarity between the gold-standard tensor decomposition output and that of the tensor with missing indices should be low as the fraction of missing indices increases (i.e., when filled with 0 and not masked, Tensor-cell2cell should be able to accurately distinguish between the gold-standard tensor and that with true biological zeroes).

2.2 Results

To determine the effect of missing values on Tensor-cell2cell's output, we created a gold-standard context-dependent CCC simulation with no missing indices, iteratively added increased the proportions of missing values, and compared the similarity (CorrIndex ranging from 0 to 1) between resultant decompositions.

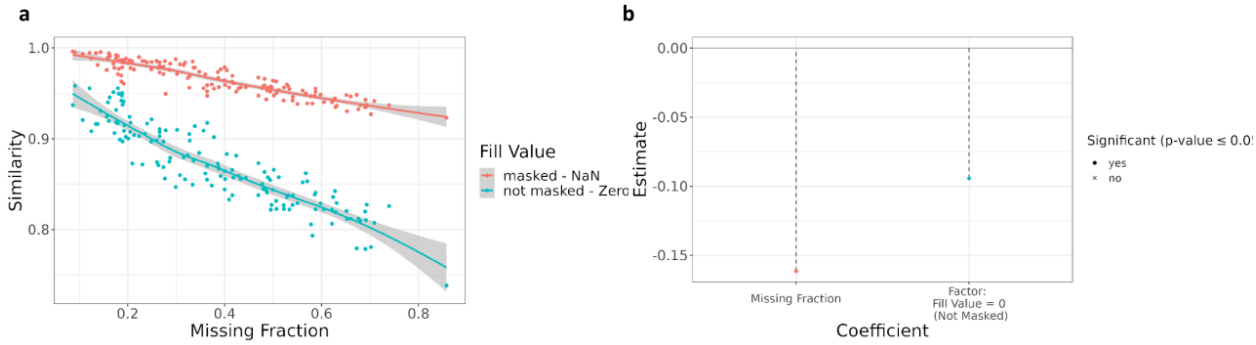

Figure S9. ***Effect of missing indices on Tensor-cell2cell's results.*** *Tensor-cell2cell's output decreases as the fraction of missing elements in the tensor increases.* ***(a)*** *Scatterplots and regression fits of the decomposition similarity (y-axis) between the gold-standard tensor and that with missing indices filled with NaN and subsequently masked (red) or filled with 0 and not masked (green) across the fraction of missing elements in the tensor (x-axis). Solid lines show the regression fit with gray shaded regions showing the 95% confidence interval for this fit.* ***(b)****Coefficient estimates of a linear regression estimating the decomposition similarity (dependent variable) according to the fraction of missing elements in the tensor and the value those missing elements were filled with (independent variables). The NaN fill type*

(masking) is set as the reference since it is expected to affect similarity less. Coefficient estimates are on the y-axis, coefficient labels are on the x-axis, and the horizontal solid black line represents 0 (no effect). Significant estimates (Wald test, $p \leq 0.05$) are marked by a circle and insignificant estimates are marked by a cross. The regression intercept is 1.03 and significant ($p << 1e\text{-}10$) (data not shown).

We found that there was a significant decrease in the similarity of Tensor-cell2cell's output with that of the gold-standard as the fraction of missing indices increased when filling both with NaN (masked) or zero (not masked). However, those that were not masked had a substantially larger decrease in similarity than those that were (**Fig. S9**). The similarity for masked outputs at even very high missing indices (>85% of tensor elements missing) remained above 0.9. The lowest similarity for masked values was 0.923, occurring when 85.8% of the tensor elements were missing. In contrast, when filling with zero at the same fraction of missing values, the similarity was 0.739. When considering the two filling methods in combination with the missing fraction, we see that similarity is lower by 0.094 on average when filling with zero (**Fig. S9b**). Altogether, these results indicate that Tensor-cell2cell is robust enough to impute missing values and sensitive enough to handle true biological zeros.

3. Methods

Here we describe our pipeline for both the Missing Indices and Batch Effects benchmarking simulations. All associated code can be found in the following repository: https://github.com/hmbaghdassarian/tc2c_benchmark. For downstream analyses, unless otherwise specified, all linear regressions in all analyses were performed using a generalized linear model (GLM) with an identity link function; multivariate regressions with > 1 independent variable were combined additively and do not include interaction terms. Additionally, all p-values were multiple-test-corrected using the Benjamini-Hochberg (BH) method to control for false discovery rates (FDRs).

