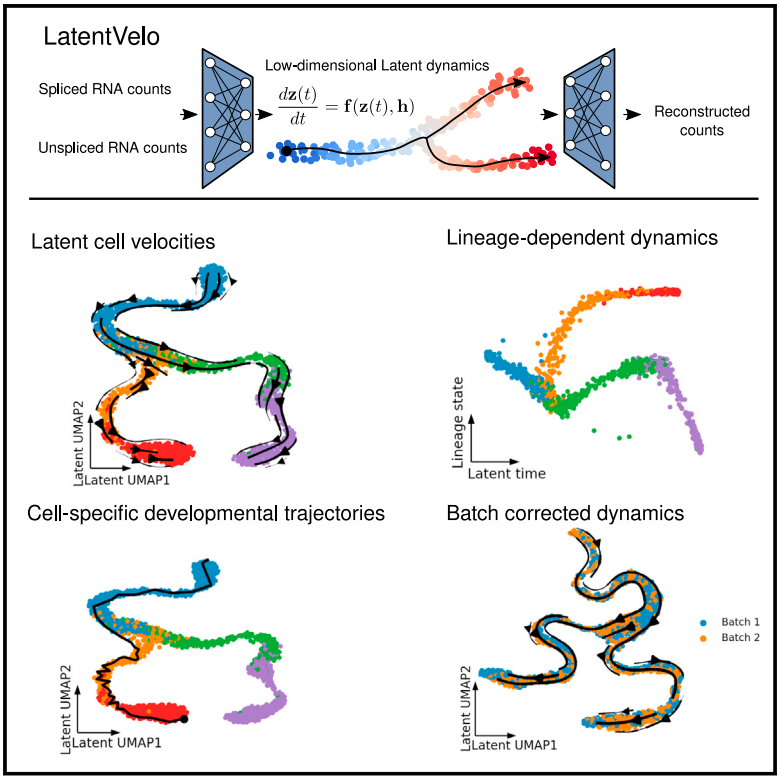
Cell Reports
Methods

Article

Inferring single-cell transcriptomic dynamics with structured latent gene expression dynamics

Graphical abstract



Highlights

- LatentVelo enables more flexible computing of RNA velocity dynamics with deep learning
- LatentVelo enables batch integration of cell states and dynamics by projecting onto a shared latent space
- LatentVelo can infer cell-specific developmental trajectories

Authors

Spencer Farrell, Madhav Mani, Sidhartha Goyal

Correspondence

spencer.farrell@utoronto.ca (S.F.), goyal@physics.utoronto.ca (S.G.)

In brief

Farrell et al. present LatentVelo, an approach to trajectory inference from single-cell RNA sequencing data. LatentVelo infers cell developmental trajectories on a low-dimensional space that is informed by both cell states and dynamics. LatentVelo is demonstrated on a variety of developmental and reprogramming datasets.

Farrell et al., 2023, Cell Reports Methods *3*, 100581
September 25, 2023
https://doi.org/10.1016/j.crmeth.2023.100581

CellPress

Cell Reports Methods

CellPress
OPEN ACCESS

Article

Inferring single-cell transcriptomic dynamics with structured latent gene expression dynamics

Spencer Farrell,[1,6,*] Madhav Mani,[2,3,4] and Sidhartha Goyal[1,5,*]
[1]Department of Physics, University of Toronto, Toronto, ON M5S1A7, Canada
[2]Department of Engineering Sciences and Applied Mathematics, Northwestern University, Evanston, IL 60208, USA
[3]NSF-Simons Center for Quantitative Biology, Northwestern University, Evanston, IL 60208, USA
[4]Department of Molecular Biosciences, Northwestern University, Evanston, IL 60208, USA
[5]Institute of Biomedical Engineering, University of Toronto, Toronto, ON M5S3G9, Canada
[6]Lead contact
*Correspondence: spencer.farrell@utoronto.ca (S.F.), goyal@physics.utoronto.ca (S.G.)
https://doi.org/10.1016/j.crmeth.2023.100581

MOTIVATION RNA velocity has been introduced to study cell differentiation processes. By inferring gene expression dynamics, RNA velocity can provide the local direction of differentiation of single cells. However, these methods are limited by restrictive assumptions. We present LatentVelo, an approach to generalizing the dynamics of RNA velocity with a deep-learning approach.

