BARcode DEmixing through Non-negative Spatial Regression (BarDensr)

Shuonan Chen[3,4,*], Jackson Loper[2,3], Xiaoyin Chen[5], Tony Zador[5], and Liam Paninski[1,2,3]

[1]Mortimer B. Zuckerman Mind Brain Behavior Institute, Columbia University
[2]Department of Statistics, Columbia University
[3]Center for Theoretical Neuroscience, Columbia University
[4]Department of Systems Biology, Columbia University
[5]Cold Spring Harbor Laboratory
[*]Corresponding author.

August 11, 2020

Abstract

Modern spatial transcriptomics methods can target thousands of different types of RNA transcripts in a single slice of tissue. Many biological applications demand a high spatial density of transcripts relative to the imaging resolution, leading to partial mixing of transcript rolonies in many pixels; unfortunately, current analysis methods do not perform robustly in this highly-mixed setting. Here we develop a new analysis approach, *BARcode DEmixing through Non-negative Spatial Regression* (BarDensr): we start with a generative model of the physical process that leads to the observed image data and then apply sparse convex optimization methods to estimate the underlying (demixed) rolony densities. We apply BarDensr to simulated and real data and find that it achieves state of the art signal recovery, particularly in densely-labeled regions or data with low spatial resolution. Finally, BarDensr is fast and parallelizable. We provide open-source code as well as an implementation for the 'NeuroCAAS' cloud platform.

Author Summary

Spatial transcriptomics technologies allow us to simultaneously detect multiple molecular targets in the context of intact tissues. These experiments yield images that answer two questions: which kinds of molecules are present, and where are they located in the tissue? In many experiments (e.g., mapping RNA expression in fine neuronal processes), it is desirable to increase the signal density relative to the imaging resolution. This may lead to mixing of signals from multiple RNA molecules into single imaging pixels; thus we need to *demix* the signals from these images. Here we introduce BarDensr, a new computational method to perform this demixing. The method is based on a forward model of the imaging process, followed by a convex optimization approach to approximately 'invert' mixing induced during imaging. This new approach leads to significantly improved performance in demixing imaging data with dense expression and/or low spatial resolution.

1 Introduction

Understanding the spatial context of gene expression in intact tissue can facilitate our understanding of cell identities and cellular interactions. How do neighboring cells' gene expressions relate to each other? How are different cell types with different activity patterns positioned in relation to each other? Is the subcellular distribution of gene expression informative about cell type or state? Multiplexed spatial transcriptomics methods offer a promising path forward to investigate these questions, allowing us to spatially resolve gene expression patterns. These assays can measure thousands of different genes simultaneously by looking at the same slice of tissue multiple times through multiple rounds of imaging. Using small barcoded sequences ('probes') which bind to target transcripts and amplify (generating easily detectable 'rolonies'), we can get exponentially more information about the nature of the tissue in each imaging round.

However, fully exploiting this new data type can be challenging, for many reasons. Insufficient optical resolution can cause parts of multiple rolonies to appear in the same imaging voxel, resulting in a 'mixed' signal (Chen et al., 2015; Alon et al., 2020). Tissue can deform or drift over multiple rounds of imaging (Qian et al., 2020), and the signal from individual rolonies can vary slightly between imaging rounds (Moffitt et al., 2016). The chemical washes may fail to complete their work in a given round, such that the imaging in the next round contains residual signal from the previous round (leading to a 'ghosting' effect). Some rolonies may entirely fail to bind to any probes in a given round (Lubeck et al., 2014; Chen et al., 2015). Most of these problems are rare, but they combine to yield a complex relationship between the signal of interest and the observed data.

Traditional techniques for extracting meaning from these images rely on good image preprocessing and clever heuristics; there are two main approaches that we are aware of. Both work well in ideal conditions. One school of thought ('blobs-first') begins by trying to identify regions in the tissue where a rolony may be present, and then tries to use the imaging data to guess the barcode identity of each rolony (Shah et al., 2016; Wang et al., 2018; Qian et al., 2020; Gyllborg et al., 2020; Alon et al., 2020). Another school of thought ('barcodes-first') begins by looking at each voxel and trying to determine whether the fluorescence signal emitted in that voxel over all the rounds is consistent with one of the barcodes (Lee et al., 2014; Moffitt et al., 2016, 2018). These two approaches are implemented in e.g. the '*starfish*' (https://github.com/spacetx/starfish) package (under the names of 'spot-based' and 'pixel-based' approaches, respectively).

Both of these general approaches face difficulties whenever different rolonies make contributions to the same voxel. This can happen in regions of high expression density, and/or insufficient optical resolution. In many cases it is desirable to maximize the signal density, to increase the number of transcripts detected per cell and therefore the power of any downstream statistical analyses — while conversely, for practical reasons, we would like to minimize imaging time and file size, encouraging lower imaging resolution. To correctly identify rolony positions and identities in images with overlap, it is then necessary to perform some kind of 'demixing.' Because of this challenge, many current methods simply discard any blobs in regions where strong mixing occurs (Chen et al., 2015; Wang et al., 2018; Gyllborg et al., 2020).

To overcome this challenge, we sought to address the multiplexing problem directly. *BARcode DEmixing through Non-negative Spatial Regression* (BarDensr) is a new approach for detecting and demixing rolonies. This approach directly models the physical process which gives rise to the observations (Figure 1), including background-noise components, color-mixing, the point-spread function of the optics, and several other features. By directly modeling these physical processes, we are able to accurately estimate overall transcript expression levels – even when the transcript density is so high that it is very difficult to isolate and decode the identity of individual rolonies.

We provide a Python package for implementing these methods on either CPU or GPU architectures (https://github.com/jacksonloper/bardensr). The method requires about two minutes of compute time on a `p2.xlarge` Amazon GPU instance to process a seven-round, four-channels 1000×1000-pixel field of view from an experiment targeting 79 different transcripts. We also provide an implementation for the NeuroCAAS web-service (Abe et al., 2020), which can be used in a drag-and-drop fashion, with no installation required. We compared this method with three alter-

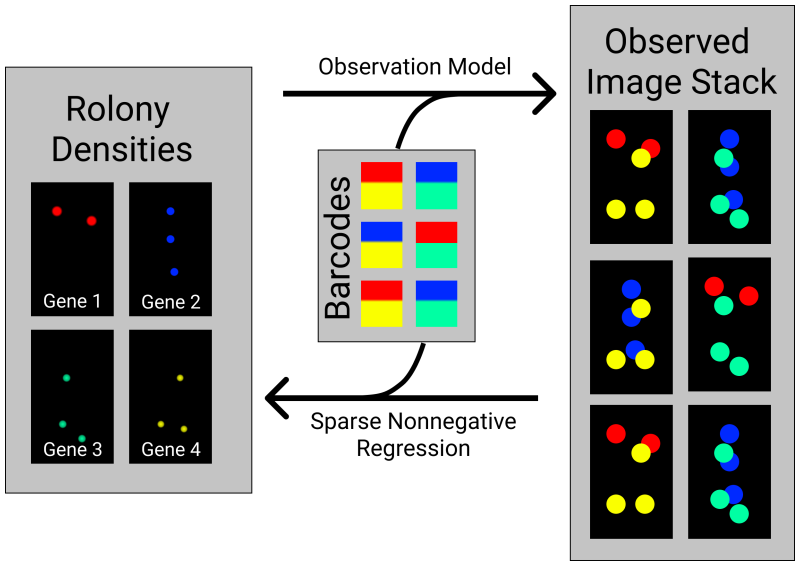

Figure 1: **BarDensr uses non-negative regression to demix and deconvolve the observed image stack, yielding a sparse intensity image for each barcode.** To find rolonies, we write down an observation model. We then use sparse non-negative regression to invert the observation model, yielding demixed and deconvolved intensities.

natives: the spot-based method of *starfish*; another 'blobs-first' approach (Single Round Matching, or SRM, based on methods from (Wang et al., 2018; Qian et al., 2020)); and a 'barcodes-first' approach (Correlation approach, or 'corr,' based on (Lee et al., 2014; Moffitt et al., 2016, 2018)). Both in simulation and real data, BarDensr improves on the state of the art in demixing accuracy.

2 Methods

Data

The experimental images were obtained using an improved version of BARseq (Chen et al., 2019) to detect 79 endogenous mRNAs in the mouse primary visual cortex. The Cold Spring Harbor Laboratory Animal Care and Use Committee approved all animal procedures and experiments. Gene identities were read out using a seven-nucleotide gene identification index (GII), which were designed with a minimal hamming distance of three nucleotides between each pair of GIIs.

Rolonies were prepared as described by (Sun et al., 2020). Imaging was performed on an Olympus IX81 inverted scope with a Crest Xlight2 spinning disk confocal, a Photometrics BSI Prime camera, and an 89 North LDI 7-line laser source. All images were acquired using an Olympus UPLFLN 40x 0.75 NA objective. The microscope was controlled by micro-manager (Edelstein et al., 2014).

See Appendix A for the preprocessing steps for this data, and Appendix E for the process of generating the simulation data.

Notation and Observation Model

Formally speaking, what is the result of a spatial transcriptomics imaging experiment? For each voxel (m) in the tissue, at each imaging round (r), in each color-channel (c), we record a fluorescence intensity. We will use $\mathbf{X}_{m,r,c}$ to denote this fluorescence intensity. Our task is to use $\mathbf{X}$ to

uncover the presence and identity of rolonies in the tissue.[1] Below we describe the parameters used to model the physical process that yields these intensities:

The rolonies, F. The transcripts in the tissue are amplified in place into a 'rolony' structure which is easy for fluorophores to bind to (Shah et al., 2016). Each voxel m may contain a different amount of rolony material, and hence a varying level of fluorescence signal. We refer to the amount of material in voxel m for rolonies associated with barcode j as the **rolony density**. We denote this density by $\mathbf{F}_{m,j}$. The variable $\mathbf{F}$ indicates where rolonies are and how bright we should expect them to be. This density should always be non-negative. Note that $\mathbf{F}$ cannot be observed directly – instead, we observe fluorescence signal in different rounds and channels, and must use these signal observations to estimate the rolony densities.

The codebook, B. In each imaging round r, the rolonies associated with gene j will bind to specific fluorescently labeled detection probes. We use the binary variable $\mathbf{B}$ to indicate which imaging rounds and fluorescent probes each gene is associated with. Specifically, we let $\mathbf{B}_{r,c,j} = 1$ whenever a rolony with barcode j should bind to a fluorescent probe associated with specific color-channel c in imaging round r (and 0 otherwise). Here we assume $\mathbf{B}$ is known. The vector of values of $\mathbf{B}$ for a particular gene j is known as the 'barcode' for that gene, and the collection $\mathbf{B}$ of all the barcodes is known as the 'codebook.'

The probe response functions, $\mathbf{K}, \varphi$. If a probe centered at a particular voxel is illuminated with a particular wavelength, the probe will emit a certain amount of signal which we can record at the corresponding voxel. We may also observe dimmer responses at neighboring voxels, due to the possible spreading of the single point object in the optical system. We use a non-negative matrix $\mathbf{K}$ to denote the *point-spread function*, i.e., the typical fluorescence signal-levels produced at each voxels in the neighborhood of a probe. We use the matrix φ to represent the responsiveness of each type of fluorescent probe to each wavelength; each element of this matrix lies in the range of $[0, 1]$. Here we assume that the number of types of fluorescent probes is the same as the number of color-channels measured (though this could be relaxed). We further assume that the voxel-resolution of the rolony density is the same as the voxel-resolution of the original images.

Phasing, ρ. A washing process is applied after each round of imaging. However, in practice this washing step may not completely remove all of the reagents from every voxel. This can result in a 'ghost' of one round appearing in the next rounds. For each color-channel c, we let $\rho_c \in [0, 1]$ indicate the fraction of activity which appears as a 'ghost' signal in the next round.

Background, a. The images we obtain may also include background fluorescence from the tissue. We assume that the background is constant across rounds. We model this effect using a non-negative per-voxel value a_m for each voxel m.

Per-round per-wavelength gain, α, and baseline, b. The brightness observed from all rolonies at a particular color-channel in a particular round may have an associated gain factor. We model this gain factor with a non-negative per-round (r) per-channel (c) multiplier $\alpha_{r,c}$ and non-negative intercept $b_{r,c}$,.

Putting all these pieces together, we obtain an *observation model*. This model states that the observed brightnesses $\mathbf{X}_{m,r,c}$ should be given by the formulae

$$\mathbf{X}_{m,r,c} \approx a_m + b_{r,c} + \alpha_{r,c} \sum_{j,m',c'}^{J,M,C} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \varphi_{c,c'} \mathbf{Z}_{r,c',j},$$

$$\mathbf{Z}_{r,c,j} = \rho_c \mathbf{Z}_{r-1,c,j} + \mathbf{B}_{r,c,j}.$$

[1] Throughout we assume that $\mathbf{X}$ is preprocessed, including background removal and image registration (see Appendix A for more detail), hence that there are no systematic shifts of the image between imaging rounds.

Here the variable $\mathbf{Z}$ is used to incorporate the round-phasing effects; i.e., $\mathbf{Z}_{r,c,j}$ measures the concentration of probes of type c which we would expect at round r, arising from a rolony with barcode j. We will also find it convenient to define

$$\mathbf{G}_{r,c,j} = \alpha_{r,c} \sum_{c'} \varphi_{c,c'} \mathbf{Z}_{r,c',j}.$$

This represents the total contribution of fluorescence signal expected to arise in round r and channel c from a rolony of type j. A summary of notation can be found in Table 1.

Overall, the model introduced above could certainly be expanded to model the physical imaging process more accurately, but we found that it was sufficient for our purposes: detecting and demixing rolonies.

Inference

Our task is to uncover the positions and barcodes of rolonies in the tissue. According to the model in the previous section, this information can be obtained from the rolony density variable, $\mathbf{F}$. However, $\mathbf{F}$ cannot be directly measured; thus our primary task is to estimate $\mathbf{F}$ from the original image data. To do this we must in a sense invert the observation model specified above: the observation model tells us how rolony densities gives rise to the fluorescence signal, but we would like to use observations of the fluorescence signal to estimate the rolony densities.

Using the observation model to estimate the rolony densities F

We use a sparse non-negative regression framework to estimate the unknown parameters. In this estimation we are guided by three ideas:

- We believe our observation model is *approximately* correct. We formalize this by saying that we believe our squared 'reconstruction loss' can be made small. We define this loss by

$$L_{\text{reconstruction}} = \sum_{m,r,c} \left(\mathbf{X}_{m,r,c} - \left(a_m + b_{r,c} + \alpha_{r,c} \sum_{j,m',c'}^{J,M,C} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \varphi_{c,c'} \mathbf{Z}_{r,c',j} \right) \right)^2 .$$

	Description	Dimensions	Support
M	number of voxels	scalar	$\mathbb{N}$
C	number of types of probes/wavelengths	scalar	$\mathbb{N}$
R	number of rounds	scalar	$\mathbb{N}$
J	number of barcodes	scalar	$\mathbb{N}$
$\mathbf{X}$	observed imaging intensities	$M \times R \times C$	$\mathbb{R}^+$
$\mathbf{F}$	spatial rolony density	$M \times J$	$\mathbb{R}^+$
$\mathbf{B}$	(known) binary codebook matrix	$R \times C \times J$	$\{0, 1\}$
ρ	per-channel phasing factor	C	$[0, 1]$
$\mathbf{Z}$	phased barcodes	$R \times C \times J$	$\mathbb{R}^+$
α	per-round per-channel scale factor	$R \times C$	$\mathbb{R}^+$
$\mathbf{G}$	scaled color-mixed phased barcodes	$R \times C \times J$	$\mathbb{R}^+$
b	per-round per-channel offset	$R \times C$	$\mathbb{R}^+$
a	per-pixel baseline intercept term	M	$\mathbb{R}^+$
$\mathbf{K}$	spatial point-spread function	$M \times M$	$\mathbb{R}^+$
φ	probe wavelength-response matrix	$C \times C$	$[0, 1]$
ω	tolerated reconstruction error	scalar	$\mathbb{R}^+$

Table 1: Notation

- We believe that all of our parameters are non-negative. For example, we do not believe it is possible to have *negative* densities for rolonies at a particular voxel. Likewise, we expect the per-round per-channel scaling factors (α) and probe-response terms (φ) to be non-negative.

- We believe that the rolony densities, $\mathbf{F}$, are sparse: many voxels will not contain any rolony at all. Ideally we would formalize this idea by putting a penalty on the number of voxels with nonzero rolony amplification. However, this penalty is difficult to optimize in practice. Instead, following a long history of work in sparse estimation theory (Hastie et al., 2015), we enforce this sparsity by placing a linear penalty on the total summed density. We define this penalty by

$$L_{\text{sparsity}} = \sum_{m,r,c,c',j} \alpha_{r,c} \mathbf{F}_{m,j} \varphi_{c,c'} \mathbf{Z}_{r,c',j}.$$

  (Note that for a general sparse estimation problem, this penalty would be defined using a summed absolute value term; however, in our case all parameters are already constrained to be non-negative, so this is not necessary.)

Together, these three ideas suggest constrained optimization as a natural way to estimate our parameters. We will seek the non-negative parameters that give the smallest possible value of L_{sparsity}, subject to the constraint that $L_{\text{reconstruction}}$ falls below a noise threshold ω. We provide an automatic way to select this noise threshold (see Appendix I), as well as an interactive process for the user to select this threshold so that the reconstruction loss appears satisfactory.

Assuming that $\mathbf{B}, \mathbf{K}$ are known, this constrained optimization problem can be written as:

$$\begin{aligned} \min_{\mathbf{F},\rho,\alpha,b,a,\varphi \geq 0} \quad & L_{\text{sparsity}}, \\ \text{subject to} \quad & L_{\text{reconstruction}} \leq \omega. \end{aligned} \tag{1}$$

To solve this optimization problem, we use a projected gradient descent approach. The linear structure of the problem makes it possible to pick all learning rates automatically; for example, the resulting algorithm reaches convergence for a single 1000×1000 field of view (with a total of 28 images, with seven rounds and four color-channels) and 81 different barcodes (79 from the original experiment, and two additional unused barcodes as described below) in about two minutes on a `p2.xlarge` Amazon GPU instance. Details can be found in Appendix I.

Before concluding this section, we will address an issue of what is known as 'identifiability.' Let us say we have learned a model via our inference method, i.e. we have learned $\mathbf{F}, \rho, \alpha, b, a, \varphi$. Now let us consider a new model, $\mathbf{F}', \rho', \alpha', b', a', \varphi'$, such that

$$\begin{aligned} \mathbf{F}' &= 4\mathbf{F} & \rho' &= \rho \\ \alpha' &= \alpha/2 & b' &= b \\ a' &= a & \varphi' &= \varphi/2. \end{aligned}$$

Under this new model, the reconstruction loss is the same and the sparsity loss is the same. As far as our inference method is concerned, the two models are identical. It follows that our inference procedure simply cannot hope to learn overall scaling factors of this kind. Thus, any learned parameters should be understood as being known up to overall scale factors. To resolve this ambiguity we normalize α by dividing by its sum (recall that α is non-negative, so this sum will be positive) and multiply $\mathbf{F}$ by the same factor. Similarly, we divide each row of φ by its diagonal value and multiply the corresponding column of α by the same value.

Finding rolonies

Let us now assume we have used the non-negative regression framework to estimate $\mathbf{F}$ (the collection of rolony density images, one for each barcode). These per-barcode density images indicate the positions of rolonies that belong to a particular barcode; see the left side of Figure 1 for a

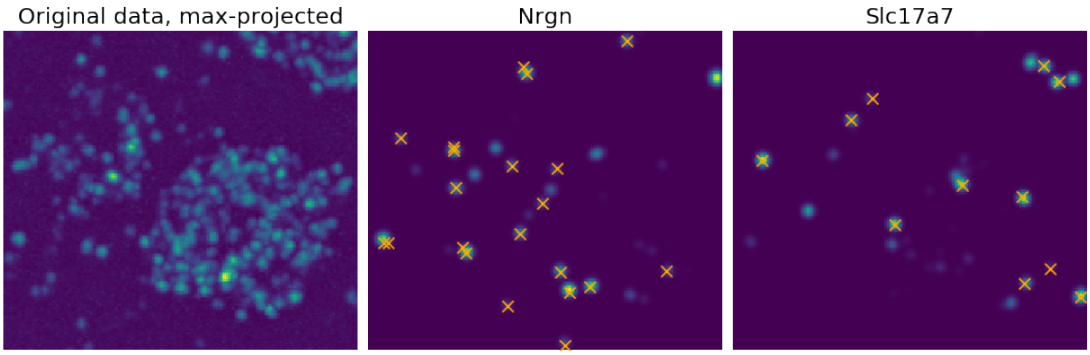

Figure 2: **Rolony densities make it easier to detect rolonies.** The left plot shows the max-projection of the original image across all rounds and channels; detecting blob-like structures in this image can be challenging, especially when two rolonies are in close proximity. By contrast, the rolony densities for particular genes are sparser, so it is easier to identify the positions of individual rolonies in the tissue. The middle and right plots show examples of these rolony densities (in fact, to make it possible to even visualize the results, we here show the rolonies densities *after* applying the point-spread function $\mathbf{K}$; the true rolony densities $\mathbf{F}$ are even sparser). The orange marks represent rolonies detected by a hand-curated approach. Note that the rolony densities appear to show several rolonies which were not detected by the hand-curated approach (in particular, we see several bright spots with no orange marks). In Figure 27 and 28, we show that these additional rolonies do indeed seem to be valid, and were simply missed by the original hand-curated approach.

schematic. We can then apply a blob-finding algorithm to these per-barcode images to find the rolonies for each barcode; in practice we simply find local maxima in the per-barcode images.