3.1 Single-Cell RNA Data Simulation

We simulated single-cell RNA-sequencing expression data using Splatter[11], adapting a previously described computational approach[7]. We generated a single-cell expression matrix containing 2000 genes and 5000 cells evenly distributed across 6 cell types and 5 samples. Each sample represents a context.

For simulations of missing indices, we introduced a small batch effect to the dataset by setting the Splatter params "batch.facLoc" and "batch.facScale" both to 0.125. This baseline batch effect ensures that average gene expression values across samples are not exactly the same, such that cell-cell communication is expected to change in a context-dependent manner and Tensor-cell2cell will identify multiple factors (rank > 1). To ensure that differences in cell type dominate over batch-effects, we also adjusted the Splatter params "de.facLoc" and

“de.facScale” to be 0.3 and 0.6, respectively.

For simulations of batch effects, we introduced context-dependent changes (across samples) independent of batch effects. First, in each iteration we created a context-independent dataset with “de.facLoc” and “de.facScale” set to 0.65 and 0.1, respectively. This context-independent dataset is preprocessed as described below. Next, we created context-dependent changes by adapting the nested batch effects method used by Luecken et al[7]. Briefly, we added a small, random amount of noise to each sample’s counts matrix; Splatter is used to create the noise added to each counts matrix with parameters “lib.loc” and “lib.scale” set to 6 and 0.6, respectively.

Next, for each sample, we applied quality control filters to the cells and genes as implemented previously[7]. Briefly, low-quality cells were identified and filtered using the scuttle package based on standard metrics (mitochondrial fraction, library size, and number of genes detected); genes detected in fewer than 1% of cells are discarded. Next, counts were normalized using scran pooling [13] and a $\log_{+1}$ transformation. For batch-effect benchmarking, batch correction was further implemented; Scanorama [6] was run on the log-normalized counts matrix and scVI [5] was run on the raw counts matrix. Default parameters were used, except for when running the scVI model, the following parameters were changed: “n_layers" was set to 2, n_latent” was set to 30, and “gene_likelihood” was set to “nb”. To convert the scVI output to a counts matrix, we run the “get_normalized_expression” method of the model, setting the library size to 1e6 and proceeded to apply a $\log_{+1}$ transformation.

3.2 Measuring Batch Severity

For batch correction benchmarking, each counts matrix was quantified for its level of batch severity using two previously applied metrics [7,14]: (1) kBET [1], is an inverse measure of “mixability”, or the extent to which batch effects are removed, and (2) normalized mutual information (NMI) between cell type identity and cluster identity - a measure of “clusterability”, or the extent to which biological variation is conserved. For the clusterability metric, we subtracted the NMI from 1 to quantity batch severity. In this manner, both mixability and clusterability ranged between 0 and 1, with increasing values indicating increasing batch severity.

To attain cluster identities, we first ran principal component analysis (PCA) on each counts matrix to 50 principal components (PCs). We then applied two different clustering algorithms. We used Louvain clustering[10] on the shared nearest neighbors (SNN) graph generated from the PCs, since it is commonly used for single-cell datasets. However, Louvain clustering does not allow one to specify the number of clusters obtained, but rather indirectly change the number of clusters through the “resolution” parameter. This is problematic in simulations where the expected number of clusters is known (in our instance, the number of cell types assigned to the simulation) because NMI is highly sensitive to whether the number of cluster labels matches the number of cell type labels. Thus, we used a binary search algorithm to identify a resolution parameter that identified the number of cluster labels to be within ±1 of the assigned cell type numbers. If after 20 iterations of the search, we could not meet this condition, we did not

measure clusterability using Louvain. Given these challenges, we also identified cluster labels using k-means clustering [9] on the PCs, because it allows us to specify the number of clusters.

3.3 Communication Scoring

From the expression counts matrices, a random subset of 200 genes were chosen to simulate a ligand-receptor interaction network as previously described[15]. Briefly, we use StabEco's[16] BiGraph function, with the power law exponent value set to 2 and the average degree value set to 3, to generate a scale-free, directed, bipartite network of the 200 genes. Half the genes were assigned to be ligands and the other half to be receptors. Not all genes were part of the connected network (70/200), and these were excluded from downstream analyses. This interaction network was used as custom ligand-receptor resource input to LIANA's cell-cell communication scoring.