SUMMARY

Gene expression dynamics provide directional information for trajectory inference from single-cell RNA sequencing data. Traditional approaches compute RNA velocity using strict modeling assumptions about transcription and splicing of RNA. This can fail in scenarios where multiple lineages have distinct gene dynamics or where rates of transcription and splicing are time dependent. We present "LatentVelo," an approach to compute a low-dimensional representation of gene dynamics with deep learning. LatentVelo embeds cells into a latent space with a variational autoencoder and models differentiation dynamics on this "dynamics-based" latent space with neural ordinary differential equations. LatentVelo infers a latent regulatory state that controls the dynamics of an individual cell to model multiple lineages. LatentVelo can predict latent trajectories, describing the inferred developmental path for individual cells rather than just local RNA velocity vectors. The dynamics-based embedding batch corrects cell states and velocities, outperforming comparable autoencoder batch correction methods that do not consider gene expression dynamics.

INTRODUCTION

Single-cell RNA sequencing enables the inference of developmental trajectories with methods that computationally reconstruct the developmental process. Traditional approaches are based on the similarity of static snapshots of RNA for individual cells.[1–3] RNA velocity extends these approaches by modeling the dynamical relationship between newer unspliced RNA and mature spliced RNA to infer the direction of the dynamics from these static snapshots. Recent techniques have also been developed to analyze RNA velocity, aiming to improve and expand upon traditional trajectory reconstruction methods.[4–9]

However, RNA velocity methods have been limited by a reliance on strict modeling assumptions. In the original formulation, Velocyto[10] modeled cells at an assumed steady state, with a linear relationship between unspliced and spliced RNA. scVelo[11] relaxed the steady-state assumption by assuming a strict form for transcription rates and modeled time-dependent transient cell states by fitting a set of linear differential equations, treating the unobserved developmental time as a latent variable inferred per cell and per gene. These methods encounter problems with complex dynamical features such as a transcriptional boost, lineage-dependent kinetics, and weak unspliced signals.[12–14]

We have developed "LatentVelo," an approach to address these problems in RNA velocity estimation. Our key insight and distinguishing feature is that LatentVelo embeds cell states in a learned latent space and infers structured dynamics on this latent space that incorporate the causal structure of RNA velocity dynamics rather than learn RNA velocity on gene space with strict linear dynamics. Since the latent dynamics and embedding are learned together, the latent embedding of cell states is informed by the dynamics. Lineage- or time-dependent dynamics are enabled by modeling state-dependent regulation of transcription with a latent regulatory state. By learning dynamics in a latent space,

LatentVelo can extract the low-dimensional dynamics describing cell differentiation. Additionally, modeling the latent space dynamics enables batch correction and the incorporation of additional information such as annotated cell-type labels as in scANVI[15] or temporal information from sequencing batches. LatentVelo also enables construction ofgeneral dynamical models and enables various structured models of regulation and multi-omics data.

Recently, several other methods have been proposed to address some of these issues to improve the accuracy and reliability of RNA velocity methods. VeloAE[16] develops an extension of the steady-state model by projecting high-dimensional noisy unspliced and spliced vectors onto a lower-dimension latent space with an autoencoder and then uses the steady-state linear model to estimate velocities on this latent space. UniTVelo[17] allows for a more general form of spliced RNA dynamics, which relaxes assumptions on the transcription rate, includes a unified cell-dependent latent time (rather than per gene as in scVelo), and introduces a method of dealing with low signal to noise in unspliced reads. DeepVelo[18] uses a deep neural network to estimate cell-specific kinetic rate parameters to model variable and lineage-dependent kinetics. VeloVAE[19] uses a variational Bayesian approach to model a unified cell-dependent latent time, cell-specific transcription rates, and cell-type-specific splicing and degradation rates. MultiVelo[20] integrates chromatin accessibility into the linear dynamical model to improve velocity estimates. Alternatively, work has been done on gene selection for velocity analysis.[12,21]

LatentVelo has similarities to these methods. LatentVelo embeds cell states in a latent space with an autoencoder like VeloAE but includes dynamics rather than just modeling steady-state cells. LatentVelo models dynamics with a unified cell developmental time like UniTVelo and VeloVAE. Like DeepVelo and VeloVAE, LatentVelo models lineage-dependent dynamics. LatentVelo also includes the scTour model[20] as a special case, where only the dynamics of a single data type are considered (e.g., just spliced RNA).