Finding rolonies, or 'blobs,' in the per-barcode images is easier than finding blobs in the original images. See Figure 2 as an example. The per-barcode images include fewer blobs and the blobs are smaller, so there are fewer problems with overlapping blobs. More specifically:

- There are fewer blobs in each rolony density than in the original image stack. In the observed images, the intensity measured for each voxel for each wavelength at each round is a sum of contributions from all nearby rolonies which emit signal at that wavelength in that round. By contrast, the intensity measured at a particular voxel in the per-barcode images is only the sum of contributions from rolonies with that one specific barcode.

- The blobs are smaller in the rolony density than in the original image stack. In the observed images, the intensity at a voxel is a contribution from all rolonies which are within the radius of the point-spread function $\mathbf{K}$. Recall that this function smears signal from a single voxel across all nearby voxels. By contrast, the intensity of a per-barcode image at a particular voxel represents the amplification level of rolonies in that one voxel. In this sense, the inference process attempts to invert the point-spread function (i.e., perform deconvolution). On its own, this inversion process would not be numerically stable; however, the sparsity penalty and non-negativity constraint ensures it is numerically well-behaved (Hastie et al., 2015).

The spatial rolony variable $\mathbf{F}$ thus represents a *demixed and deconvolved* version of the raw data. The original data is mixed, insofar as each raw intensity represents contributions from many barcodes. It is also convolved, insofar as each raw intensity represents contributions from many positions in space via the point-spread function. The non-negative sparse regression allows us to simultaneously demix and deconvolve, yielding per-barcode images which are cleaner and easier to understand.

Although it is easier to find blobs in the rolony densities, there is still one obstacle to be overcome: the threshold. Any blob-finding algorithm must specify an intensity above which a blob

is considered real. How can this threshold be chosen? Here we make use of 'unused barcodes.' There could be as many as C^R unique barcodes in a codebook for an experiment with R rounds and C channels of measurement (assuming only one channel emits signal in each round, which is the case in the experiments we studied). However, most of these barcodes are not used in the actual experiment. These unused barcodes give us a way to pick a sensible threshold. Along with the real codebook, we additionally include several unused barcodes; we enumerated all possible barcodes such that each round contained exactly one active channel, then selected uniformly at random from the set of barcodes such that each barcode differed from every other barcode in at least three rounds. We then run BarDensr on this augmented codebook. Blobs in the rolony densities associated with the unused barcodes must correspond to noise, since the true data-generating process did not include any signal from such barcodes. We therefore set the threshold to be the smallest value which guarantees that no blobs were detected in the unused barcodes. (In practice, using just two unused barcodes sufficed to estimate a stable and accurate threshold.)

Accelerating computation

The time required to apply BarDensr scales roughly linearly with the number of voxels in the data. There are several approaches the BarDensr package uses to relieve the computational burdens of working with large datasets:

1. **Exploiting barcode sparsity.** In any given patch of the data, many of the barcodes may not appear at all. If we can use a cheap method to detect genes which are completely missing from a given patch, we can then remove these genes from consideration in that patch, yielding faster operations. We call this 'sparsifying' the barcodes.

2. **Coarse-to-fine.** As we will see below, BarDensr is effective even when the data has low resolution. This suggests a simple way to accelerate computation: downsample the data, run BarDensr on the downsampled data (which will have fewer voxels), and then use the result to initialize the original fine-scale problem. If this initialization is good, fewer iterations of the optimization will be necessary to complete the algorithm.

3. **Parallelization.** BarDensr can use multiple CPU cores or GPUs (when available) to speed up parallel aspects of the optimization (e.g., processing data in spatial patches).

Details on these methods (which can be used in combination with each other) can be found in Appendix H.

Code availability

The BarDensr Python package is available from https://github.com/jacksonloper/bardensr. The NeuroCAAS implementation of BarDensr can be found at http://www.neurocaas.com/analysis/8. This NeuroCAAS implementation requires no software or hardware installation by the user. The BarDensr NeuroCAAS app has a simple input-output model. As input, the user must upload a stack of images, a codebook, and a configuration file specifying parameters such as the radius of the smallest rolonies of interest (see the NeuroCAAS link above for further details regarding the data format.) We assume that the images have been registered and background-subtracted before input into NeuroCAAS. There are two outputs from BarDensr NeuroCAAS implementation. The first output takes the form of a comma-separated-value file listing all entries in the rolony density $\mathbf{F}$ which have signal greater than zero. The second output is a structured HDF5 file, which stores the results of singular value composition (SVD) on the cleaned images for each spot detected; this helps the user assess the quality of the spots detected by the algorithm (see the next section as well as Figures 9 - 10 for detail). See the NeuroCAAS link provided above for full details. Also see Appendix B for further details on the AWS hardware selected here.

3 Results

The rolony densities estimated by BarDensr provide sparse, single images to detect spots for individual barcodes.

As emphasized in Section 2, the sparse non-negative regression approach aims to yield per-gene rolony density images which are easy to work with. The cartoon in Figure 1 may help illustrate this idea. Our belief is that the true per-gene rolony densities will be sparse images, so the learned rolony densities should also be sparse images.

To test this belief, we applied BarDensr to the experimental data described in Section 2. Figure 2 compares the raw data with the learned rolony densities for *Nrgn* and *Slc17a7* in a small region of the tissue. As hoped, the spatial rolony densities are indeed quite sparse compared to the raw data. This ensures that blob-detection is relatively easy. This figure also shows that many of the bright spots in the rolony density images appear next to rolony locations found by a hand-curated method (see Appendix C for details). For visualization purposes, this figure shows the blurred version of the spatial rolony densities (i.e. **KF**); these make it easier to see the bright spots.

To get a sense for what all the different genes look like, we also examined the rolony densities for all the barcodes (81 in total in this dataset, including two unused barcodes); see Figures 11 and 12. These sparse images enable us to identify the rolony location easily for each barcode.

BarDensr provides improved demixing and detection accuracy compared to existing approaches.

To benchmark BarDensr against other methods, we generated simulated data with rolony density, gene expression levels, and noise levels matched to the experimental data, as shown in Figure 3, and then examined how well we could recover the 'true' rolonies from the simulated data. Qualitative results for several different genes are shown in Figures 20 - 23. Quantitatively, we present a Receiver Operating Characteristic curve (ROC curve) in Figure 4, which summarizes the percentage of true detected rolonies (also known as '1-FNR', the complement of the False Negative Rate (FNR)). Depending on the False Positive Rate (FPR) we are willing to tolerate, different detection rates can be achieved; the ROC curve summarizes this relationship.

We compare BarDensr to several other approaches. *Starfish* is one package developed for analyzing spatial transcriptomics data. This method has many hyperparameters. To give this method its best chance, we first tried to find the best parameters manually, and additionally used the *BayesianOptimization* package (Nogueira, 2014) to find the hyperparameters which allowed it to perform as well as possible on the simulated data. Figure 4 shows that this performance falls short of the detection rates achieved by BarDensr. We also investigated SRM (see Appendix C) and a correlation-based method ('corr', see Appendix D) for comparison. These two methods represent 'blobs-first' and 'barcodes-first' approaches. BarDensr has better recovery prediction than either of these.

Our simulated data here do not capture the full biological content of the real observed data. For example, in real data, the tissue often has some regions with dense rolony concentrations (e.g. nuclei) and other regions which are more sparse. In order to quantify performance in more realistic biological contexts, we performed a 'hybrid' simulation, a la (Pachitariu et al., 2016). We started with the original experimental data and injected varying numbers of spots at random locations in the image with varying peak intensities (cf. Appendix E). To test if the model is able to recover these injected spots with the original image background, we computed the FNR (FPR could not be computed here since we do not know the ground truth in the original experimental data). We ran two variants on this simulation: one ordinary simulation and one simulation with 'dropout,' in which some rolonies emit a strong bright signal in most of the rounds but simply vanishes in one or more rounds (see Appendix E). The results of the dropout and non-dropout experiments are shown in Figure 5. As expected, the performance decreases when the intensity of the injected spots is smaller. However, as long as the intensity of injected spots was at least half the maximum intensity of the original image, BarDensr was able to find all the spots, even in the

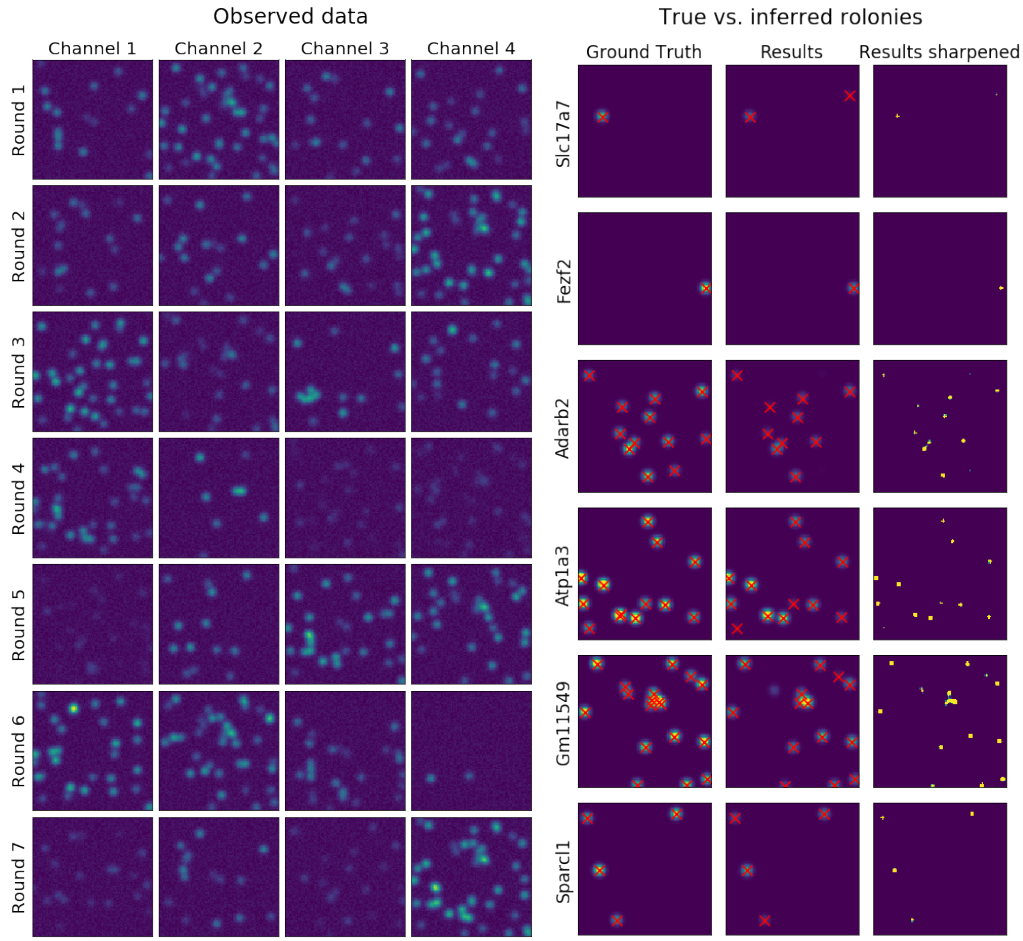

Figure 3: **BarDensr accurately recovers the ground truth in simulated data**. The left plot shows the simulated data in all rounds and channels. Our simulator uses the same barcodes as the true data, the same distribution of gene prevalence found by a hand-curated approach, a similar point-spread function and per-round-and-channel scaling α, and a similar density of spots and noise level. In the right plot, we applied BarDensr to this simulated data, and found that we were able to largely recover the true rolonies in this simulation. The first column of plots shows the true positions of rolonies which were used to generate the simulated data. Each plot shows rolonies for a particular gene. The final column of plots shows the rolony densities learned by BarDensr. The middle column of plots shows a blurred version of the rolony densities (which are a bit easier to see) and the spots discovered from these rolony densities. The algorithm accurately recovers most of the simulated ground truth rolonies, with a few mistakes. In some cases, multiple rolonies of exactly the same barcode lie right next to each other, but the algorithm identified a single large rolony instead of several small rolonies. There are also rare false positives (where we detect a spot that did not exist in the ground truth) and false negatives (where we failed to detect a spot that did exist in the ground truth).

simulation with dropout; by contrast, the SRM approach was unable to find all the injected spots in the hybrid experiment, especially in the dropout variant.

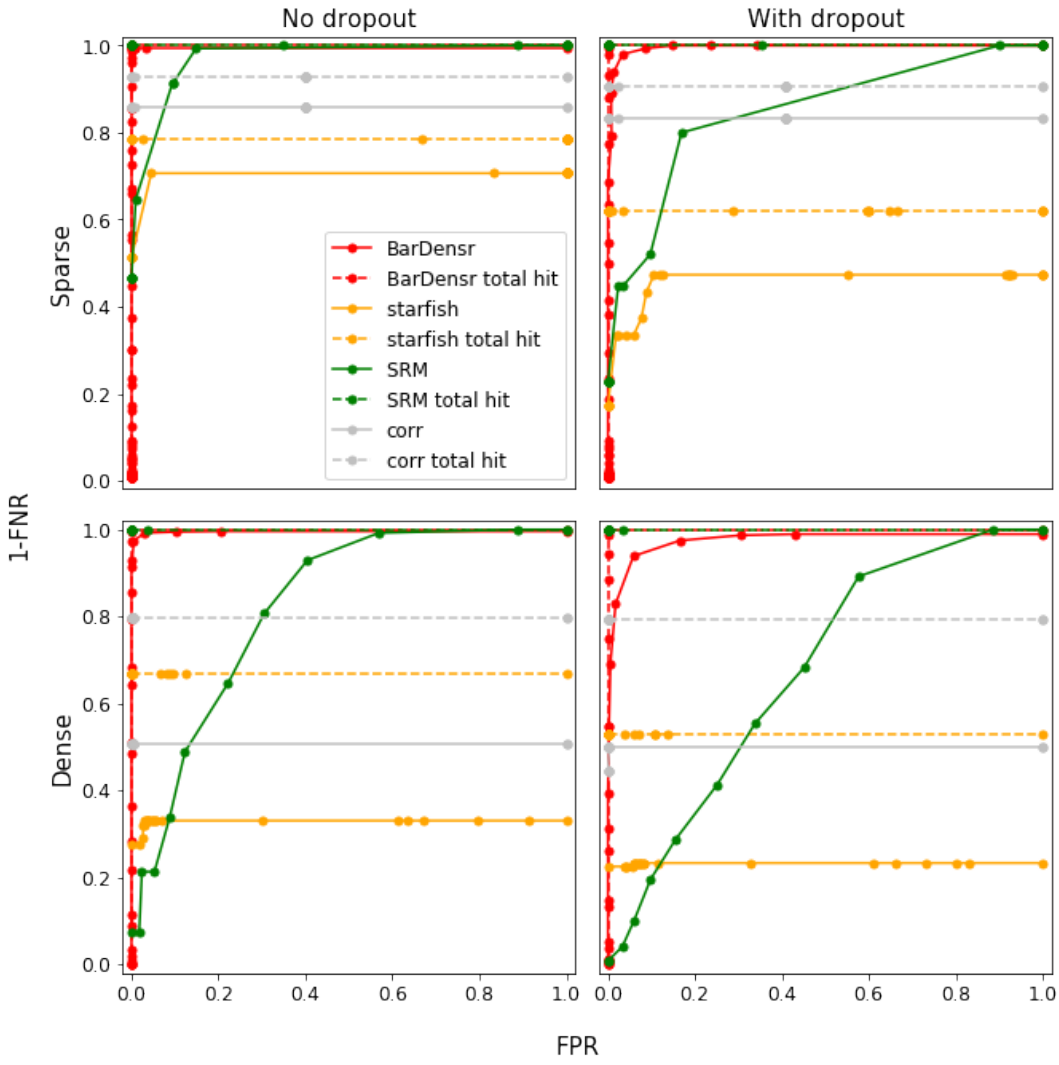

Figure 4: **BarDensr discovers more correct rolonies in simulated data.** What percentage of rolonies are correctly detected? We use the Receiver Operating Characteristic curve (ROC curve) to look at this percentage (the complement of False Negative Rates, or 1-FNR) as a function of the tolerated False Positive Rate (FPR), for BarDensr (red), *starfish* (orange), Single Round Matching (SRM, green), as well as the correlation-based method ('corr', gray); cf. Appendix C and D for details on these other methods. Figure 20 and 21 (for the simulation with sparser spots, top plots of this figure), as well as 22 and 23 (for the simulation with denser spots, bottom plots of this figure) illustrate these simulation data. In drawing these curves, we consider two qualitatively different kinds of errors: errors because a rolony isn't detected at all, and errors because a rolony is detected but it is assigned the wrong barcode. The dotted lines reflect ROC for the former, the solid lines reflect ROC for the latter. The left plots show these curves for simulated data. The right plots show these curves for simulated data with 'dropout' – a form of noise present in some spatial transcriptomic methods (cf. Appendix E for details). For all four kinds of simulations, we found BarDensr is able to find significantly more spots.

Errors are mostly mis-identification on the barcodes, not missed detections.

We used simulated data to investigate the failures represented by the FPR and FNR described above: are they caused by failure in assigning the rolonies to the correct barcodes ('barcode mis-

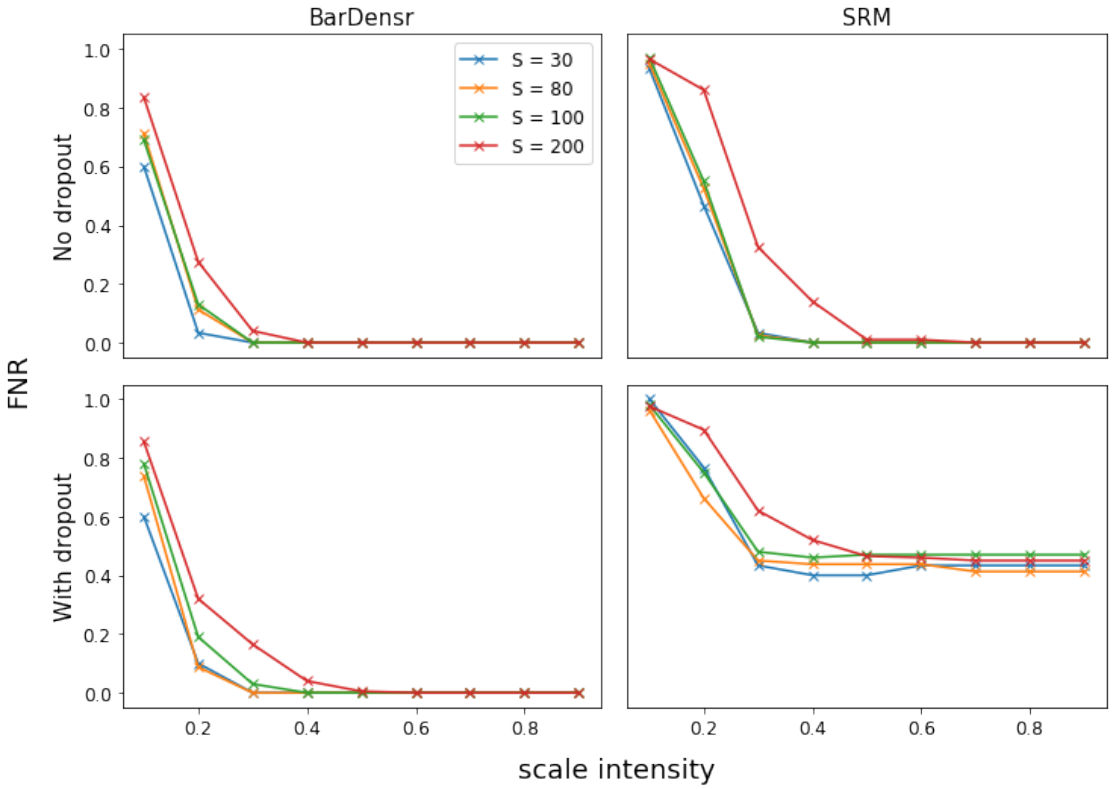

Figure 5: **Hybrid simulation.** Showing the False Negative Rates (FNR, y-axis) as the function of scale intensity (x-axis) and spot number (S, colored lines), without (left) and with (right) dropout, using BarDensr (top) and SRM (bottom). Scale intensity indicates the intensity of the injected spots in the simulation, relative to the maximum intensity in each frame in the original data. See Appendix E for detail.

identification'), or failure in detecting rolonies? To find out, we computed how the failure rates would change if mis-identified barcodes were not considered 'errors.' We denote this the 'total hit rate' analysis (cf. Figure 4, dotted lines); both BarDensr and SRM have very high total hit rates for the simulated data examined here, indicating that both of these methods detect spots well, but sometimes mis-classify the spot identity. See Appendix F for further details.

BarDensr remains effective on data with low spatial resolution.