With simulations providing the necessary inputs of an expression counts matrix and a ligand-receptor interaction resource with iteratively more missing values, we then used LIANA to score communication in each sample of each iteration. For Scanorama batch-corrected counts matrices, we replaced negative values with zero prior to communication scoring.

To assess samples in a manner independent of the scoring method, we used LIANA's aggregate ranking approach to generate a consensus score across the magnitude scoring types. Thus, we scored communication using the methods that output a magnitude score only (CellPhoneDB, SingleCellSignalR, and Connectome/NATMI which both output the same magnitude score). After obtaining the consensus score, we inverted them using (1 - score), such that relevant interactions have a higher value.

We assigned the following non-default parameters to the "liana_wrap" function: we set "expr_prop" to 0.05 and "return_all" to True. Decreasing expr_prop from the default value of 0.1 allowed for more interactions to be considered within a sample. This increased the number of interactions that are present but would have otherwise been thresholded out, thus allowing the assessment of missing tensor indices to be influenced more by their explicit exclusion. When the return_all parameter is set to True, the interactions that were present in the sample but did not receive a communication score due to thresholding are filled with the worst of the scored interactions. In the missing indices simulations, we set this to True because any missing values that were not explicitly simulated, i.e. thresholded out, are true biological zeros. In the batch correction, we set this to True because we do not want the tensor decomposition to impute low scores such as those caused by replacing negative counts with 0.

3.4 Tensor Building and Decomposition

4D-Communication tensors were built from the output of LIANA using the "liana_tensor_c2c" function with default parameters. Similarly, tensors were decomposed using the "decompose_tensor" function with default parameters, except that "tf_optimization" was set to "regular" and "init" is set to 'svd' when estimating the tensor rank. For missing indices, the optimal factor rank was estimated on the gold-standard tensor using the automated elbow

analysis described in the main text, and this rank was used to decompose tensors with omitted values. For the batch effects, since the gold-standard tensor was re-calculated in each iteration, estimating the rank every time would be computationally intractable. Instead, we simulated a baseline tensor prior to iteration using the methods and parameters described above and with no batch effects, calculated the rank on this, and assumed this rank is a reasonable estimate for the new gold-standard tensor generated in each iteration.

For missing indices, the tensor built from the dataset with omitted values by default has the missing indices masked. The fraction of indices with missing values in the tensor was calculated by taking the total number of masked values and dividing it by the total number of interactions stored in the tensor. To assess the effect of filling these values with true biological zero in addition to NaN, we ran the decomposition twice, once with these indices masked and a second time with them unmasked (and the communication scores as these indices being 0). For batch effects, a tensor was built and decomposed using each of the four counts matrices that were generated as described above.

In each iteration, we compared decomposition outputs using CorrIndex[8], as previously described [15]. Briefly, the CorrIndex represents a dissimilarity between decomposition outputs and lies between 0 and 1; we convert this to a similarity metric by using (1-CorrIndex). For missing indices, both of the tensors generated from the dataset with omitted values in each iteration were compared to the gold-standard tensor prior to iteration. For batch effects, in each iteration, three similarity comparisons were made between each of the tensor built from the log-normalized counts matrix with batch effects, that from the Scanorama batch-corrected counts matrix, and that from the scVI batch-corrected counts matrix with that of the gold-standard log-normalized counts matrix with no batch effects.

3.5 Iterating Missing Indices and Batch Effects

To generate missing indices in the 4D-Communication Tensor, we iteratively omitted a random subset of genes or cell types from the expression data. Specifically, we iterated through combinations of the following two variables: the fraction of cell types to remove in a given sample (16, 13, 12, and 23), the fraction of genes (within the 130 in the simulated LR interaction network) to remove in a given sample (110, 310, 12), and the fraction of samples to apply these omissions to (15, 25, 23).