We benchmark LatentVelo with synthetic data and real developmental, regeneration, and reprogramming data. LatentVelo outperforms the traditional linear RNA velocity methods for inferring transitions between cell types. We also show that our approach is able to do batch correction of RNA velocity by projecting batches onto a common latent space independent of technical variation between the samples. Batch correction is unaddressed by other models of RNA velocity, presenting an application of LatentVelo, and we show that LatentVelo outperforms methods that only attempt to batch correct gene expression and ignore dynamics. Additionally, rather than just outputting RNA velocity vectors, LatentVelo infers developmental trajectories for individual cells, describing the inferred path a specific cell took to reach its current state, and we validate these trajectories with Gene Ontology analyses. This feature is distinct from the traditional velocity methods.

Overview of LatentVelo

RNA velocity

RNA velocity methods estimate the direction of differentiation using static snapshots of unspliced and spliced RNA. The quantity of unspliced and spliced RNA is determined by the processes of transcription, splicing, and degradation of RNA:

$$\emptyset \underset{\text{transcription}}{\rightarrow} \text{unspliced} \underset{\text{splicing}}{\rightarrow} \text{spliced} \underset{\text{degradation}}{\rightarrow} \emptyset. \quad \text{(Equation 1)}$$

The objective is to overcome the limitations of snapshot single-cell RNA sequencing (scRNA-seq) observations by using a model that imposes the causal relationship of splicing, linking the observed quantities of unspliced and spliced RNA. The goal is to estimate the time derivative of the spliced RNA for single cells $\frac{ds}{dt}$ in terms of the amount unspliced and spliced RNA, which characterizes the future direction of differentiation for a cell.

The traditional model for RNA velocity uses linear rate equations for the transcription, splicing, and degradation dynamics per gene g,

$$\frac{du_g(t)}{dt} = \alpha_g(t) - \beta_g u_g(t) \quad \text{(Equation 2)}$$

$$\frac{ds_g(t)}{dt} = \beta_g u_g(t) - \gamma_g s_g(t) \quad \text{(Equation 3)}$$

where s and u are neighbor-averaged spliced and unspliced counts and t is an unknown developmental time that is either eliminated in the steady-state model (Velocyto)[10] or inferred per gene and per cell in the linear dynamical model (scVelo).[11] Instead of assuming the accuracy of these linear models in accounting for the data, we construct a more general deep-learning model that incorporates only the causal relationships outlined in Equation 1 while remaining agnostic to the underlying kinetics of the process. The goal is to encode the causal relationship but generalize the linearity assumption due to the complexity of the underlying kinetics. As such, the approach is more data driven.

LatentVelo

LatentVelo is formulated as a variational autoencoder (VAE)[22,23] that embeds spliced and unspliced RNA count data into latent states $\hat{\mathbf{z}}$ along with a latent developmental time $\hat{t}$ per cell, which provides a time ordering of the cells. If available, we can include batch ID in the input of the encoder and decoder to perform batch correction.

Dynamics on this latent space are described by a neural ordinary differential equation (neural ODE),[24] using the inferred latent time as the time variable in the differential equation. The basic form of the latent dynamics are

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}(\mathbf{z}(t)) \quad \text{(Equation 4)}$$

$$\mathbf{z}(0) = \mathbf{z}_0 \quad \text{(Equation 5)}$$

where the dynamics are determined by a neural network $\mathbf{f}$ and an initial state $\mathbf{z}_0$. Note that the initial state $\mathbf{z}_0$ is learned together with the dynamics and autoencoder, in contrast to pseudotime methods, which specify a root cell. These dynamics are incorporated into the VAE by constraining the encoded state $\hat{\mathbf{z}}$ with the solution to these dynamics at the corresponding latent time for

the cell $\mathbf{z}(\widehat{t})$, e.g., minimizing $|\widehat{\mathbf{z}} - \mathbf{z}(\widehat{t})|$. This encourages the encoder to output latent states $\widehat{\mathbf{z}}$ that follow the latent dynamics and encourages the latent dynamics $\mathbf{z}(t)$ to match the output of the encoder. The result is that LatentVelo learns a latent embedding that is informed by the dynamics.