High-resolution imaging can be expensive and time-consuming. BarDensr can also work on low-resolution images. To show this, we spatially downsampled the experimental images for each frame (each round and each channel). We then fit BarDensr to these lower-resolution images. An example is shown in Figure 6 (additional examples with 5× and 10× lower resolutions can be seen in Figure 24). These figures show that BarDensr correctly detects the overall expression levels of each gene in low-resolution images – even when the downsampling is so extreme that picking out individual rolonies is not feasible.

To test if BarDensr can recover the correct gene expression level when applied on the low-resolution data, we also quantified the cell-level gene activity on a larger region where 43 cells are detected using a seeded watershed algorithm (see Appendix G for detail). The bottom plots of Figure 6 suggest that with 5× downsampled data, the cell-level gene expression, as well as the cell clusters, are preserved with high consistency compared to the results of applying the method to the original fine scale.

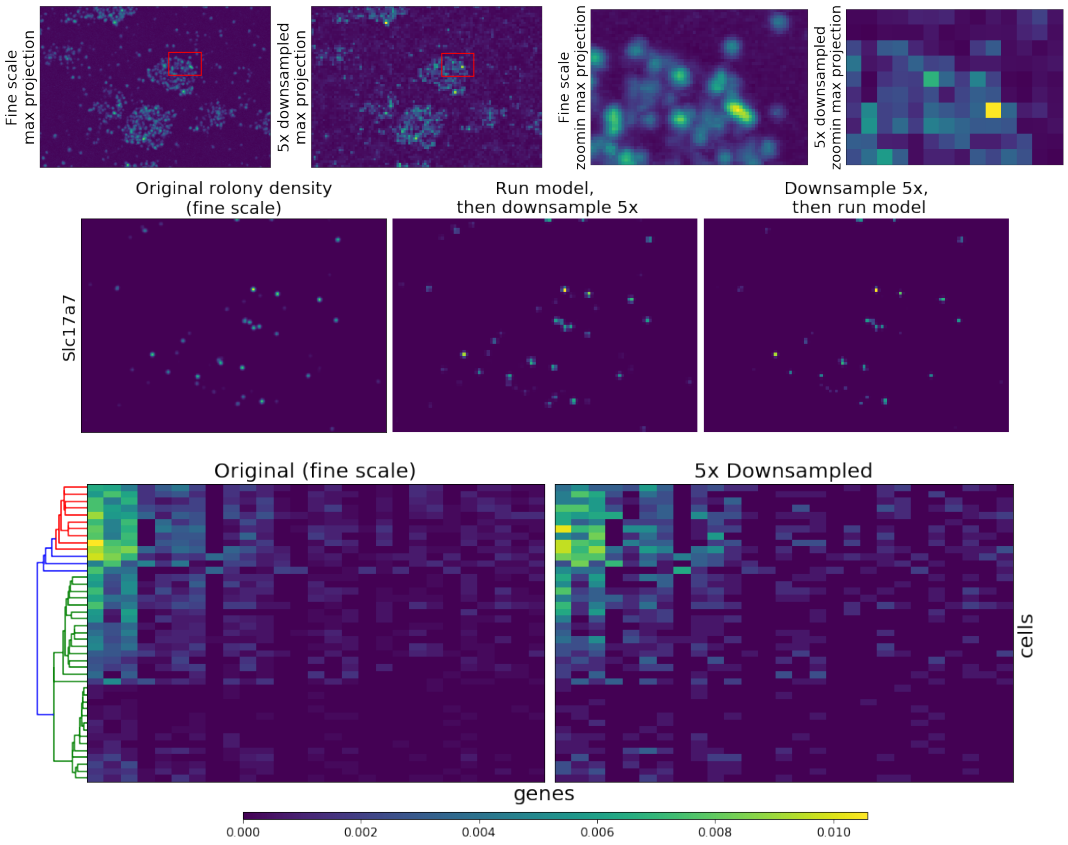

Figure 6: **BarDensr works on low-resolution data. Top row**. The 5× downsampled image is compared to the original 'fine scale' image. All these plots show the max-projection across all rounds and channels, with the right two showing the zoomed region indicated by the red rectangles in the left two. Note that it is difficult to visually isolate single spots from the downsampled image. To test the performance of BarDensr on this low-resolution data, we first run the model on the original data (i.e., top left), obtain rolony densities, and then finally downsample the rolony densities ('run-then-downsample'). Next, we run BarDensr on downsampled data (i.e,. the second plot on the top row) and examine the estimated rolony densities ('downsample-then-run'). **Middle row**. The rolony densities for a selected gene (*Slc17a7*) estimated using the original fine scale (left), as well as these two approaches (middle for 'run-then-downsample' and right for 'downsample-then-run'). For a more complete example, see Figure 24. **Bottom row**. The cell-level gene expression quantification, for those genes that have more than four spots in the fine scale in a 1000×1000 region. The color of the heatmap indicates the proportion of gene counts (i.e., the total counts of each gene divided by the total counts of all genes detected in the region). The x-axis represents the 24 genes that were chosen, ordered based on the counts in the fine scale. The y-axis represents the cells, ordered based on the hierarchical clustering result from the fine scale, as shown in the dendrogram on the left. A total of 43 cells are segmented from the original image using a seeded watershed algorithm (cf. Appendix G). The two different results yield nearly identical clusterings, indicating that BarDensr recovers gene activity with accuracy sufficient to cluster cells even given low-resolution images.

BarDensr computations can be scaled up to tens of thousands of barcodes via sparsifying and coarsening accelerations.

In Section 2, we described how the barcode sparsity could help us potentially apply the method to a large dataset with more barcodes. To test if we can use a much larger dataset, we considered a

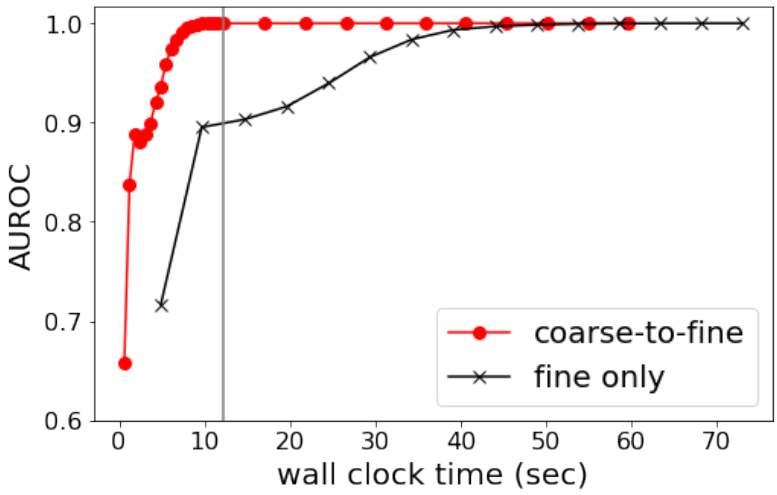

Figure 7: **Speed-up of BarDensr using coarse-to-fine method.** Area Under the ROC curve (AUROC) as a function of wall clock time. The red curve ('coarse-to-fine') is the result where we fit the model to the two-times-downsampled 'coarse' data, followed by running the model at the original fine scale using the parameters learned from the coarse scale as the initial conditions. The black curve ('fine only') indicates the result of running the model on the original fine scale for all the iterations. The total number of updates in the model is 20 and 10 for 'coarse-to-fine' (the gray line indicates the end of the 20 coarse updates), and is 15 for 'fine only.'

simulated example with more unique barcodes (53,000 unique barcodes and 17 sequencing rounds). With so many barcodes, naively running BarDensr is prohibitively expensive (in both compute time and memory) on large datasets. However, we also expect such datasets are extremely sparse in terms of barcodes – any given small region of the image is quite unlikely to include rolonies from all 53,000 barcodes. This is particularly true when each barcode corresponds to a unique cell instead of a unique gene (Chen et al., 2019): a small region of tissue may contain many different transcripts, but it will only contain a small number of different cells. Thus we should be able to take advantage of this sparsity to speed up BarDensr. We simulated a 50×80 small region where 40 rolonies were present in total. We then obtained a coarse, downsampled image, and then ran BarDensr and learned the parameters for this low-resolution data. If the learned parameters from the coarse scale indicated a particular barcode did not appear, then we assumed that this barcode should be absent even if we used the data at original resolution. The result in Figure 25 shows nearly perfect prediction performance. This problem was quite small, so we could also run the method without using any sparsity-based acceleration techniques; we found that the unaccelerated version did not outperform the accelerated version, suggesting that BarDensr can be used for datasets of this kind with larger number of molecular or cellular barcodes (cf. (Kebschull et al., 2016; Han et al., 2018; Chen et al., 2018, 2019)).

Finally, given a small number of barcodes, BarDensr can run without these acceleration techniques – but these accelerations are still worth applying, to help cut down on computation times and reduce memory usage. We found that these techniques reduced runtime by a factor of four (Appendix H). Figure 7 shows the speed-up of the BarDensr using 'coarse-to-fine' accelerations. Further, as shown in Figure 8, BarDensr performs well while taking advantage of the gene-sparsity for each small region after coarsening.

BarDensr recovers interpretable parameters.

BarDensr uses a data-driven approach to estimate all the relevant features of the physical model: the per-channel phasing factor, the per-round per-channel scale factor, the per-round per-channel

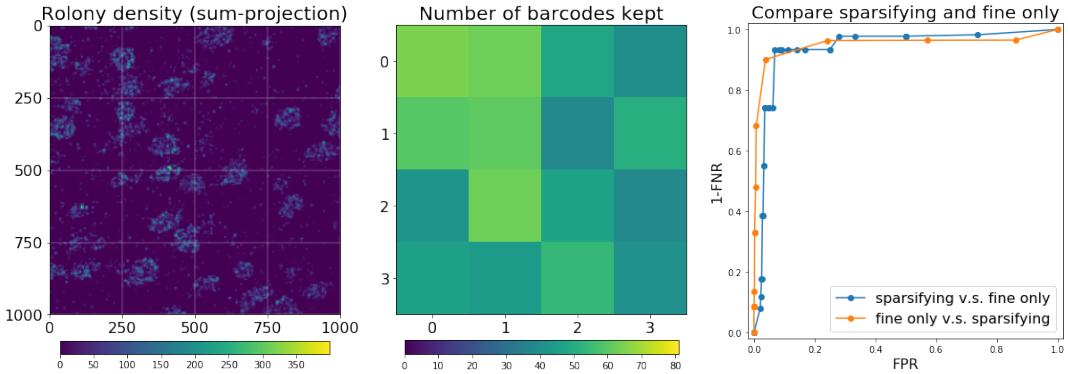

Figure 8: **BarDensr can take advantage of gene-sparsity.** Here we used two different approaches to analyze a 1000×1000 region of the experimental data. The first approach uses BarDensr naively, applying it directly to the image. The second approach is illustrated on the left and middle plots. This approach accelerates the method using a 'coarse-to-fine' method by taking advantage of 'gene-sparsity', i.e., the fact that many barcodes do not appear in any given small region. Specifically, we split this region into 4×4 patches (the borders of these patches are indicated as the white lines on the left plot). After the relatively fast 'coarse' step, the barcodes that have very low maximum rolony densities were removed before the following 'fine' step (cf. Appendix H for more detail). This keeps only a relatively small number of barcodes to consider for each patch (ranging from 38 to 65 out of 81 barcodes, as shown in the middle plot), therefore reducing the computation time and the memory usage for the 'fine' step later. Since we are here analyzing real experimental data, there is no ground truth we can use to compare the efficacy of the two methods. However, we here show that both methods yield nearly the same result, as shown in the ROC curves on the right plot. In particular, we can treat one method as the 'truth' and construct an ROC curve indicating the accuracy of the other method. We can then do the reverse, treating the other method as 'truth.'

offset, the per-pixel background, the per-wavelength response matrix, and the spatial rolony densities (the latter of which have already been described in detail above). In the data analyzed here, we found that the per-channel phasing factor was relatively small, suggesting very little 'ghosting' in this data. The wavelength-response matrix was almost diagonal, although we found some slight color-mixing from channel 2 to channel 1, consistent with visual inspection (see the fifth round in Figure 16 as an example). This indicates that our model is able to correctly recover the color-mixing effects. We also investigated whether all of the features of our model were necessary for the purposes of finding rolonies. For each feature of the model, we tried removing that aspect of the model and seeing whether the method still performed well. For the data analyzed here, we found that the φ and ρ parameters were not essential (though they did seem to improve the performance, at least qualitatively). By contrast, all of the other parameters were essential; removing any of them yielded nonsensical results.

BarDensr is able to capture the important signal based on the assessment on the predicted signal intensities.

Our algorithm is based upon a physical model of how this data is generated. Rolonies appear at different positions in the tissue, they emit fluorescence signal in different conditions, the fluorescence signal is smeared by a point-spread function, and finally we observe this signal, together with certain background signal and noise. As long as this model captures all the important features of the physical process, observed intensities should match the predicted intensities at each voxel in each round and in each channel. To think about this more clearly, let's define these predicted

intensities as the 'reconstruction':

$$\text{reconstruction}_{m,r,c} \triangleq a_m + b_{r,c} + \alpha_{r,c} \sum_{j,m',c'}^{J,M,C} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \varphi_{c,c'} \mathbf{Z}_{r,c',j}.$$

To test our model, we can visually compare the reconstruction to the observed data. If the residual between the two includes significant highly-structured noise, then it is likely that we are missing important aspects of the data. Figures 13 - 18 show the results of these comparisons. They appear fairly promising, but certain structured features do appear in the residual. Most strikingly, we have found that a minority of rolonies 'dropout' for one or more rounds: a rolony may give a strong bright signal in most of the rounds but simply vanish in one round. Our current physical model does not accommodate this, and this limitation appears in the residual as bright and dark spots. However, as mentioned above and shown in the hybrid simulation data in Figure 5, our method is robust to these 'dropout' effects; it is still able to capture the correct rolony positions when it occurs on a small number of rounds.

Diagnostics based on 'cleaned' images are useful to check the accuracy of BarDensr.

The reconstruction is made up of many parts: it has the background component a, the per-round per-channel offset and scale terms (α, b), and rolony contributions arising from $\mathbf{F}, \varphi, \mathbf{Z}$. As shown above, it is straightforward to compare the total reconstruction to the observed data. However, this does not isolate the contributions of individual estimated rolonies.

Therefore we adapted a partial subtraction approach from (Lee et al., 2020). We pick one barcode, j^*, and focus only on the contributions to the reconstruction from this one barcode. In particular, we assume that all other aspects of the model are exactly correct. We assume that a, α, b, φ and $\mathbf{Z}$ are all exactly right. We further assume that $\mathbf{F}_j$ is exactly correct for every $j \neq j^*$. Assuming all these aspects of the model were perfect, we can look at what the data *would have looked like* if it had only included one type of barcode, namely j^*. We call this counterfactual simulation the 'cleaned image':

$$\mathbf{X}^{(j^*)}_{m,r,c} = \mathbf{X}_{m,r,c} - a_m - b_{r,c} - \sum_{j \neq j^*, m'} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \mathbf{G}_{r,c,j}. \tag{2}$$

This is the data with all aspects of the model subtracted away – *except* for the contributions from barcode j^* (see Figure 9 as an example). The cleaned image for the barcode j^* has much in common with the rolony density for j^*. However, $\mathbf{X}^{(j^*)}$ differs from $\mathbf{F}_{j^*}$ in one crucial way. For each voxel m, $\mathbf{F}_{j^*}$ gives exactly one value. However, for each voxel m, $\mathbf{X}^{(j^*)}$ gives $R \times C$ values – one for each round and channel of the experiment. According to our model, however, it should be possible to express all these values in terms of a mathematical 'outer product':

$$\mathbf{X}^{(j^*)}_{m,r,c} \approx \mathbf{F}_{j^*,m} \mathbf{G}_{j^*,r,c}.$$

In this outer product we see that $\mathbf{X}^{(j^*)}_{m,r,c}$ (which varies across voxels, rounds, and channels) is the product of two objects: the rolony density (which varies across voxels) and the transformed barcode $\mathbf{G}$ (which varies across rounds and channels) for j^*. This is actually a very strong assumption; most tensors would not exhibit this kind of structure. We can empirically check for this 'rank-one' structure by computing the singular value decomposition (SVD) of $\mathbf{X}^{(j^*)}_{m,r,c}$. If the SVD yields only one strong singular value, then $\mathbf{X}^{(j^*)}_{m,r,c}$ can be well-approximated by this rank-one outer product, and furthermore the SVD yields the correct values for $\mathbf{F}_{j^*,m}$ and $\mathbf{G}_{j^*,r,c}$. We can compare the values for these quantities (as returned by the SVD analysis) to the estimated values (as returned by BarDensr). We show some examples in Figure 10 comparing the estimated value of $\mathbf{G}_{j^*}$ with SVD results (a similar but more complete set of the spots can be seen in Figure 19). Note that the match isn't quite perfect (the temporal singular vector of the corresponding cleaned images

varies a bit from our estimate). In future work we hope to investigate whether these differences could be accounted for by a more accurate physical model. For now, we content ourselves that the method is accurate enough to provide a useful diagnostic for the detected rolonies.

We can also use these cleaned images to help us compare BarDensr with other methods by eye. Figure 26 investigates cleaned images for gene *Arpp19*, comparing the results of our method to the hand-curated results. In cases where the results of the two approaches disagree, these cleaned images suggest that our results are often reasonable.

4 Conclusion and future work

By directly modeling the physical process that gives rise to spatial transcriptomics imaging data, we found that BarDensr can correctly detect transcriptomic activity – even when rolonies are densely packed in tissue or optical resolution is limited.

BarDensr is computationally scalable, but so far we have only investigated real-world transcriptomic experiments with less than a thousand barcodes. To scale to larger barcode libraries we need to address the possibility that the barcode library may be unknown or corrupted. In experiments with tens of thousands of barcodes, some barcodes present in the data may be unknown to the experimentalist. If these barcodes are ignored, the performance of our method may be negatively impacted. In the future we hope to adapt our method to learn these barcodes directly, using the model outlined in this paper. Together with the computational acceleration approaches used in this paper, this would extend BarDensr to larger-scale data with potentially corrupted barcode libraries.

Acknowledgements

We thank Abbas Rizvi, Li Yuan, Daniel Soudry, Ruoxi Sun, Darcy Peterka, and Ian Kinsella for many helpful discussions. This work was supported by the National Institutes of Health [NIH 5RO1NS073129, 5RO1DA036913, RF1MH114132, U19MH114821, and U01MH109113 to A.M.Z., and 1U19NS107613 to L.P.], the Brain Research Foundation (BRF-SIA-2014-03 to A.M.Z.), IARPA MICrONS [D16PC0008 to A.M.Z. and D16PC0003 to L.P.], Paul Allen Distinguished Investigator Award [to A.M.Z.], Simons Foundation [350789 to X.C.], Chan Zuckerberg Initiative (2017-0530 ZADOR/ALLEN INST(SVCF) SUB awarded to A.M.Z and 2018-183188 to L.P.], and Robert Lourie (to A.M.Z.). This work was additionally supported by the Assistant Secretary of Defense for Health Affairs endorsed by the Department of Defense, 1120 Fort Detrick, Fort Detrick, MD 21702 through the FY18 PRMP Discovery Award Program W81XWH1910083 (to X.C). Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the U.S. Army. In conducting research using animals, the investigator adheres to the laws of the United States and regulations of the Department of Agriculture.

References

Taiga Abe, Ian Kinsella, Shreya Saxena, Liam Paninski, and John P Cunningham. Neuroscience cloud analysis as a service. *bioRxiv*, 2020.

Shahar Alon, Daniel R Goodwin, Anubhav Sinha, Asmamaw T Wassie, Fei Chen, Evan R Daugharthy, Yosuke Bando, Atsushi Kajita, Andrew G Xue, Karl Marrett, et al. Expansion sequencing: Spatially precise in situ transcriptomics in intact biological systems. *bioRxiv*, 2020.

E Kelly Buchanan, Ian Kinsella, Ding Zhou, Rong Zhu, Pengcheng Zhou, Felipe Gerhard, John Ferrante, Ying Ma, Sharon Kim, Mohammed Shaik, et al. Penalized matrix decomposition for denoising, compression, and improved demixing of functional imaging data. *arXiv preprint arXiv:1807.06203*, 2018.