To generate batch effects, we iteratively increased the number of batches (2, 4, and 5) as well as both “batch.facLoc” and “batch.facScale” from 0.01 to 0.1. Unlike iterations in the missing indices, we generated a new gold-standard counts matrix in each iteration. This necessity arises from Splatter's requirement that input for dataset generation must include parameters for the batch effects, which can subsequently be removed to generate a batch-free dataset (“batch.rmEffect” = TRUE). Thus, for each iteration we generated four counts matrices: a gold-standard with no batch effects, a log-normalized counts matrix, a Scanorama batch-corrected counts matrix, and a scVI batch-corrected counts matrix. Across all iterations, both the gold-standard and log-normalized counts matrices had the same amount of

context-dependent noise added. These matrices were then batch-corrected as described above.

References

1. Büttner, M., Miao, Z., Wolf, F.A., Teichmann, S.A., and Theis, F.J. (2019). A test metric for assessing single-cell RNA-seq batch correction. Nat. Methods *16*, 43–49. 10.1038/s41592-018-0254-1.

2. Blainey, P., Krzywinski, M., and Altman, N. (2014). Points of significance: replication. Nat. Methods *11*, 879–880. 10.1038/nmeth.3091.

3. Tran, H.T.N., Ang, K.S., Chevrier, M., Zhang, X., Lee, N.Y.S., Goh, M., and Chen, J. (2020). A benchmark of batch-effect correction methods for single-cell RNA sequencing data. Genome Biol. *21*, 12. 10.1186/s13059-019-1850-9.

4. Loza, M., Teraguchi, S., Standley, D.M., and Diez, D. (2022). Unbiased integration of single cell transcriptome replicates. NAR Genom. Bioinform. *4*, lqac022. 10.1093/nargab/lqac022.

5. Lopez, R., Regier, J., Cole, M.B., Jordan, M.I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. Nat. Methods *15*, 1053–1058. 10.1038/s41592-018-0229-2.

6. Hie, B., Bryson, B., and Berger, B. (2019). Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. Nat. Biotechnol. *37*, 685–691. 10.1038/s41587-019-0113-3.

7. Luecken, M.D., Büttner, M., Chaichoompu, K., Danese, A., Interlandi, M., Mueller, M.F., Strobl, D.C., Zappia, L., Dugas, M., Colomé-Tatché, M., et al. (2022). Benchmarking atlas-level data integration in single-cell genomics. Nat. Methods *19*, 41–50. 10.1038/s41592-021-01336-8.

8. Sobhani, E., Comon, P., Jutten, C., and Babaie-Zadeh, M. (2022). CorrIndex: A permutation invariant performance index. Signal Processing *195*, 108457. 10.1016/j.sigpro.2022.108457.

9. Lloyd, S. (1982). Least squares quantization in PCM. IEEE Trans. Inform. Theory *28*, 129–137. 10.1109/TIT.1982.1056489.

10. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. J. Stat. Mech. *2008*, P10008. 10.1088/1742-5468/2008/10/P10008.

11. Zappia, L., Phipson, B., and Oshlack, A. (2017). Splatter: simulation of single-cell RNA sequencing data. Genome Biol. *18*, 174. 10.1186/s13059-017-1305-0.

12. Armingol, E., Officer, A., Harismendy, O., and Lewis, N.E. (2021). Deciphering cell-cell interactions and communication from gene expression. Nat. Rev. Genet. *22*, 71–88. 10.1038/s41576-020-00292-x.

13. Lun, A.T.L., Bach, K., and Marioni, J.C. (2016). Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. Genome Biol. *17*, 75. 10.1186/s13059-016-0947-7.

14. Chen, W., Zhao, Y., Chen, X., Yang, Z., Xu, X., Bi, Y., Chen, V., Li, J., Choi, H., Ernest, B.,

et al. (2021). A multicenter study benchmarking single-cell RNA sequencing technologies using reference samples. Nat. Biotechnol. *39*, 1103–1114. 10.1038/s41587-020-00748-9.

15. Armingol, E., Baghdassarian, H.M., Martino, C., Perez-Lopez, A., Aamodt, C., Knight, R., and Lewis, N.E. (2022). Context-aware deconvolution of cell-cell communication with Tensor-cell2cell. Nat. Commun. *13*, 3665. 10.1038/s41467-022-31369-2.
16. Feng, W., and Bailey, R.M. (2018). Unifying relationships between complexity and stability in mutualistic ecological communities. J. Theor. Biol. *439*, 100–126. 10.1016/j.jtbi.2017.11.026.