These dynamics are considered "unstructured" in the sense that $\mathbf{f}$ is a general dense feedforward neural network and that interactions between all components of $\mathbf{z}$ are allowed. A very similar model with these unstructured dynamics was also recently developed in scTour.[20] However, these unstructured models do not utilize biological knowledge of the dynamics of splicing. We add this structure by separately representing the different data modalities in the latent space by decomposing $\mathbf{z} = (\mathbf{z}_u, \mathbf{z}_s)$ and $\mathbf{f} = (\mathbf{f}_u, \mathbf{f}_s)$:

$$\frac{d\mathbf{z}_u(t)}{dt} = \mathbf{f}_u(\mathbf{z}_u(t)) \quad \text{(Equation 6)}$$

$$\frac{d\mathbf{z}_s(t)}{dt} = \mathbf{f}_s(\mathbf{z}_u(t), \mathbf{z}_s(t)). \quad \text{(Equation 7)}$$

The structure comes in by separately modeling spliced $\mathbf{z}_s$ and unspliced $\mathbf{z}_u$ latent representations and restricting the interactions between them to model splicing dynamics. This results in an estimate of a latent representation of RNA velocity, $d\mathbf{z}_s/dt$. This model is a generalized form of traditional RNA velocity methods—modeling the dynamics of transcription, splicing, and degradation with non-linear functions of spliced and unspliced RNA and including interactions between different genes through this low-dimensional latent space.

However, this form of dynamics does not address the issue of multiple lineages. Since the dynamics start from a single initial state $\mathbf{z}_0$ and all cells follow the same ODEs, there is only one unique solution—which means no branching for different lineages. Lineage-dependent dynamics can be incorporated by including regulation of transcription with state-dependent dynamics of a new latent variable controlling regulation $\mathbf{z}_r$:

$$\frac{d\mathbf{z}_u(t)}{dt} = \mathbf{f}_u(\mathbf{z}_u(t), \mathbf{z}_r(t)) \quad \text{(Equation 8)}$$

$$\frac{d\mathbf{z}_s(t)}{dt} = \mathbf{f}_s(\mathbf{z}_u(t), \mathbf{z}_s(t)) \quad \text{(Equation 9)}$$

$$\frac{d\mathbf{z}_r(t)}{dt} = \mathbf{f}_r(\mathbf{z}_s(t), \mathbf{z}_r(t), \mathbf{h}) \quad \text{(Equation 10)}$$

$$\mathbf{h} = \mathbf{f}_h(\widehat{\mathbf{z}}_s, \widehat{\mathbf{z}}_u). \quad \text{(Equation 11)}$$

The variable $\mathbf{h}$ is a constant state-dependent parameter controlling the regulatory dynamics for a particular cell. Note that $\mathbf{h}$ depends on the observed cell state because it depends on the encoded $\widehat{\mathbf{z}}$, not the trajectory $\mathbf{z}(t)$. Since $\mathbf{h}$ is estimated from the observed cell state, it enables dynamics to depend on cell states or lineages. Since the dynamics are deterministic and start from a constant $\mathbf{z}(0)$, without $\mathbf{h}$, we would expect no branching of the dynamics. We can think of $\mathbf{h}$ as conditioning the dynamics to follow a particular branch of the system. This enables us to model complex multi-lineage dynamics. Note that while the VAE includes stochasticity in the latent embedding, there is no stochasticity in the dynamics (e.g., as with stochastic differential equations). We address this in the discussion as a future direction. Since all cells have the same initial state $\mathbf{z}_s(0), \mathbf{z}_u(0)$, and $\mathbf{z}_r(0)$ and $\mathbf{f}_u$ depends on $\mathbf{z}_r$ instead of $\mathbf{h}$, all cells start with similar velocities and branch as latent time increases.