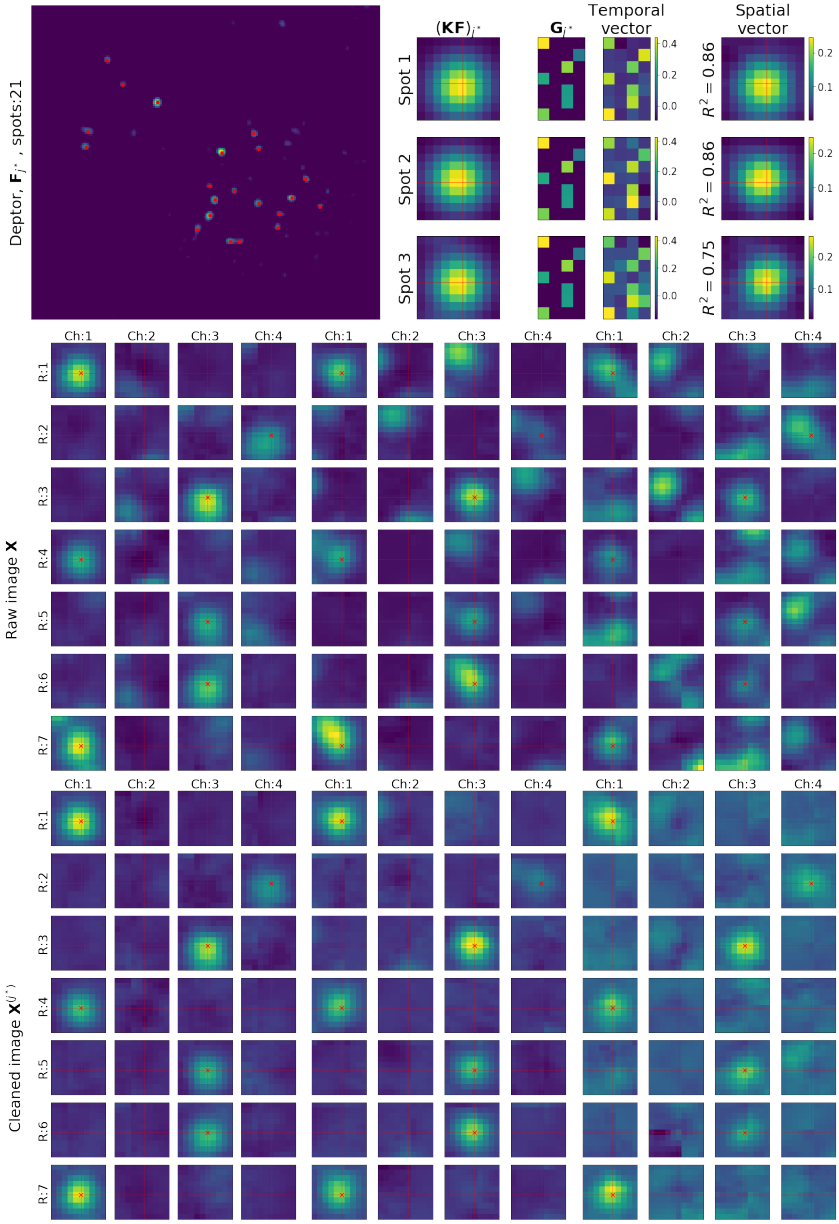

Figure 9: **Using cleaned images and SVD to examine model fit quality and variability.** Top left: spots are identified in $\mathbf{F}_{j^*}$ for each barcode j^* using local-max-peak-finding. For the gene barcode (*Deptor*) shown here, 19 spots were detected (red dots) and three spots with highest accuracy are shown on the right panel. The middle and bottom panels show the zoomed-in $R \times C$ plots of the raw image $\mathbf{X}$ (middle) and 'cleaned' image $\mathbf{X}^{(j^*)}$ (bottom) at these three spot locations for barcode j^*. Note that 'cleaned' images are significantly sparser than the raw images, as desired. Top right: we applied SVD to the cleaned image $\mathbf{X}^{(j^*)}$ at these three spot locations. The first two columns show the zoomed-in image of the original spot $(\mathbf{KF})_{j^*}$ and the learned weighted barcode matrix $(\mathbf{G}_{j^*})$ corresponding to this gene barcode j^*. The top singular vectors are plotted in the last two columns (showing a good match with $\mathbf{G}_{j^*}$ and the cropped $(\mathbf{KF})_{j^*}$). R^2 is the squared correlation coefficient between $\mathbf{X}^{(j^*)}$ and the outer product of these two singular vectors; the high R^2 values seen here indicate that the model accurately summarizes $\mathbf{X}^{(j^*)}$.

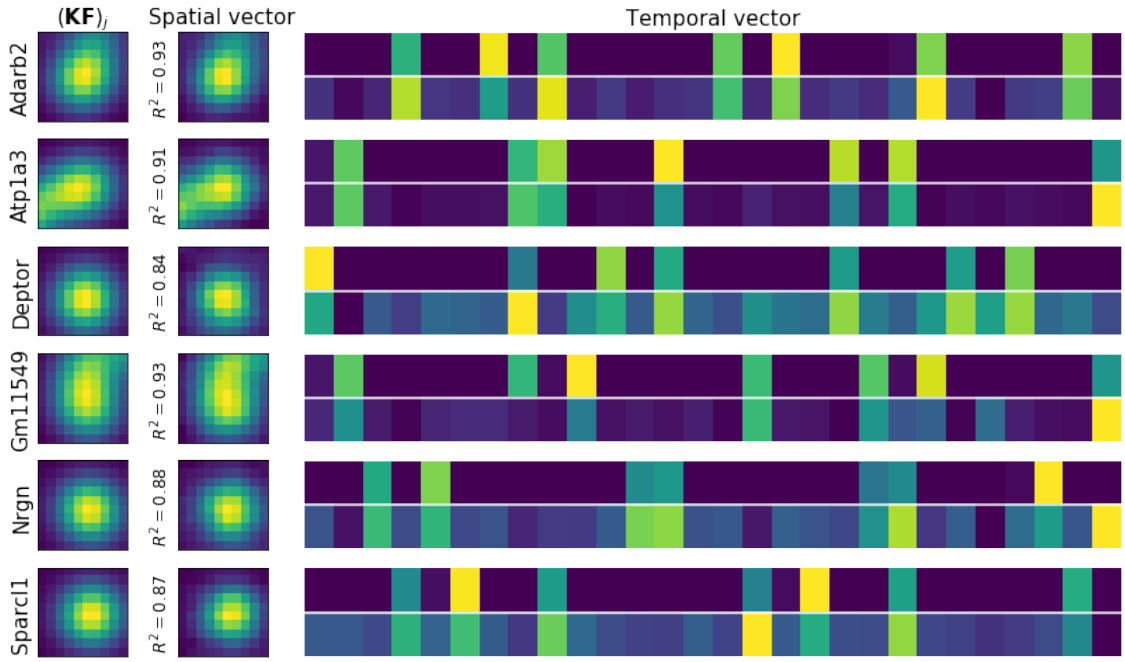

Figure 10: **Results of SVD analysis of cleaned images for the top high-R^2 spots.** This plot summarizes the results of the analysis illustrated in Figure 9. The first column shows $(\mathbf{KF})_{j^*}$ cropped around the brightest spots; the second column shows the top spatial singular vectors for the same crops, and the last column shows the top temporal singular vectors for these spots. For the last column, the top row (above the white line) shows the scaled $\mathbf{G}_{j^*}$ learned from the model, and the bottom row shows the corresponding top temporal singular vectors for these spots. Note that there is some variability visible in these temporal singular vectors. R^2 is computed as in Figure 9. Only six barcodes that are most abundant in the selected region are shown here; Figure 19 provides a more complete illustration.

Kok Hao Chen, Alistair N Boettiger, Jeffrey R Moffitt, Siyuan Wang, and Xiaowei Zhuang. Spatially resolved, highly multiplexed rna profiling in single cells. *Science*, 348(6233):aaa6090, 2015.

Xiaoyin Chen, Yu-Chi Sun, George M Church, Je Hyuk Lee, and Anthony M Zador. Efficient in situ barcode sequencing using padlock probe-based baristaseq. *Nucleic acids research*, 46(4): e22–e22, 2018.

Xiaoyin Chen, Yu-Chi Sun, Huiqing Zhan, Justus M Kebschull, Stephan Fischer, Katherine Matho, Z Josh Huang, Jesse Gillis, and Anthony M Zador. High-throughput mapping of long-range neuronal projection using in situ sequencing. *Cell*, 179(3):772–786, 2019.

Arthur D Edelstein, Mark A Tsuchida, Nenad Amodaj, Henry Pinkard, Ronald D Vale, and Nico Stuurman. Advanced methods of microscope control using μmanager software. *Journal of biological methods*, 1(2), 2014.

Georgios D Evangelidis and Emmanouil Z Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, 2008.

Daniel Gyllborg, Christoffer Mattsson Langseth, Xiaoyan Qian, Sergio Marco Salas, Markus M Hilscher, Ed S Lein, and Mats Nilsson. Hybridization-based in situ sequencing (hybiss): spatial transcriptomic detection in human and mouse brain tissue. *bioRxiv*, 2020.

Yunyun Han, Justus M Kebschull, Robert AA Campbell, Devon Cowan, Fabia Imhof, Anthony M Zador, and Thomas D Mrsic-Flogel. The logic of single-cell projections from visual cortex. *Nature*, 556(7699):51–56, 2018.

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 2015. ISBN 9781498712163. URL https://books.google.com/books?id=LnUIrgEACAAJ.

Justus M Kebschull, Pedro Garcia da Silva, Ashlan P Reid, Ian D Peikon, Dinu F Albeanu, and Anthony M Zador. High-throughput mapping of single-neuron projections by sequencing of barcoded rna. *Neuron*, 91(5):975–987, 2016.

Dongmin Kim, Suvrit Sra, and Inderjit S Dhillon. A non-monotonic method for large-scale non-negative least squares. *Optimization Methods and Software*, 28(5):1012–1039, 2013.

Je Hyuk Lee, Evan R Daugharthy, Jonathan Scheiman, Reza Kalhor, Joyce L Yang, Thomas C Ferrante, Richard Terry, Sauveur SF Jeanty, Chao Li, Ryoji Amamoto, et al. Highly multiplexed subcellular rna sequencing in situ. *Science*, 343(6177):1360–1363, 2014.

JinHyung Lee, Catalin Mitelut, Hooshmand Shokri, Ian Kinsella, Nishchal Dethe, Shenghao Wu, Kevin Li, Eduardo B Reyes, Denis Turcu, Eleanor Batty, et al. Yass: Yet another spike sorter applied to large-scale multi-electrode array recordings in primate retina. *bioRxiv*, 2020.

Eric Lubeck, Ahmet F Coskun, Timur Zhiyentayev, Mubhij Ahmad, and Long Cai. Single-cell in situ rna profiling by sequential hybridization. *Nature methods*, 11(4):360, 2014.

Jeffrey R Moffitt, Junjie Hao, Guiping Wang, Kok Hao Chen, Hazen P Babcock, and Xiaowei Zhuang. High-throughput single-cell gene-expression profiling with multiplexed error-robust fluorescence in situ hybridization. *Proceedings of the National Academy of Sciences*, 113(39): 11046–11051, 2016.

Jeffrey R Moffitt, Dhananjay Bambah-Mukku, Stephen W Eichhorn, Eric Vaughn, Karthik Shekhar, Julio D Perez, Nimrod D Rubinstein, Junjie Hao, Aviv Regev, Catherine Dulac, et al. Molecular, spatial, and functional single-cell profiling of the hypothalamic preoptic region. *Science*, 362(6416):eaau5324, 2018.

Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014. URL https://github.com/fmfn/BayesianOptimization.

Marius Pachitariu, Nicholas A Steinmetz, Shabnam N Kadir, Matteo Carandini, and Kenneth D Harris. Fast and accurate spike sorting of high-channel count probes with kilosort. In *Advances in neural information processing systems*, pages 4448–4456, 2016.

Xiaoyan Qian, Kenneth D Harris, Thomas Hauling, Dimitris Nicoloutsopoulos, Ana B Muñoz-Manchado, Nathan Skene, Jens Hjerling-Leffler, and Mats Nilsson. Probabilistic cell typing enables fine mapping of closely related cell types in situ. *Nature methods*, 17(1):101–106, 2020.

Sheel Shah, Eric Lubeck, Wen Zhou, and Long Cai. In situ transcription profiling of single cells reveals spatial organization of cells in the mouse hippocampus. *Neuron*, 92(2):342–357, 2016.

Stanley R Sternberg. Biomedical image processing. *Computer*, (1):22–34, 1983.

Yu-Chi Sun, Xiaoyin Chen, Stephan Fischer, Shaina Lu, Jesse Gillis, and Anthony M Zador. Integrating barcoded neuroanatomy with spatial transcriptomics reveals the molecular logic of cortical projections. Unpublished, 2020.

Xiao Wang, William E Allen, Matthew A Wright, Emily L Sylwestrak, Nikolay Samusik, Sam Vesuna, Kathryn Evans, Cindy Liu, Charu Ramakrishnan, Jia Liu, et al. Three-dimensional intact-tissue sequencing of single-cell transcriptional states. *Science*, 361(6400):eaat5691, 2018.

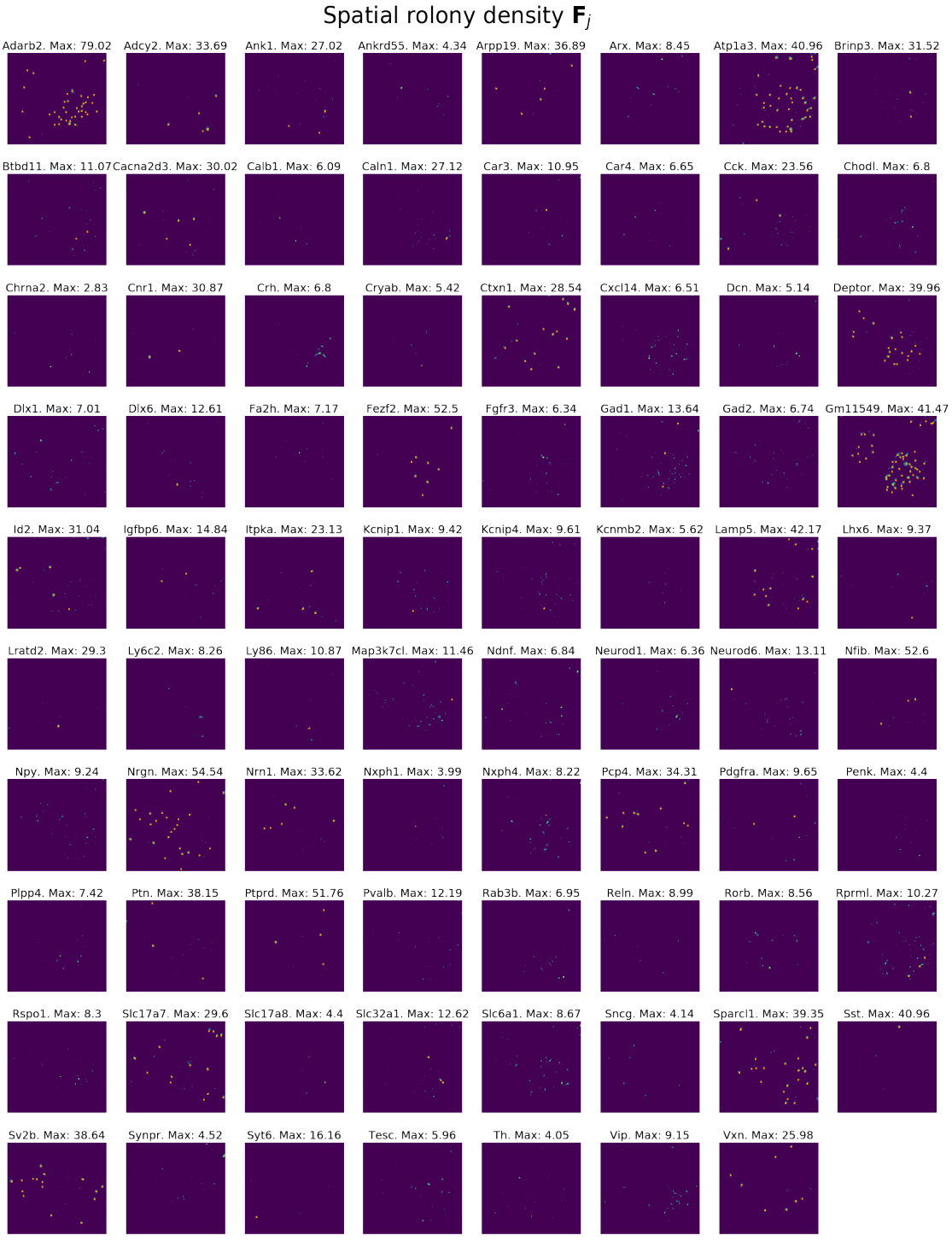

Figure 11: **Spatial rolony density $\mathbf{F}_j$ of all 81 barcodes.** These images are the supplement to Figure 2 in the main text. The rolony densities represent a *demixed* view of the data. Each plot corresponds to a single barcode, and indicates the rolony density at different spatial locations. Above we show these rolony densities for one region in the experimental data. The title for the plots above indicates the gene associated with the barcode as well as the maximum intensity of the plot. The orange dots represent rolonies detected by a hand-curated approach.

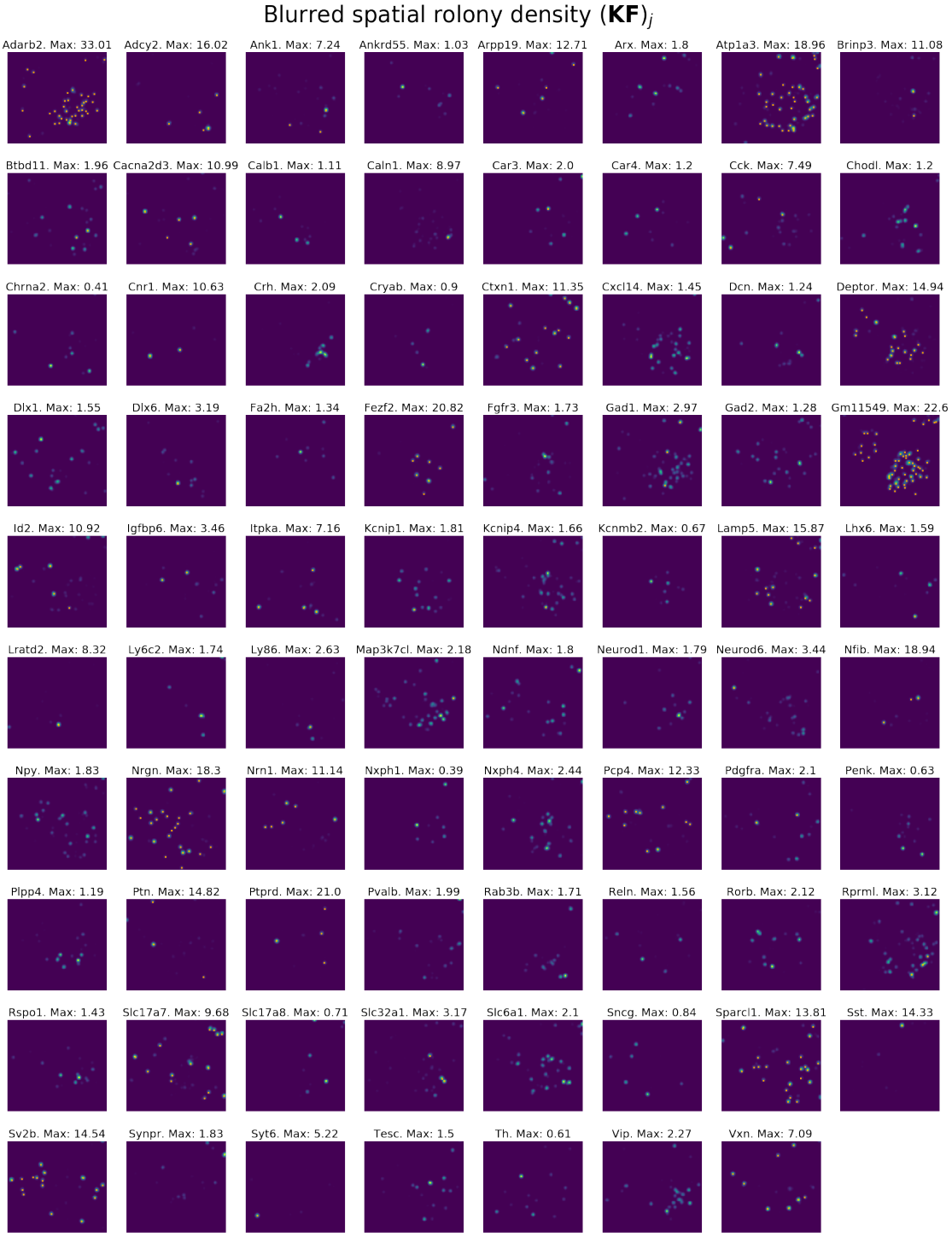

Figure 12: **Spatial rolony density $(\mathbf{KF})_j$ of all 81 barcodes, after applying the point-spread function.** As of Figure 11, these images are the supplement to Figure 2 in the main text, except we display $(\mathbf{KF})_j$ instead of $(\mathbf{F})_j$ for each barcode j. Recall that the point-spread function $\mathbf{K}$ has the effect of smearing signal over a spatially localized area. It represents physical processes which blur the signal of interest. Under the BarDensr model, the signal intensities observed at each voxel m from a given barcode will arise directly from linear combinations of $(\mathbf{KF})_{m,j}$ over different barcodes j.