Figure 1 demonstrates some of the features of LatentVelo on a synthetic dataset (generated with dyngen[25]). The structure of LatentVelo described in Equations 8, 9, 10, and 11 is visualized in Figure 1A. The graph showing the interactions of $\mathbf{z}_u$, $\mathbf{z}_s$, and $\mathbf{z}_r$ summarizes the model equations, where arrows indicate interactions between the variables (including self interactions). LatentVelo infers low-dimensional latent representations of RNA velocity, which can be projected onto the uniform manifold approximation and projection (UMAP), and latent times similar to a pseudotime, which correctly correspond to the differentiation direction (Figure 1C). LatentVelo also infers a latent embedding of cell states informed by the transcriptomic dynamics (Figure 1D). LatentVelo's latent regulatory state z_r can highlight branching lineages (Figure 1E).

Current RNA velocity methods estimate the spliced velocity of each observed cell. LatentVelo infers cell-specific trajectories on the latent space instead of simply inferring local cell velocities (Figure 1F). Typically, RNA velocity has been interpreted by visualizing global velocity fields of smoothed local cell velocities projected onto a 2-dimensional representation of the data (e.g., UMAP or t-distributed stochastic neighbor embedding [tSNE]). LatentVelo's inferred trajectories offer a new way of interpreting RNA velocity without resorting to the 2-dimensional representation of velocities. These trajectories directly indicate the path of differentiation, which more clearly indicates transitional cell types.

LatentVelo can perform batch correction of dynamics within the latent space (Figure 1G). This is done by including the batch ID with the encoder and decoder (seen in Figure 1A), resulting in a batch-independent latent space and latent dynamics.

We can add more structure to the latent embedding by including cell-type information, similar to scANVI.[15] This approach modifies the prior of the latent space to be cell-type specific. This modification prevents the prior from ignoring biologically relevant clusters, and in particular, we show below that this improves batch correction. The full details of this modification are in the STAR Methods section incorporating cell-type annotations.

Since LatentVelo has many free parameters, it can be useful to include the experimental time of observation of cells to further constrain the dynamics. We add the correlation between the inferred latent time and experimental time to the loss function. Additionally, we can specify a "root cell" (or terminal cells) by penalizing the latent time at $\widehat{t} = 0$ or 1 to correspond to particular cell types. The details of this are in the STAR Methods section incorporating experimental time points or root cells.

While the main focus of LatentVelo is to infer latent dynamics, we can also infer the dynamics of single genes. The velocity of single genes is computed by transforming the latent space velocities into gene-space velocities with the decoder $\dot{\mathbf{s}} = J[D_s](\mathbf{z}_s)\mathbf{f}_s(\mathbf{z}_s, \mathbf{z}_u)$, where $J[D_s](\mathbf{z}_s)$ is the Jacobian of the spliced