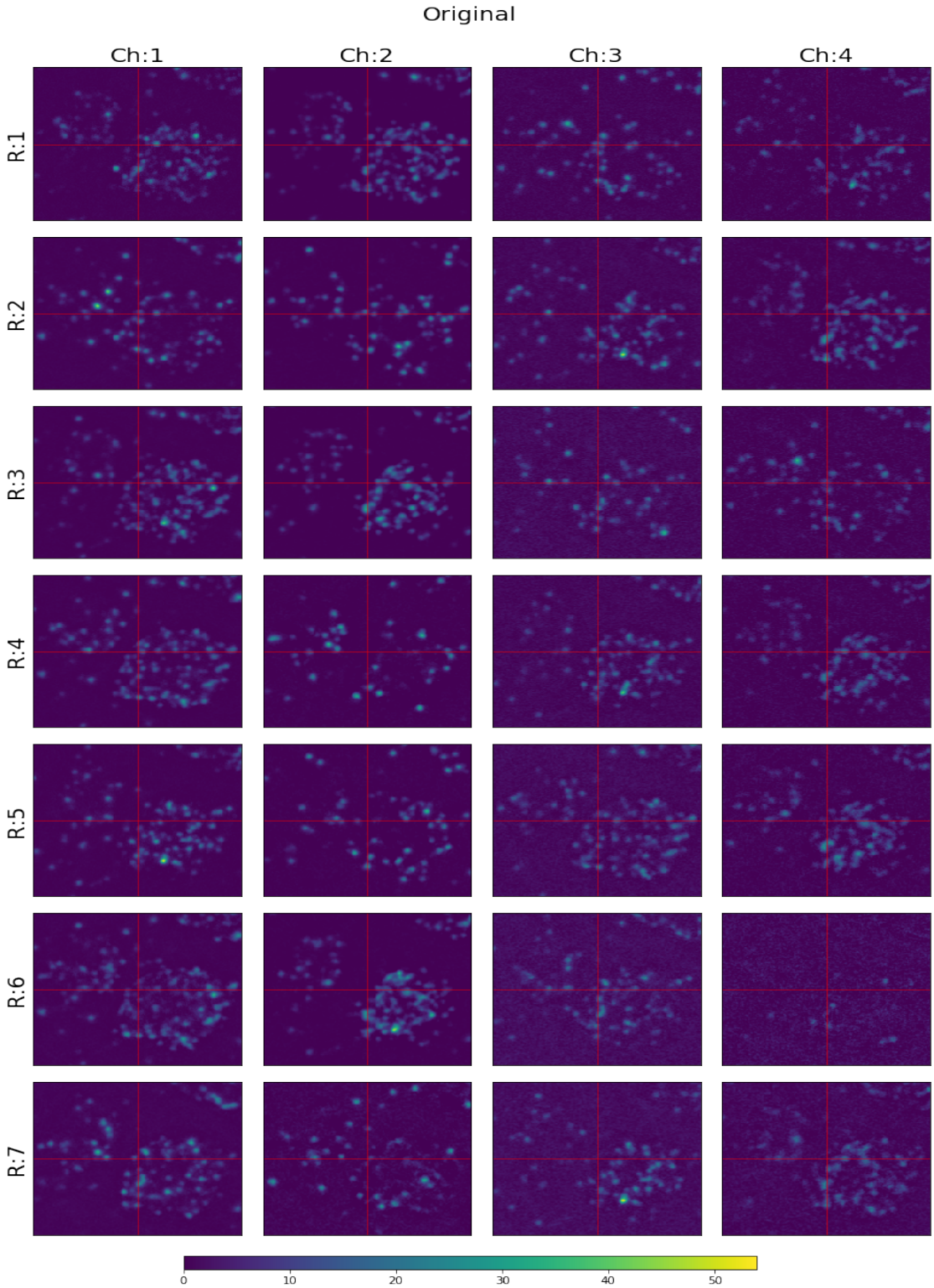

Figure 13: **Original data (X) after normalization for each round and channel.** In order to create clearer visualizations, we noise-normalized the data as described in Appendix A, so that images from all rounds and channel are on the same scale.

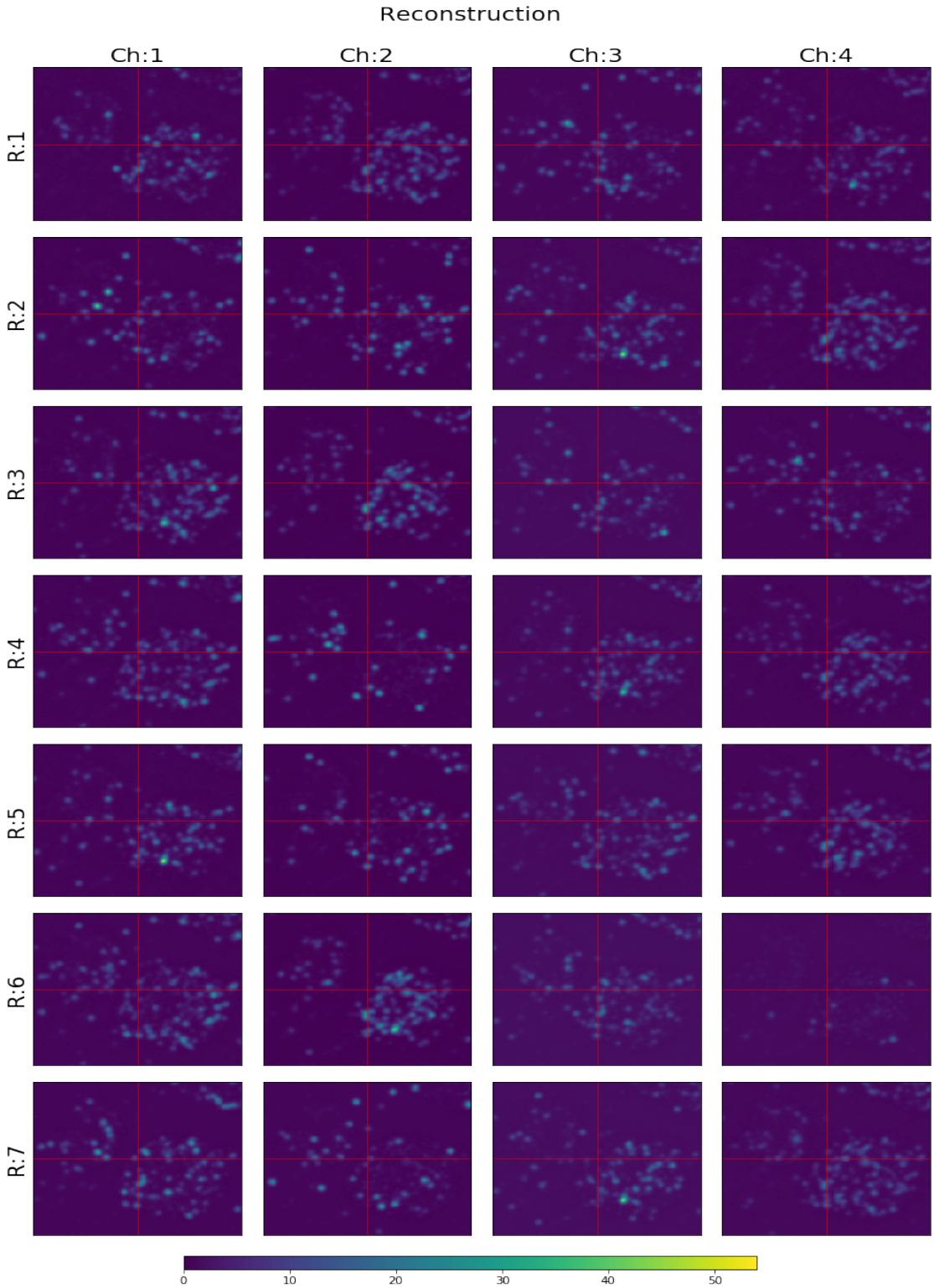

Figure 14: **Data reconstructed from BarDensr**. Under the BarDensr model, the fluorescence signal observed at each voxel in Figure 13 should be approximately given by the equations from Section 2. We here plot the results of those equations, visualized using the same colormap-intensity scale as used Figure 13. At least by eye, we see excellent agreement between the data and the model's predictions.

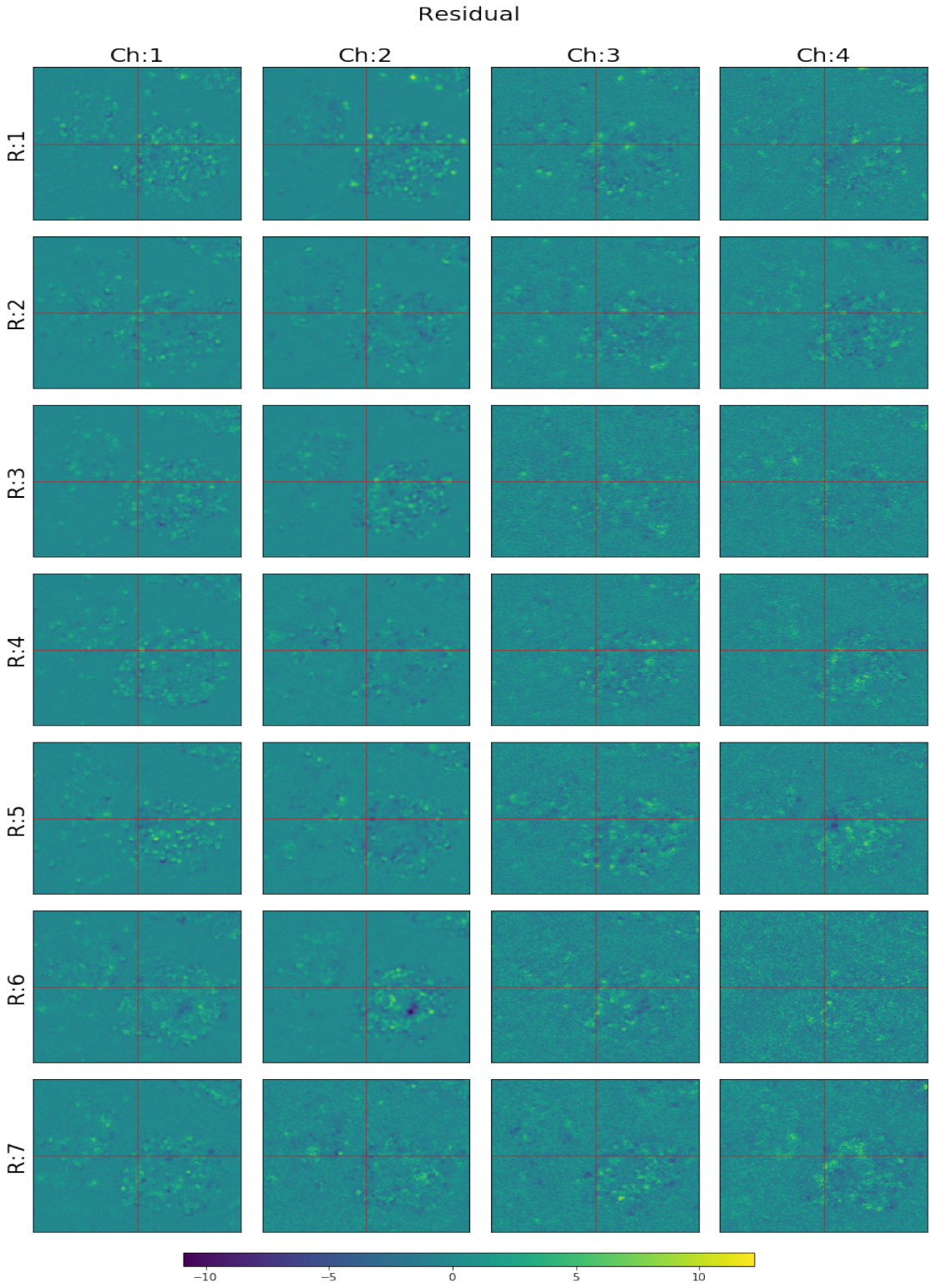

Figure 15: **Residuals.** As mentioned in Figure 14, the BarDensr model makes predictions about what the observed data should look like. There is broad agreement, but there is some disagreement. Here we highlight the the residual between the predictions and the data. Note the difference in scale compared to the previous two figures.

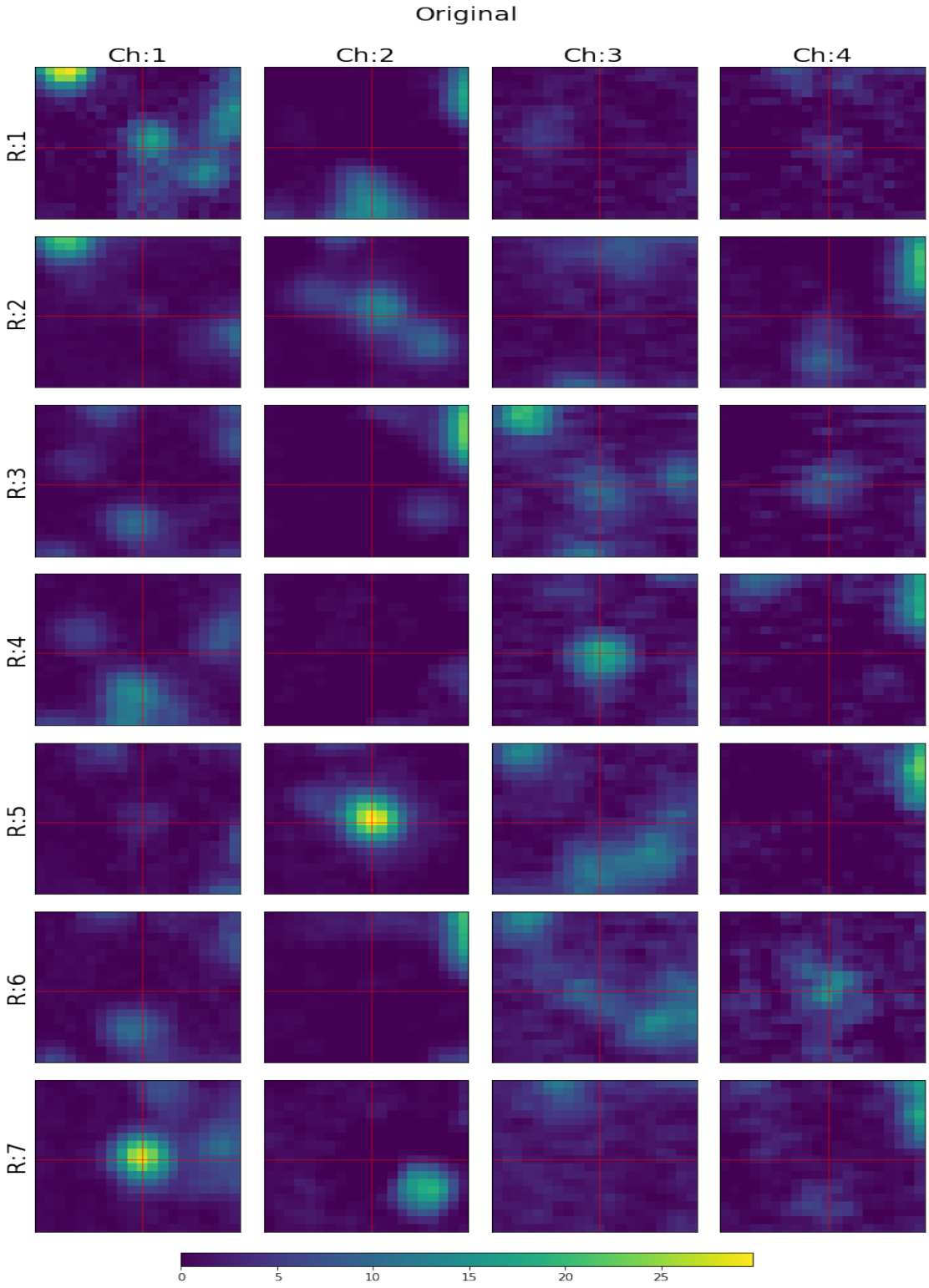

Figure 16: **Zoomed original data (X) after normalization for each frame**. Zoomed in for one of the target spots (a 20×20 region). See Figure 13 for more details.

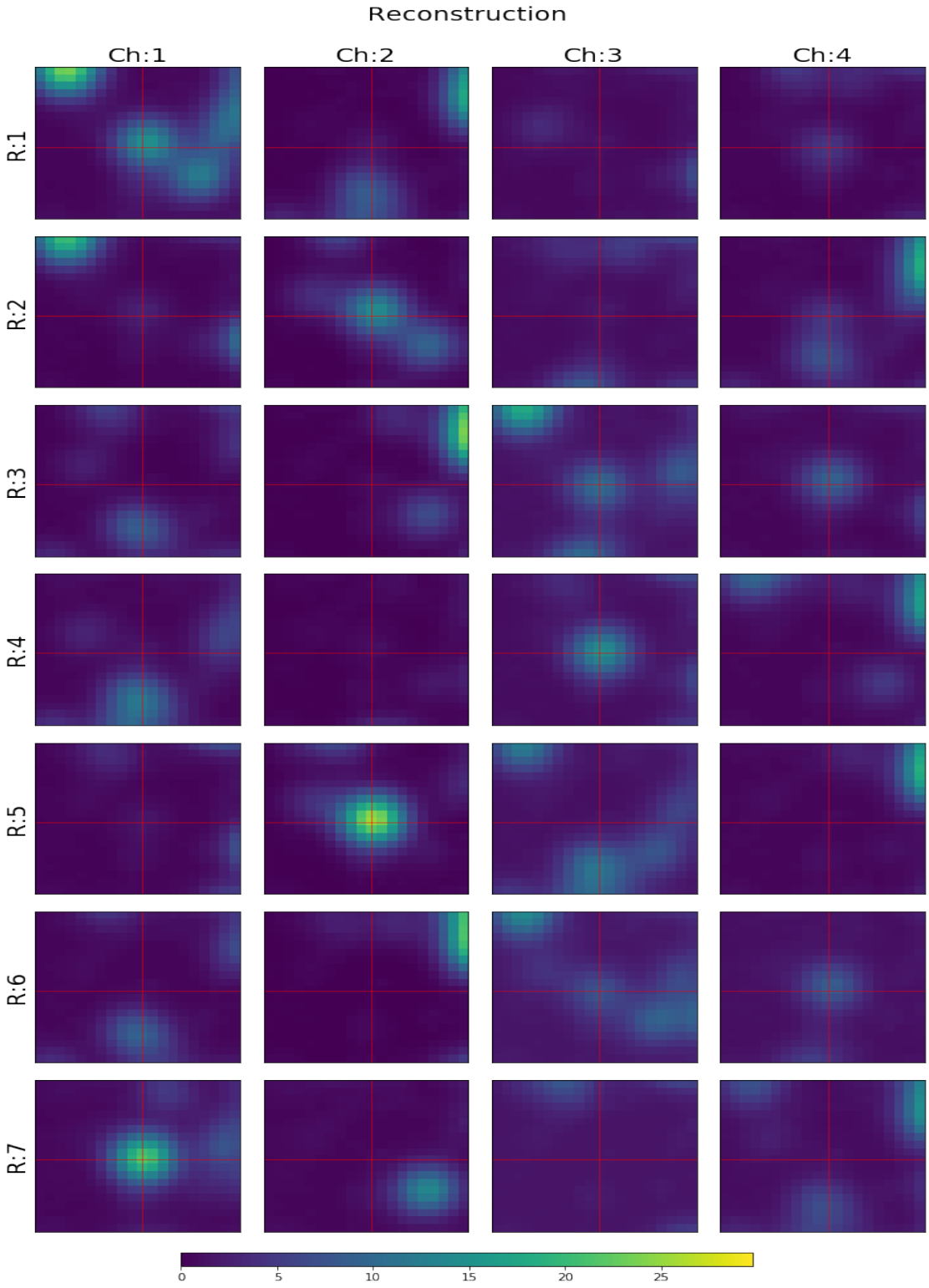

Figure 17: **Zoomed reconstructed data**. Zoomed in for the target spots (20×20), plotted at the same intensity scale as Figure 16. See Figure 14 for more details.

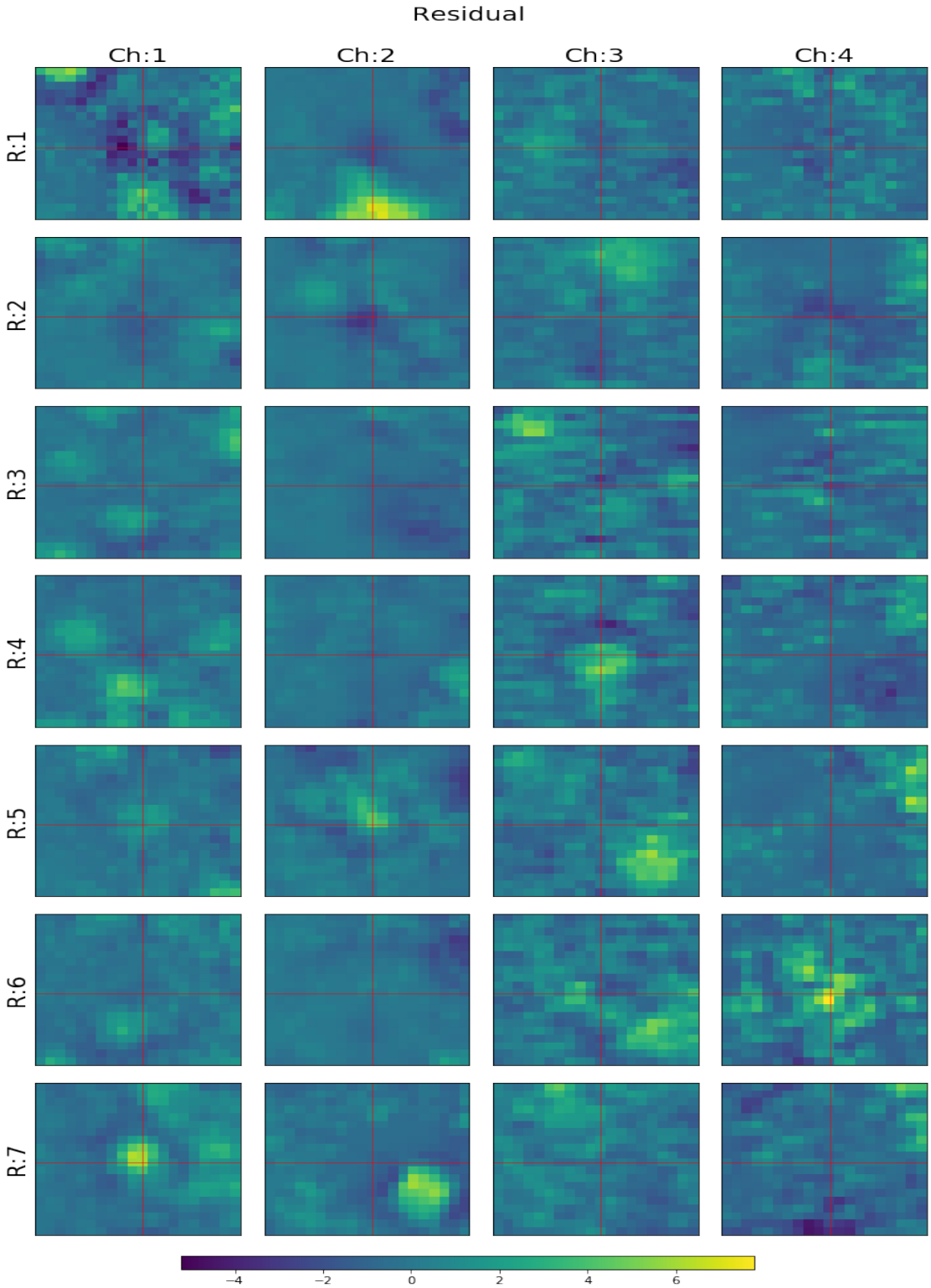

Figure 18: **Zoomed residual plot.** Zoomed in for the target spots (20×20). See Figure 15 for more details.

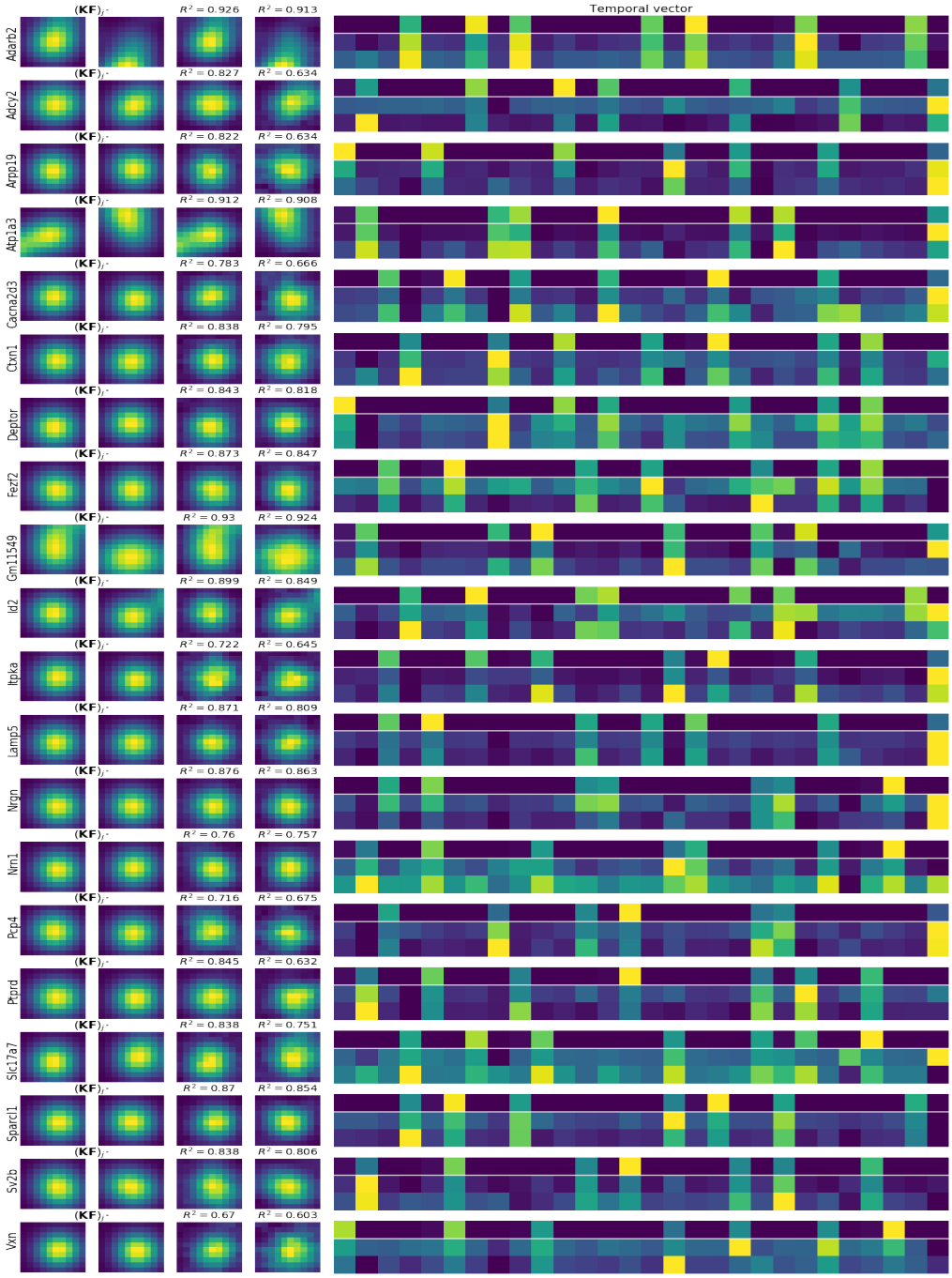

Figure 19: **Further SVD analysis of cleaned images for the top high-R^2 spots.** The figure supplements Figure 10 in the main text, and is structured in the same way, except that this plot shows more examples (with more barcodes and spots). Each row shows two spots for a given barcode. The first two columns show $(\mathbf{KF})_{j^*}$ cropped around the two spots; the third and forth columns show the top spatial singular vectors for the same crops. The final wide column shows the top temporal singular vectors for these spots, with the first row (above the thin white line) showing the scaled $\mathbf{G}_{j^*}$ learned from the model, and the following two rows showing the corresponding top temporal singular vectors for these spots. The two spots are ordered by R^2, which is computed as in Figure 9.

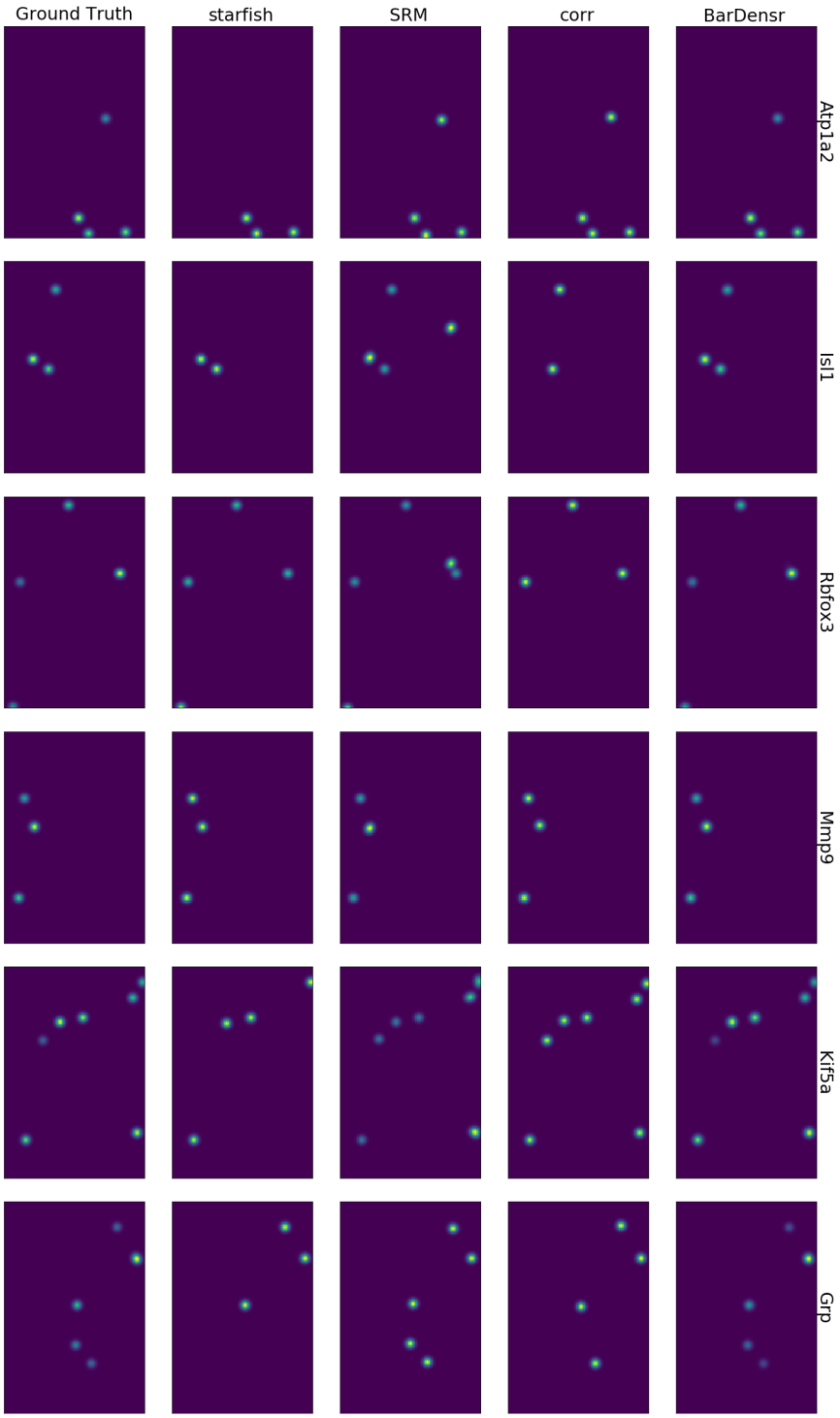

Figure 20: **Benchmarking results with no dropout**. Comparing *starfish*, SRM, 'corr', and BarDensr results, to the ground truth. Showing the top six barcodes with highest density (the gene density was generated randomly, see Appendix E). This figure corresponds to the top left plot in Figure 4 in the main text. Without dropout, BarDensr accurately detects the barcodes in the original data.

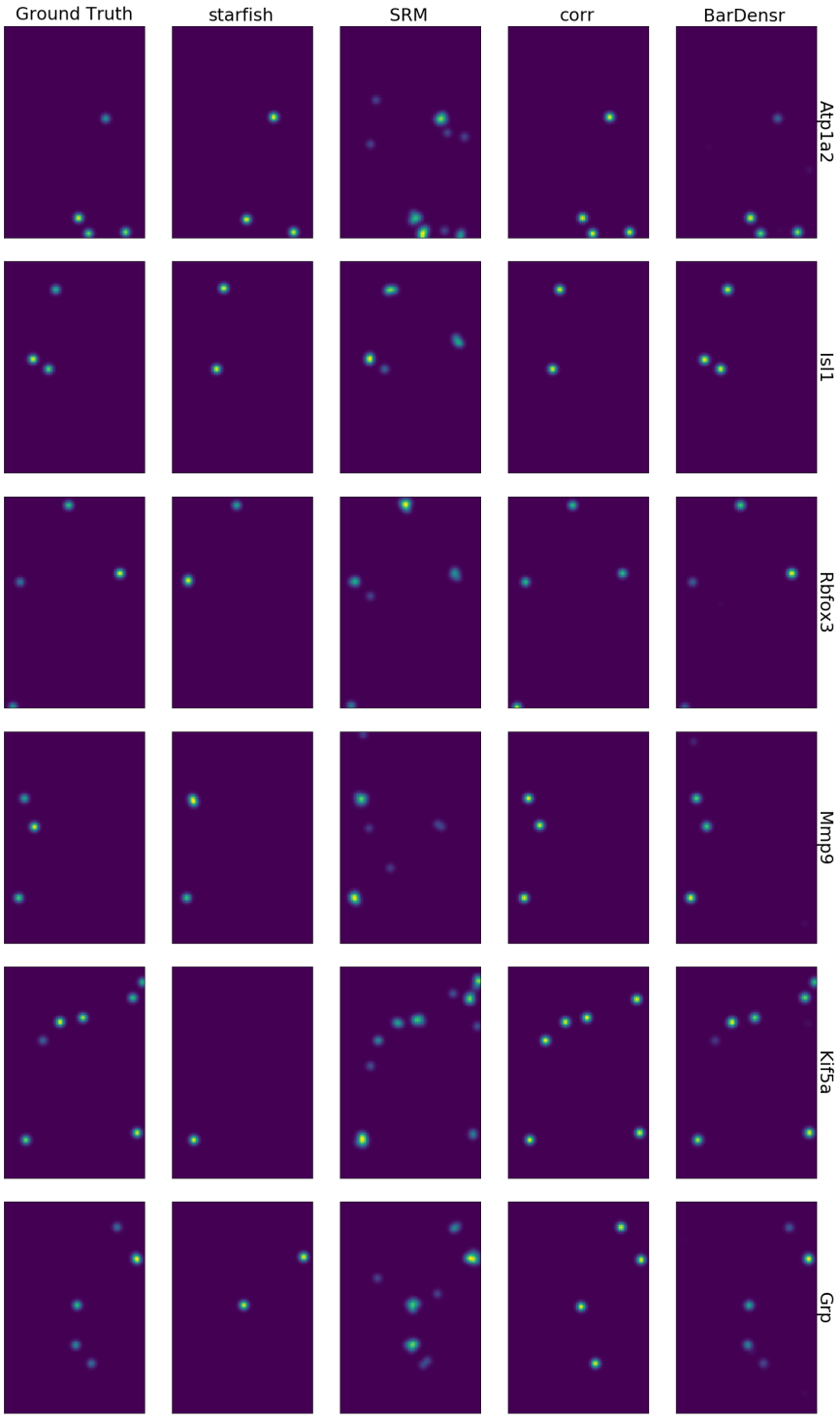

Figure 21: **Benchmarking results with** 50% **dropout**. Similar to Figure 20 but with dropout for 50% of the simulated spots. This figure corresponds to the top right plot in Figure 4 in the main text.

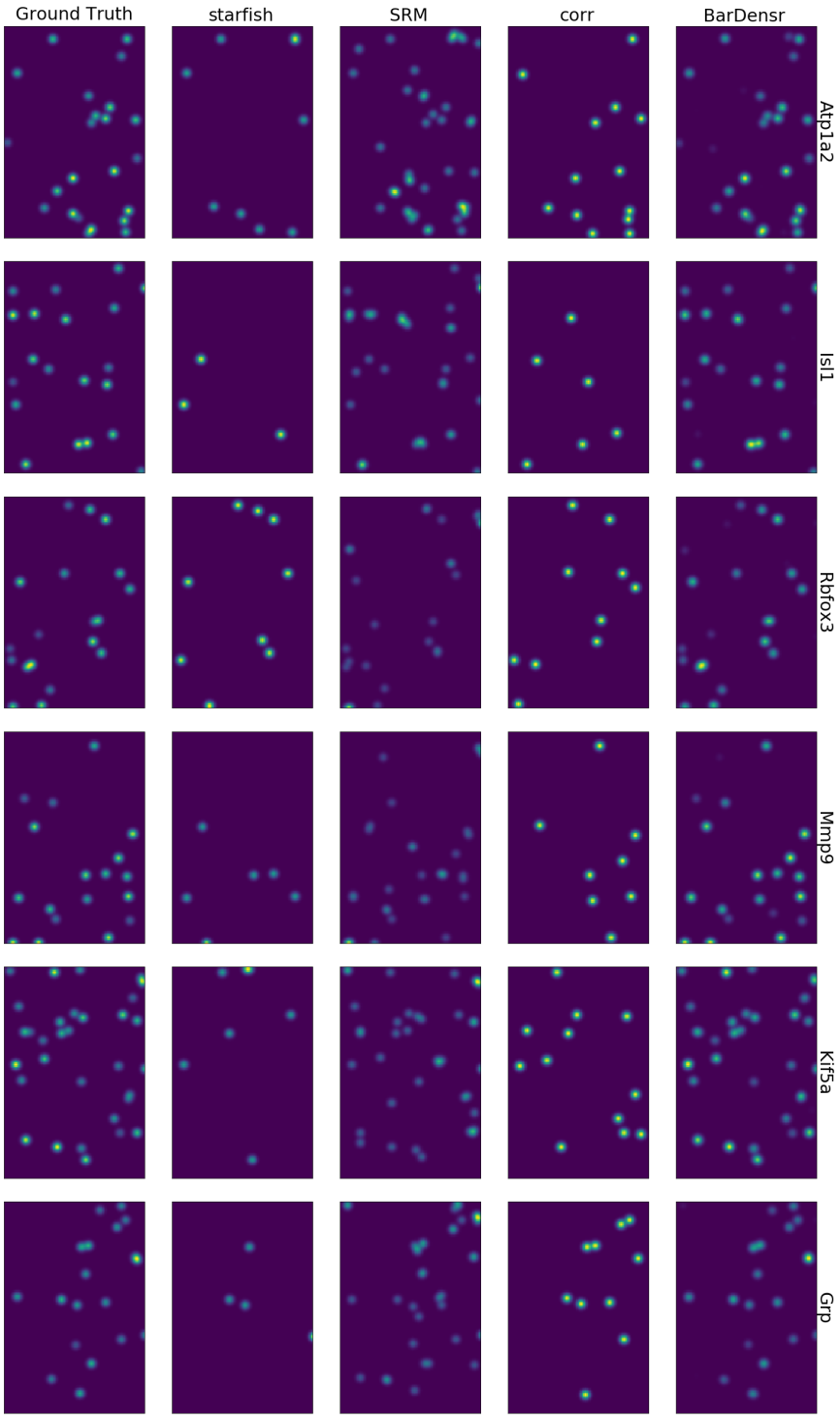

Figure 22: **Benchmarking results with no dropout, using five times denser simulation.** This is the similar process except that the spots density is five times denser than Figure 20 and 21. This figure corresponds to the bottom left plot in Figure 4 in the main text.

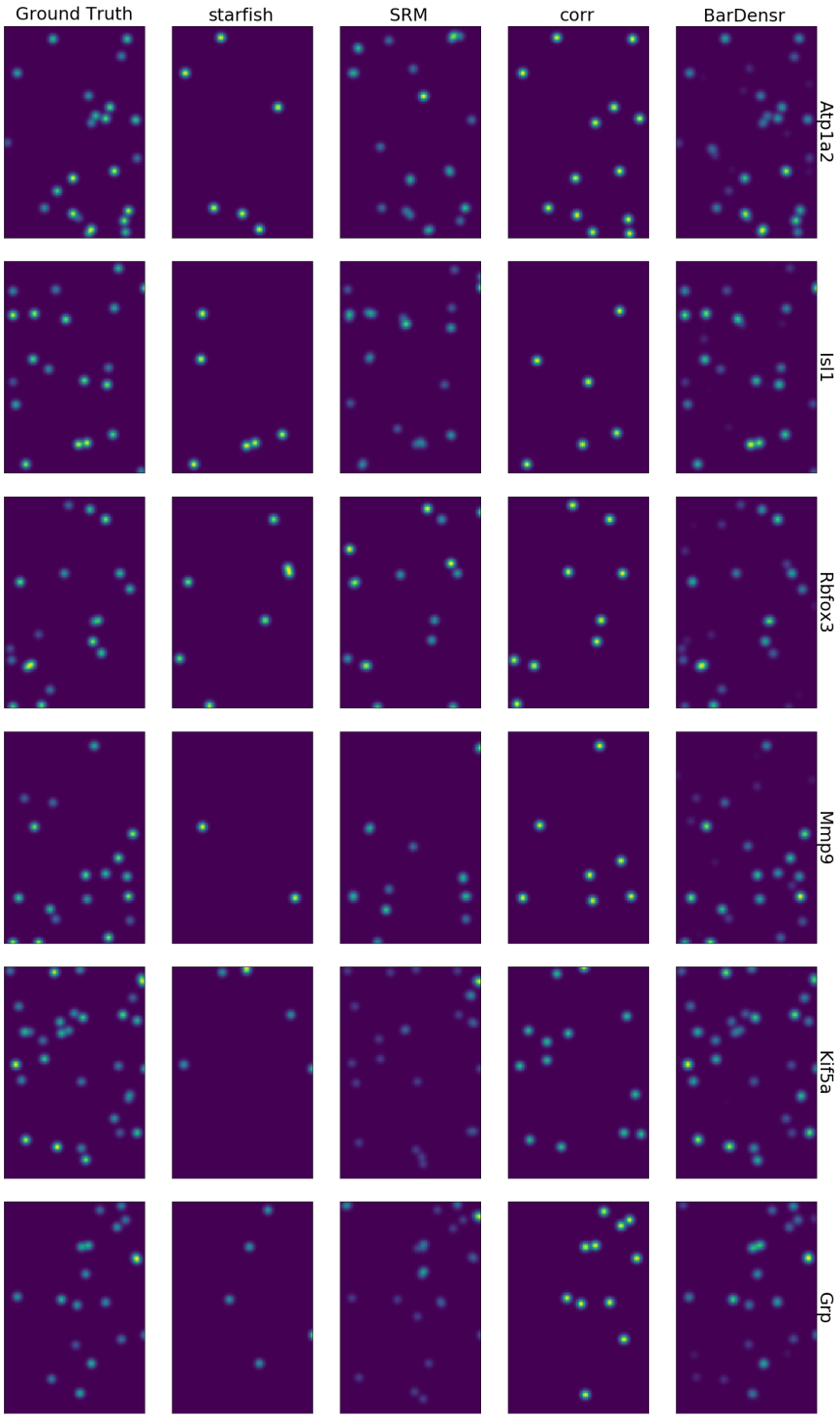

Figure 23: **Benchmarking results with 50% dropout, using five times denser simulation**. Similar to Figure 22 but with dropout for 50% of the simulated spots, for the denser simulation. Some missing spots (FN) can be observed from our model as well as other two methods (e.g., see the fifth row *Kif5a*). False discovery (FP) can also seen in this plot for SRM (e.g., see the third row *Rbfox3*). This figure corresponds to the bottom right plot in Figure 4 in the main text.

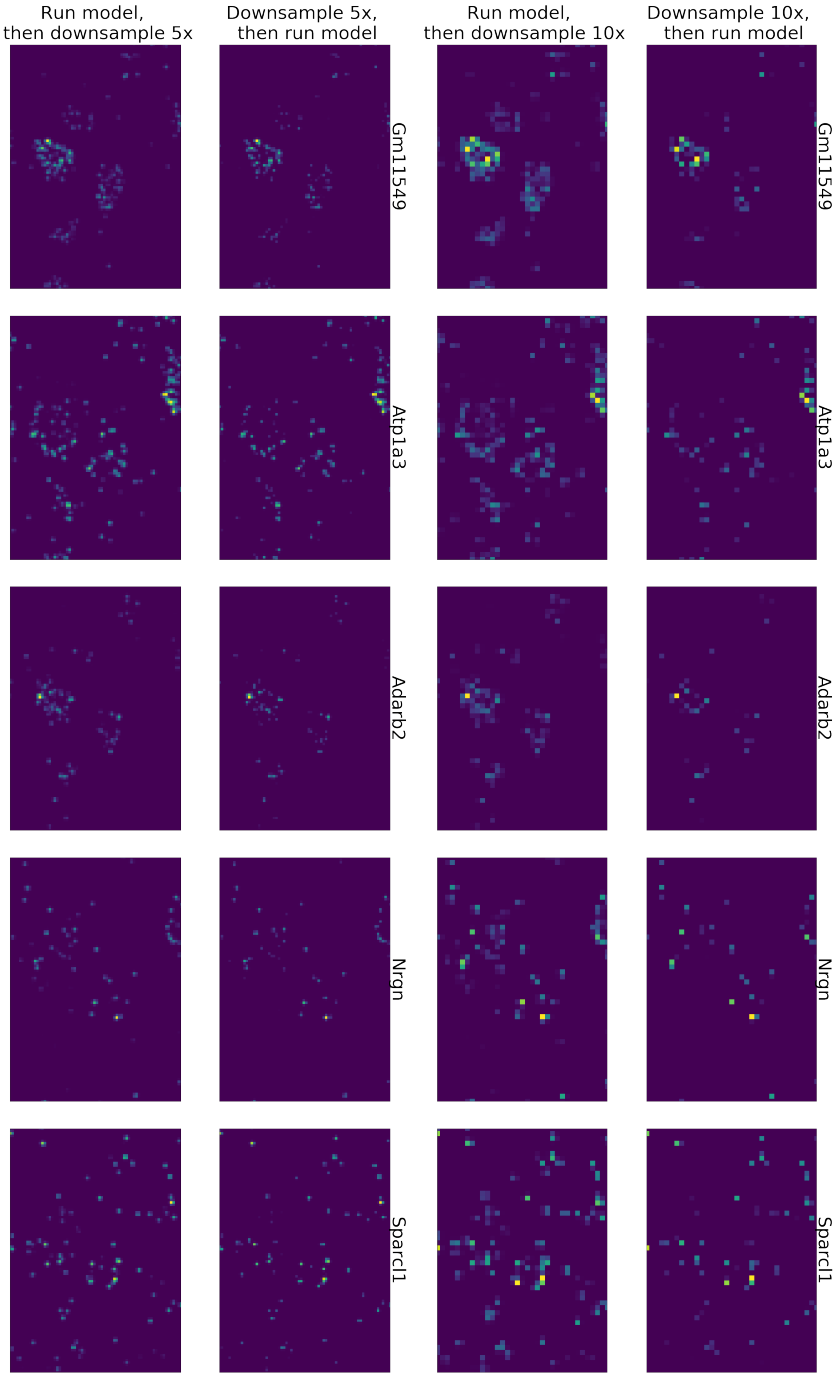

Figure 24: **Further demonstration of BarDensr applied to low-resolution data (supplementing Figure 6.)** To test BarDensr's performance on low-resolution data, we first run BarDensr on the original data, obtain rolony densities, and then finally downsample the rolony densities ('run-then-downsample'). Next, we run BarDensr on downsampled data and look at the learned rolony densities ('downsample-then-run'). For highly-expressed genes, these two results are nearly indistinguishable.

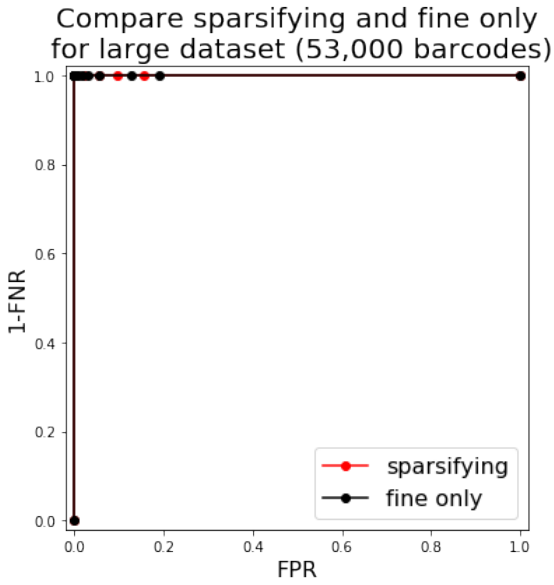

Figure 25: **BarDensr can be scaled up to a larger number of barcodes by using sparsified image.** To test if we can scale up BarDensr, we computed an ROC curve for the method using a simulated dataset with 53,000 barcodes and 17 sequencing rounds. After running the model on a $5\times$ downsampled 50×80 pixels simulated image, barcodes that are set to zero at the coarse scale were removed and the model was run at the original scale, with the parameters learned from the downsampled image as the initial conditions. See also Appendix H.

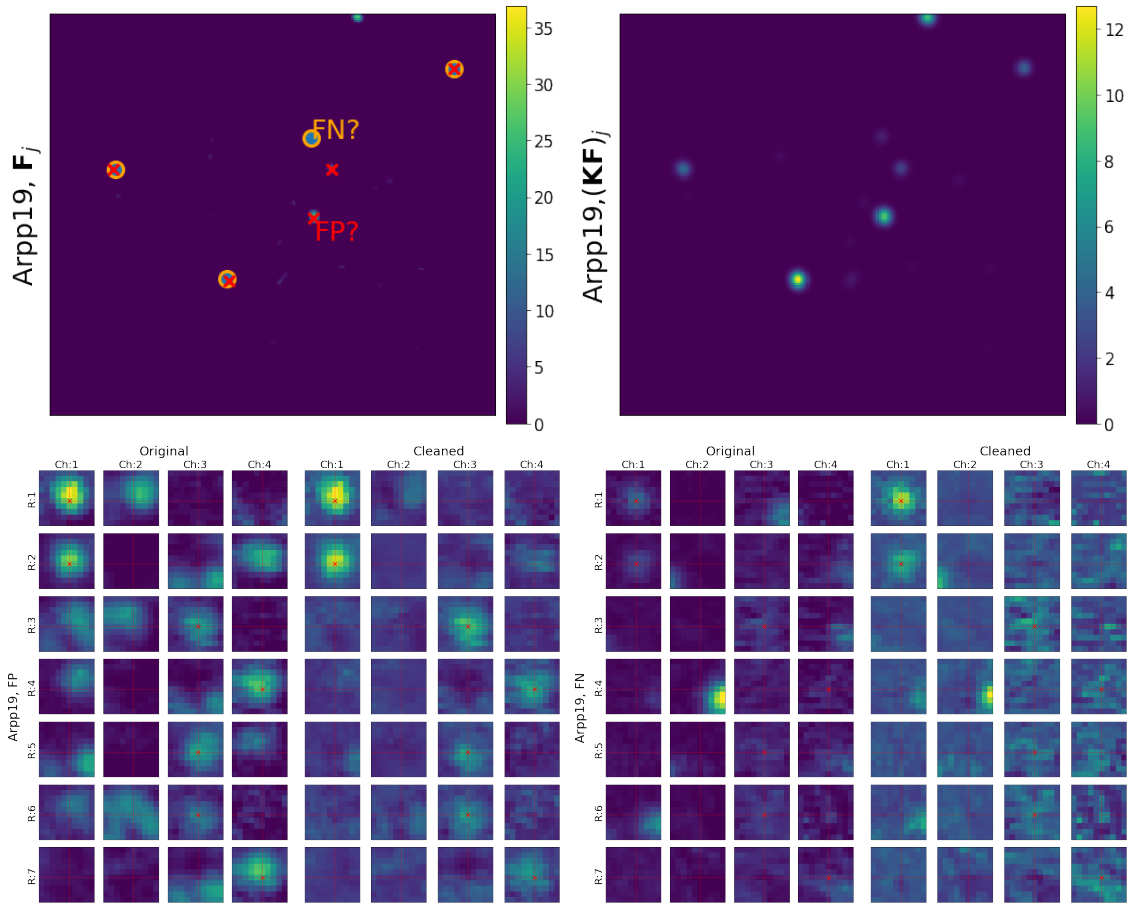

Figure 26: **Spatial rolony density comparison against hand-curated results.** On the top two plots, we show the rolony density $\mathbf{F}_j$ (left) and the blurred rolony density $(\mathbf{KF})_j$ (right) for gene *Arpp19*, derived from the experimental data. These rolony densities indicate the presence of *Arpp19*-rolonies. However, they might be incorrect, indicating that these detected rolonies might not be present in the real data. In this figure we investigate this question qualitatively. First, we compare with the rolony positions detected by a hand-curated method (as represented by orange circles on the left top plot) with the rolonies suggested by the rolony densities (as indicated by red crosses on the left top plot). We see a broad agreement. Where there is a point of disagreement, we can visualize the signal intensities in all the voxels near that point. The two plots on the bottom-left show the original data from a spot that was detected by BarDensr, but not detected in the hand-curated results (as indicated as False Positive (FP) in the top left plot); the left columns show the original image and the right columns show the 'cleaned' image (similar to Figure 9, see Equation 2 for details). The red cross in each round indicates the channels that are activated by this barcode. These crosses line up well with the observed signal, suggesting BarDensr has correctly identified a new rolony. It appears that the hand-curated method failed to detect this rolony because of the presence of nearby rolonies, leading to a mixed signal; BarDensr is specifically designed to handle these kinds of confusing situations. The two plots on the bottom-right show a spot which is detected in the hand-curated result but not detected by BarDensr (as indicated as False Negative (FN) in the top left plot). We show both the original data and the cleaned data, as in the bottom left plots. In this case, the data do not appear to support the presence of a rolony, suggesting BarDensr correctly rejected this region as a rolony and the the hand-curated approach labelled it incorrectly. We conjecture that the hand-curated approach misidentified this as a spot because of the signal arising from a nearby rolony in round 7, channel 4; this again created a mixture of signals BarDensr was better equipped to recognize.

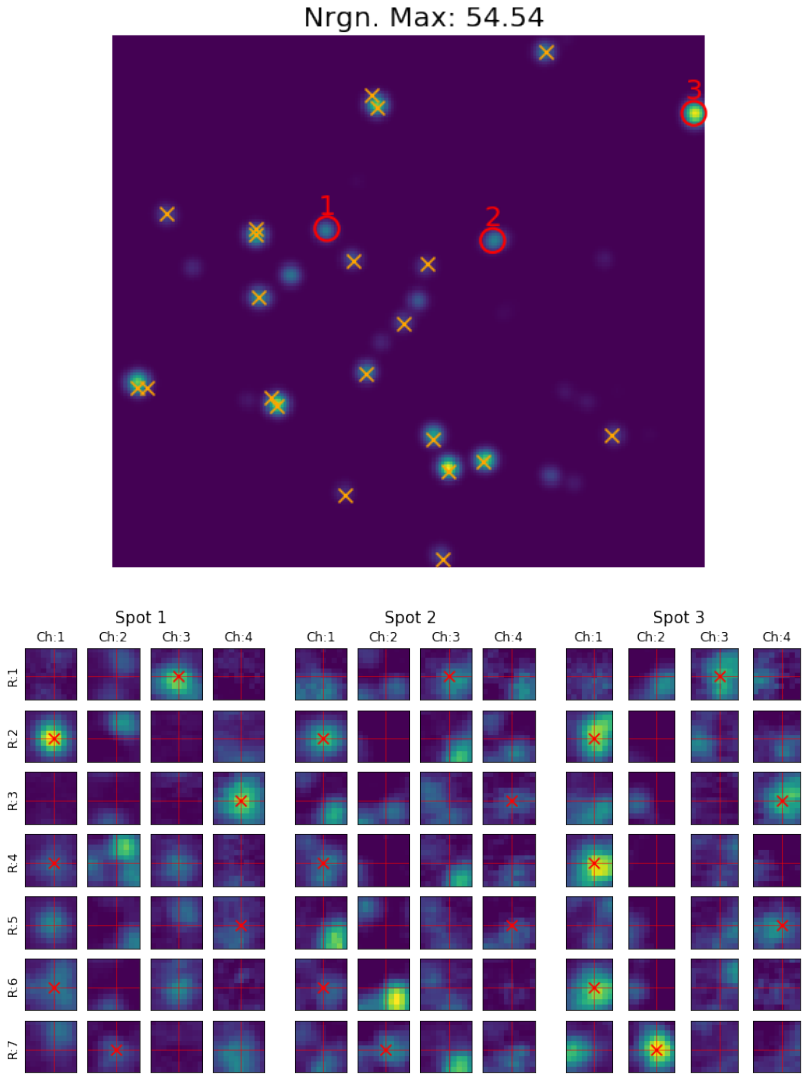

Figure 27: **Supplement to Figure 2 for *Nrgn*.** The top plot shows the same spatial rolony density of *Nrgn* as in Figure 2. The orange crosses indicate the spots detected in the hand-curated results. The three spots highlighted with red are further zoomed in the bottom, with their indices on the titles. These spots were detected to have large signal intensities by BarDensr, but were not detected in the hand-curated results. The correct barcode frames for *Nrgn* are indicated with red crosses in the bottom plots, suggesting that each of these spots appear to be well-modeled as *Nrgn* spots.

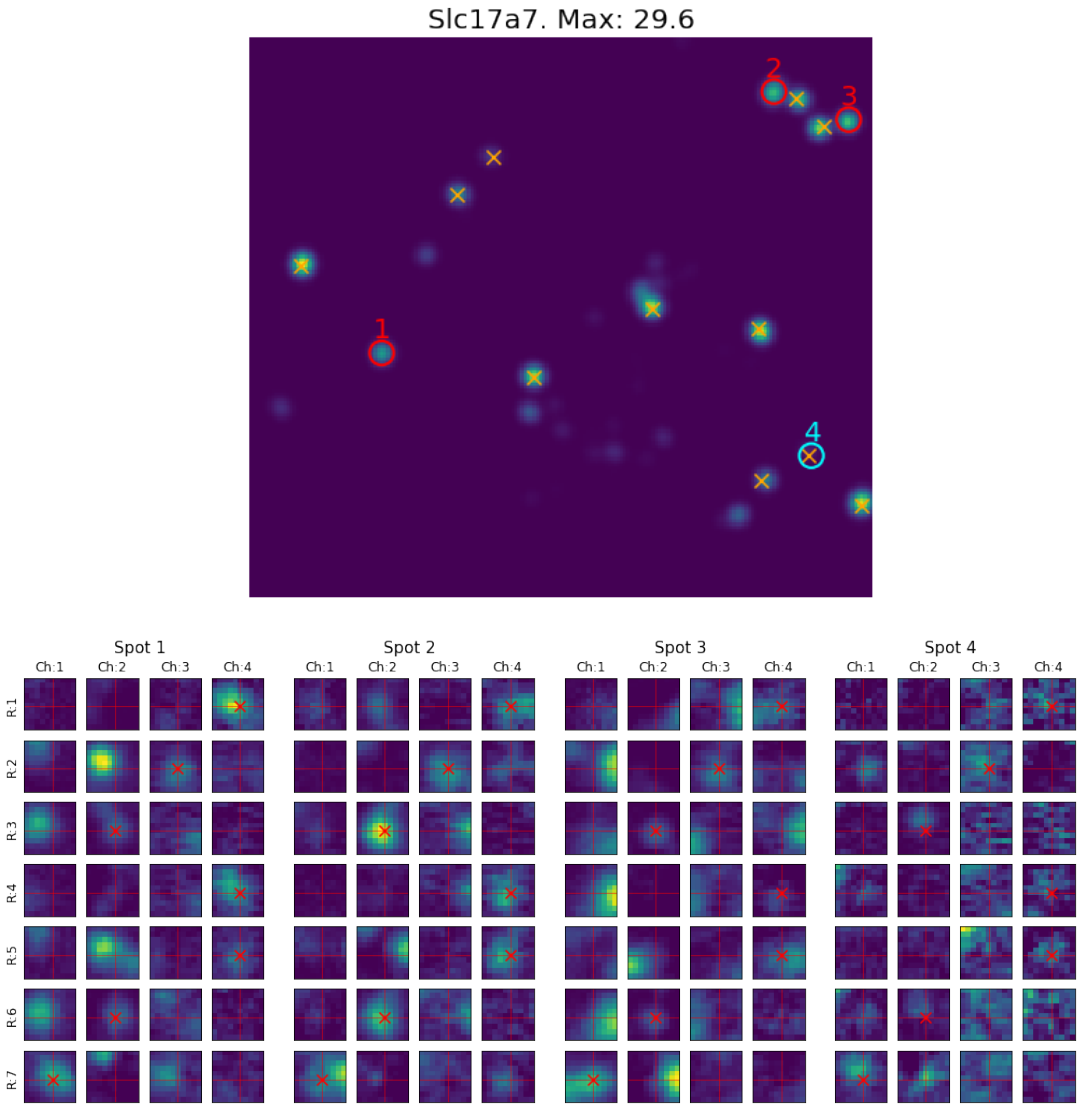

Figure 28: **Supplement to Figure 2 for *Slc17a7***. The top plot shows the same spatial rolony density of *Slac17a7* as in Figure 2. The orange crosses indicate spots that detected by hand-curated method. The four spots highlighted with red or cyan are further zoomed in the bottom, with their index on the titles. The first three spots (Spot 1 - 3, shown in red) were found by BarDensr but were not detected by hand-curated results, as in Figure 27. The fourth spot (Spot 4, shown in cyan) is the spot that is detected by hand-curated results, but no signal detected in BarDensr. The correct barcode frames for *Slc17a7* are indicated with red crosses in the bottom plots.

A Data preprocessing

The data is preprocessed before input into the model as follows: first the data was max-projected across all z-stacks. The channel color-mixing was corrected and the background was removed using rolling-ball background subtraction (Sternberg, 1983). Then the different image stacks were registered to the same voxels, using the Image Alignment Toolbox (ECC image alignment algorithm) (Evangelidis and Psarakis, 2008).

Finally, we performed a crude noise-normalization on each frame. First we estimated the noise level on each frame by spatially high-pass filtering (i.e., original image minus a Gaussian-filtered image, with a sigma of 2 pixels) to isolate spatially-uncorrelated noise, then computing the standard deviation. (See e.g. (Buchanan et al., 2018) for a related approach applied in the temporal domain.) Then we divided each original frame by its estimated noise scale to obtain the noise-normalized images.

B Hardware time and cost comparisons

To develop an efficient implementation of BarDensr on the NeuroCAAS cloud platform (Abe et al., 2020), we needed to find the most cost-effective hardware for the job. Using a 1000×1000 sized image from the experimental data described in the main text, we ran the model on several different AWS instance types. The most cost-effective machine was `m5.2xlarge`, which completed the analysis in three and a half minutes with a total cost of two cents. On the other extreme, the `p3.2xlarge` machine completed the analysis in one minute with a total cost of five cents. As a compromise between speed and cost, we settled on the `p2.xlarge` machine, which completes the analysis in two minutes with a total cost of three cents.

C Single Round Matching (SRM) and 'hand-curated' method

We compared BarDensr against several different alternative methods, including one we call 'SRM.' This method is an implementation of the widely-used 'blobs-first' algorithms suggested in the literature (Wang et al., 2018; Qian et al., 2020). First, blobs were detected in every channel in the first round by finding local maxima on a per-channel basis. These blobs were then used as a reference in understanding subsequent rounds (the first round is used as the reference since it usually has the least corruption by noise and artifacts, such as phasing and photo-bleaching).

At each detected rolony position, SRM then read out the signal intensities from all channels/rounds as a vector of length $R \times C$. This vector was compared against each barcode in the library. Each detected rolony was assigned to the barcode with the greatest similarity (as measured by a dot-product). For some rolonies the similarity was low to all barcodes; these rolonies were filtered out. Thresholds were determined using the Bayes optimization method from (Nogueira, 2014).

The 'hand-curated' method in the main text corresponds to SRM described here. After the process of SRM with the chosen threshold, we manually checked the detected rolonies and the assigned barcodes to make sure that the results were reasonable.

D Correlation-based method

We also compared BarDensr against a 'correlation-based method' (Moffitt et al., 2016, 2018). This approach begins by computing a vector of length $R \times C$ for every voxel, indicating the fluorescence signal in each round and channel at that voxel. At each voxel, for each barcode, the cosine distance between this vector and the barcode was computed. The barcode with the minimum cosine distance was assigned to be a *potential* gene identity for each voxel. Finally, a 'minimum distance image' was constructed: for each voxel, this image contains the cosine distance between that voxel's $R \times C$ vector and the barcode which it is most similar to. Coordinates of blobs were

found by a seeking local minima in this image. Thresholds were again determined via (Nogueira, 2014).

E Simulation

Generating arbitrary distribution for genes

For the simulation benchmarking in Figures 4 and 20-23, we used barcodes from a STARmap experiment (developed in (Wang et al., 2018), unpublished data), with total of 57 genes. This data is similar to our original experimental data in that it has six rounds and four channels in total, and the scale of the number of barcodes is also similar. This data was chosen instead of our experimental data in order to directly apply *starfish* method (the *starfish* application on our original experimental data was not available at the time this analysis was conducted). In creating simulations, we wanted to accurately represent the uneven distribution of genes; in real data some genes are more abundant than others. Therefore, we began by randomly selecting 10 out of 57 genes to be 'abundant' genes. In generating a dataset with simulated rolonies, we created rolonies with these abundant genes roughly ten time more often than the other rolonies.

Dropout

We used two setups to generate simulated testing data: without dropout and with dropout. In the experimental data, it is commonly observed that a small portion of rolonies disappear/diminish in some rounds. (Based on our visual inspection of the experimental data, qualitative dropout events were observed in $< 5\%$ of the rolonies detected in this data, but we did not attempt to estimate this dropout rate precisely.) In the 'Dropout' simulations, we tried to mimic this phenomenon.

Specifically, for the 'no dropout' simulations, we generated the data with the following process. 1. Generate the spot position with a uniform distribution across the voxels. 2. For each spot position, generate the spot identity (gene) using a prespecified gene distribution (as discussed above). 3. For each position m and gene j pair from steps 1 and 2, the *magnitude* of the rolony density at (m, j) was generated from a uniform distribution in the range $(10, 40)$. We use these values to fill in the rolony density, $\mathbf{F}$. 4. We then generate synthetic data according to the BarDensr model. Finally, we add some speckle noise. Note that in our simulation, parameters such as the per-frame intensity (α), phasing (ρ), and color-mixing (φ) were left out for simplicity.

For the 'dropout' simulations, 50% of the simulated spots were randomly selected to be the 'dropout spots.' For each 'dropout spot', one round is selected randomly and the signal intensity for that spot for that round is diminished to 10% of the original signal. The simulation process is otherwise the same.

Hybrid simulation

To test the efficacy on our model on even more realistic data, we used a 'hybrid simulation,' as in e.g. (Pachitariu et al., 2016). In essence, the hybrid simulation creates a fake dataset by superimposing the real data with additional synthetic rolonies to the data. The question is whether the algorithm can at least find the synthetic rolonies which were added. The results are shown in Figure 5. In generating the synthetic rolonies, we used the codebook used in the original experiment, and the genes of the synthetic rolonies were given by the observed gene distribution from the hand-curated analysis of the same dataset.

We ran a few different versions of these simulations. There were several key parameters which we varied:

- We had both 'dropout' and 'no dropout' versions; in the dropout versions some of the synthetic rolonies had signal in one round diminished.
- We could vary the number of synthetic spots which were injected (S). According to the hand-curated analysis of the real data, the real data contained approximately 400 spots in

this field of view. We investigated how the number of spots affected the results, looking at $S \in \{30, 80, 100, 200\}$.

- We could vary the intensity of the synthetic spots, relative to the maximum intensity observed in the data. We varied this between 10% and 90%.

F Error analysis

In simulated data, we can exactly quantify the different kinds of errors that BarDensr makes, by comparing against the true rolony positions used to make the simulated data. We first examined the 'total hit rate' in our Receiver Operating Characteristic curve (ROC curve, shown as the dotted lines in Figure 4). For this ROC, we consider a spot to be successfully detected by the algorithm as long as the algorithm finds *any rolony* near the site of a true rolony – even if the algorithm incorrectly assigns the gene associated with that true rolony. The ROC for BarDensr clings closely to the upper left side of the plot, suggesting nearly perfect performance. We also looked at what we call the 'hit rate' – for this ROC we consider a spot to be succesfully detected only if the algorithm detects a rolony in the right place and of the right gene. Figure 4 shows the results, suggesting most errors were caused by gene mis-identification.

G Cell segmentation

For the bottom plots on Figure 6, we first segmented the cells in the selected region with the following process. We first obtained the max projection across $R \times C$ frames from the the image stacks. After applying a Gaussian filter with a sigma of 8 pixels, all the pixels with intensity lower than 10% of the maximum intensity were assigned to be zero. Finally, we used a watershed segmentation algorithm to identify contiguous cellular regions. This results in 47 segmented cells in the region. Four of these occupied less than 100 pixels in total and were removed from the analysis.

H Sparsifying and coarse-to-fine

H.1 Handling tens of thousands of barcodes with sparsifying and coarse-to-fine

To scale up BarDensr, we tested if eliminating unnecessary barcodes could help accelerate computation. For this purpose, we set up a simulation with 53,000 barcodes and 17 sequencing rounds, similar to the setup in a larger scale experiment such as (Chen et al., 2018) (Figure 25). We generated a dataset with 50×80 voxels and a total of 40 spots. We then processed this data in two steps.

In the first step (the 'coarse' step), the image was five times downsampled, and BarDensr was applied to the downsampled data. For each gene, if the maximum intensity for a rolony density was lower than 10^{-5}, that gene was considered to be absent.

In the second step (the 'sparsified fine' step), we then applied BarDensr to the original, full-resolution data – but only using those barcodes that weren't 'absent' in the previous step. To make this approach even faster, we also used the learned parameters from the downsampled data as initial conditions for the algorithm's run on the full-resolution data. Moreover, in this second step, the parameter b and α were not updated at all, since we found that they were learned quite accurately in the first step.

After the first step, 71 out of 53,000 barcodes were kept to be used in the second step. This sparsified coarse-to-fine approach sped up our analysis by more than a factor of 10.

H.2 Coarse-to-fine

The method described above involves both sparsifying and using a kind of coarse-to-fine approach. We also investigated performance using only the coarse-to-fine aspect. These investigations are summarized in Figure 7. First, a 1000×1000 image was simulated with $20,000$ spots following the simulation process described above, with no dropout (Appendix E). The image was then downsampled to 500×500 and BarDensr was applied to this downsampled data to estimate $\mathbf{F}, \alpha, a$ and b. with 20 iterations. These parameters were then used as the initial conditions to run the model with the full size image (note that in order to use the parameters from the downsampled image to initialize the full-resolution model, we needed to upsample $\mathbf{F}$ and a). Finally, we tried running the model directly on the full-resolution data (without using the downsampled data to get initial conditions). We then compared the results. Both approaches work better if they are allowed to run longer, because they use an iterative approach to optimize the loss function. Eventually, both approaches yield the same results. However, Figure 7 shows that the coarse-to-fine approach is able to achieve the best possible performance three times faster.

H.3 Sparsifying leads to speedups even in the case of a small barcode library

We also tested the sparsifying approach on the experimental data (Figure 8). The original 1000×1000 image was first five times downsampled to obtain a 200×200 'coarse' image, and the parameters were learned from this downsampled image ('coarse' process). After five times upsampling of learned $\mathbf{F}$ to obtain the image on the original scale, both the original image and the upsampled $\mathbf{F}$ were split into 4×4 patches. Each patch is of size 250×250 (plus 20 pixel edges in the end of both dimension, whenever the coverage of the patch does not exceed the image region). These 16 patches cover the entire 1000×1000 image with overlaps on the edge regions. For each patch, the barcodes that have the maximum intensity lower than the maximum intensity of the two unused barcodes in the upsampled $\mathbf{F}$ were considered to be absent from the region. Each patch was then used to fit the model at the original scale to learn $\mathbf{F}$ and a, but using a smaller amount of barcodes for the binary codebook matrix $\mathbf{B}$. For this 'fine' process, the parameter b and α were not updated but the ones learned from the coarse process were used. After the model was run on all the 16 patches, we needed to stitch the results back together into a single result for the entire field of view. This is slightly involved, because the patches concerned overlapping regions of voxels. Indeed, we insured that the border-regions between any two patches contained 20 pixels of overlap. For each voxel in these overlap regions, we used the signal from the patch whose center was closest to that voxel. The results were compared to the 'fine only' approach. We filled $\mathbf{F}$ with zero for the removed barcodes in each patch, in order to keep the original dimension for the following process.

In order to compare the agreement between 'sparsifying' approach and the 'fine only' approach, we computed two ROC curves. In each curve, one of the results was used as the gold standard for the other. Specifically, when using the 'fine only' as the gold standard, a threshold was determined based on the maximum intensity of the unused barcodes in the 'fine only' results, and a binary matrix of the size of the original image (1000×1000) was generated for each barcode ('0' indicates there is no signal, '1' indicates there is signal, for each pixel), which is the final gold standard to compare with the sparsifying result. The same process applies when using the sparsifying result as the gold standard. For computing ROC, rolonies within 3 pixel radius of $\mathbf{F}$ with the same barcode j were considered to belong to the same rolony.

I Algorithm Details

The primary computational challenge of this method is to solve a constrained optimization problem.

$$\begin{aligned} \min_{\theta\in\Theta} \quad & L_{\text{sparsity}}(\theta) \\ \text{subject to} \quad & L_{\text{reconstruction}}(\theta) \le \omega, \end{aligned}$$

where

$$\begin{aligned} L_{\text{reconstruction}}(\theta) &\triangleq \sum_{m,r,c}\left(X_{m,r,c} - \left(a_m + b_{r,c} + \alpha_{r,c}\sum_{j,m',c'}^{J,M,C}\mathbf{K}_{m,m'}\mathbf{F}_{m',j}\varphi_{c,c'}\mathbf{Z}_{r,c',j}\right)\right)^2 \\ L_{\text{sparsity}}(\theta) &\triangleq \sum_{m,r,c,c',j}\alpha_{r,c}\mathbf{F}_{m,j}\phi_{c,c'}\mathbf{Z}_{r,c',j} \\ \theta &= (\mathbf{F},\rho,\alpha,b,a,\varphi)\,. \end{aligned}$$

Here Θ denotes the set of feasible parameters; in our case, Θ simply requires that all variables are at least 10^{-10}. The threshold 10^{-10} was chosen somewhat arbitrarily and serves to ensure numerical stability of the optimization process. Technically we also believe that $\rho_c < 1$ for every c, but in practice we found it unnecessary to enforce this constraint.

We approach the reconstruction constraint using Lagrange multipliers. We define:

$$\begin{aligned} \mathcal{L}(\theta,\lambda) &= L_{\text{reconstruction}}(\theta) + \lambda L_{\text{sparsity}}(\theta) \\ \theta^*(\lambda) &= \arg\min_{\theta\in\Theta}\mathcal{L}(\theta,\lambda) \\ L^*_{\text{reconstruction}}(\lambda) &= L(\theta^*(\lambda)). \end{aligned}$$

Assuming we can evaluate $L^*_{\text{reconstruction}}(\lambda)$, we can solve the overall constrained optimization problem by taking

$$\lambda^* = \max\{\lambda :\ L^*_{\text{reconstruction}}(\lambda) \le \omega\}$$

and taking our final parameters to be $\theta^*(\lambda^*)$. It is unclear whether strong duality holds in this case, so the resulting parameters may not be optimal. However, in practice we find that they give useful results for uncovering rolonies.

In conclusion, to solve our overall problem it suffices to be able to solve $\min_{\theta\in\Theta}\mathcal{L}(\theta,\lambda)$ for any fixed λ. We approach this problem via a blockwise coordinate descent approach. Specifically, we start with an initial guess and iterate through a variety of updates until convergence is achieved. Throughout, we will use the notations

$$\begin{aligned} [x]_+ &= \begin{cases} x & \text{if } x > 10^{-10} \\ 10^{-10} & \text{otherwise} \end{cases} \\ \texttt{xmab}_{m,r,c} &= X_{m,r,c} - a_m - b_{r,c} \\ X^{(j)}_{m,r,c} &= X_{m,r,c} - a_m - b_{r,c} - \alpha_{r,c}\sum_{j'\neq j}\sum_{m',c'}\mathbf{K}_{m,m'}\mathbf{F}_{m',j'}\varphi_{c,c'}\mathbf{Z}_{r,c',j'} \\ \tilde{\mathbf{F}}_{m,j} &= \sum_{m'}\mathbf{K}_{m,m'}\mathbf{F}_{m',j}. \end{aligned}$$

α update. For each r, c, the relevant portion of the loss for $\alpha_{r,c}$ is given by

$$\begin{aligned} \mathcal{L}(\alpha_{r,c}) = \sum_m &\left(X_{m,r,c} - \left(a_m + b_{r,c} + \alpha_{r,c}\sum_{j,c'}\tilde{\mathbf{F}}_{m,j}\varphi_{c,c'}\mathbf{Z}_{r,c',j}\right)\right)^2 \\ &+ \lambda\sum_{m,c',j}\alpha_{r,c}\mathbf{F}_{m,j}\varphi_{c,c'}\mathbf{Z}_{r,c',j} + \cdots. \end{aligned}$$

Fixing all other variables, subject to the constraint that $\alpha_{r,c} \geq 10^{-10}$, the lowest possible value of this loss is given by

$$\alpha_{r,c} \leftarrow \left[\frac{\sum_m \left(\texttt{xmab}_{m,r,c} \sum_{j,c'} \tilde{\mathbf{F}}_{m,j} \varphi_{c,c'} \mathbf{Z}_{r,c',j} \right) - \sum_{m,c',j} \mathbf{F}_{m,j} \varphi_{c,c'} \mathbf{Z}_{r,c',j}}{\sum_m \left(\sum_{j,c'} \tilde{\mathbf{F}}_{m,j} \varphi_{c,c'} \mathbf{Z}_{r,c',j} \right)^2} \right]_+ .$$

Note that this update can be done in parallel across all r, c.

ρ update. We update the variable ρ via a line-search. One at a time, we look at ρ_c and consider possible values for this parameter in the interval $[\rho_c/2, 3\rho_c/2]$. We search for values of ρ_c in this interval which minimizes the loss.

a, b updates. Fixing all other variables, the best possible values for a are easy to find. The same goes for b. These values are given by

$$a_m \leftarrow \left[\frac{1}{RC} \sum_{r,c} \left(X_{m,r,c} - b_{r,c} - \alpha_{r,c} \sum_{j,c'} \tilde{\mathbf{F}}_{m,j} \varphi_{c,c'} \mathbf{Z}_{r,c',j} \right) \right]_+$$

$$b_{rc} \leftarrow \left[\frac{1}{M} \sum_{m} \left(X_{m,r,c} - a_m - \alpha_{r,c} \sum_{j,c'} \tilde{\mathbf{F}}_{m,j} \varphi_{c,c'} \mathbf{Z}_{r,c',j} \right) \right]_+ .$$

F update. We make updates to $\mathbf{F}$ one column at a time. We select a random column, j^*, and then update the values of $\{\mathbf{F}_{m,j^*}\}_{m \in \{1 \cdots M\}}$. We update these values via a projected coordinate descent algorithm. Let f denote the jth column of $\mathbf{F}$ and define

$$g_{r,c} = \alpha_{r,c} \sum_{m',c'} \varphi_{c,c'} \mathbf{Z}_{r,c',j^*}$$

$$\|g\|_2^2 = \sum_{r,c} g_{r,c}^2$$

$$\phi_m = \sum_{r,c} \left(\sum_{m'} \left(K_{m,m'} X^{(j)}_{m',r,c} \right) - \frac{1}{2} \lambda \right) g_{r,c}.$$

In terms of these objects, it is straightforward to show that the relevant portion of the loss can be written as

$$\mathcal{L}(f) = \|g\|^2 \|Kf\|^2 - 2 f^T \phi + \cdots .$$

We would like to minimize this subject to the constraint that $f_m \geq 10^{-10}$. To approach this problem we use a projected gradient descent approach. We start by selecting a search direction, namely the gradient of the Lagrangian:

$$\Delta = \phi - \|g\|^2 K^T K f.$$

We then zero out the coordinates of this search directions which point negatively along the active constraints:

$$\tilde{\Delta}_m = \begin{cases} 0 & \text{if } f_m = 10^{-10} \text{ and } \Delta_m \leq 0 \\ \Delta_m & \text{otherwise.} \end{cases}$$

We then update f by moving it somewhat in this search direction and then forcing it to be positive. How far should we move in the search direction? Following (Kim et al., 2013), we use the following carefully-chosen step-size:

$$f \leftarrow \left[f + \frac{\phi^T \tilde{\Delta} - \|g\|^2 K^T K \tilde{\Delta}}{\left\| K \tilde{\Delta} \right\|^2} \tilde{\Delta} \right]_+ .$$

If we did not force f to be positive, such updates would yield the best possible distance to travel along the search direction. However, due to the positivity-enforcement, one can find pathological examples where applying this update actually makes the loss worse. To be safe, we use a backtracking procedure; as long as the loss is made actively worse by this step, we cut the learning rate in half and try again.

φ update Fix c^*. Let us look at the loss with respect to $\varphi_{c^*,1}, \varphi_{c^*,2} \cdots \varphi_{c^*,C}$. We find that it is given by

$$\mathcal{L}(\varphi_{c^*}) = \sum_{m,r} \left(\texttt{xmab}_{m,r,c^*} - \sum_{c'} \varphi_{c^*,c'} \left(\sum_{j,m'} \alpha_{r,c^*} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \mathbf{Z}_{r,c',j} \right) \right)^2 + \lambda \sum_{c} \varphi_{c^*,c'} \left(\sum_{m,j} \alpha_{r,c} \mathbf{F}_{m,j} \mathbf{Z}_{r,c',j} \right) + \cdots .$$

Define

$$\Gamma_{c_1,c_2} = \sum_{m,r} \left(\sum_{j,m'} \alpha_{r,c^*} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \mathbf{Z}_{r,c_1,j} \right) \left(\sum_{j,m'} \alpha_{r,c^*} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \mathbf{Z}_{r,c_2,j} \right)$$

$$\phi_c = \sum_{m,r} \texttt{xmab}_{m,r,c^*} \left(\sum_{j,m'} \alpha_{r,c^*} \mathbf{K}_{m,m'} \mathbf{F}_{m',j} \mathbf{Z}_{r,c,j} \right) - \frac{1}{2} \lambda \left(\sum_{m,j} \alpha_{r,c*} \mathbf{F}_{m,j} \mathbf{Z}_{r,c,j} \right) .$$

Fixing all other variables, the problem of minimizing the loss with respect to φ_{c^*} can then be understood as a quadratic programming problem.

$$\begin{aligned} \min_{\varphi_{c^*}} \quad & \frac{1}{2} \varphi_{c^*}^T \Gamma \varphi_{c^*} - \varphi_{c^*}^T \phi \\ \text{subject to} \quad & \varphi_{c^*} \geq 10^{-10} \end{aligned}$$

This problem is low dimensional and easy to solve using an off-the-shelf package. We use `scipy.optimize.nnls`.

I.1 Selecting ω

So far, we have assumed that ω used in Equation 1 is a user-provided parameter. This ω represents the the maximum tolerated reconstruction error. There are three methods for choosing this parameter which we can suggest:

Interactively. If the observation model is correct, the predicted values of $\mathbf{X}$ should be 'close' to the observed values. To discern this, our package provides an interactive method for selecting an ω which is satisfactory. This function starts with very large error tolerance (specifically we take ω to be half the maximum observed intensity. The function then allows the user to visually compare the true observations with the predicted values estimated with this value of ω. If the predicted values appear to miss important features of the observed data, the user can then reduce ω. The optimization will be re-run (warm-starting from the old initial condition, so this does not require very much time), and new predicted values are displayed. The function allows the user to continually reduce ω until the user deems that all the important features of the observed data are captured by the predicted values.

Automatically. An automatic method can be achieved by starting with the original data, slightly blur it, and take average squared magnitude of the difference. This magnitude can be used to estimate the amount of speckle noise in the image. We can then choose ω so that the average reconstruction loss at each voxel is less than twice the value of this speckle noise.

Via manually-labeled data. If the user is willing to annotate a portion of their data with their beliefs about which rolonies are located at which positions, this annotated data can be used to select the ω. Specifically, one can select the ω which yields the most accurate rolony detection.

In practice, we find that the interactive method is the most straightforward to use.