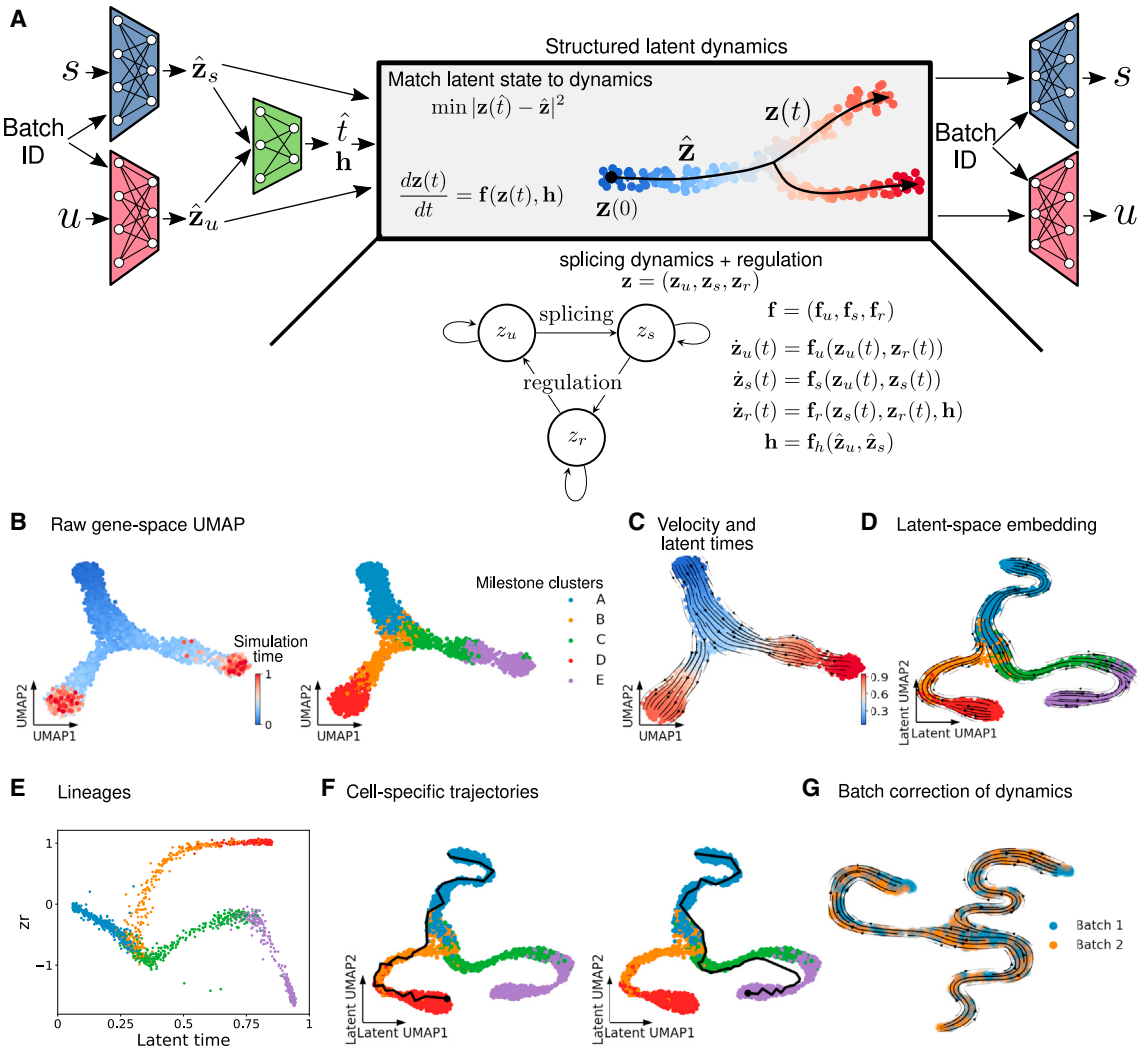


Figure 1. LatentVelo

(A) LatentVelo is a variational autoencoder with structured dynamics on the latent space. An encoder (left trapezoids) encodes the transcriptomic cell state (i.e., unspliced [*u*] and spliced [*s*] RNA counts) into corresponding latent states $\hat{z}_s$ and $\hat{z}_u$ and a latent developmental time *t* and regulation state **h**. The regulation state **h** conditions dynamics for each cell to follow a particular branch. We match these latent states to the dynamics obtained by structured dynamics on this latent space, $\mathbf{z}(t)$, described by ODEs starting from an initial state $\mathbf{z}(0)$. These structured dynamics can take a variety of forms; here, we show regulated splicing dynamics incorporating state-dependent regulation of transcription. In the shown notation, we use $\hat{\mathbf{z}}$ to denote latent cell states estimated directly from the encoder with the input data and $\mathbf{z}(t)$ to represent latent cell states estimated from the latent dynamics. When fitting the model, the distance between the encoder latent states and the dynamics latent states is minimized. For datasets with multiple batches, LatentVelo can integrate multiple batches by including the batch ID into the encoder and decoder.

(B–F) We demonstrate LatentVelo on an example synthetic dataset.

(B) Gene-space UMAP for a synthetic dataset generated with dyngen.[25] The developmental trajectory is shown by increasing simulation time and cell-type transitions with the milestone clusters.

(C) LatentVelo latent velocities projected onto the UMAP and latent times.

(D) UMAP plot of LatentVelo latent embedding.

(E) Latent regulatory state $\mathbf{z}_r$ infers branching.

(F) Inferred cell-specific trajectories for two cells on different branches of the bifurcation.

(G) Batch correction with LatentVelo.

decoder evaluated at $\mathbf{z}_s$ and $\mathbf{f}_s$ is the latent space velocity. Since the high-dimensional gene space is compressed into a low-dimension latent space, we expect the dynamics of only the best reconstructed genes to have their dynamics be well modeled.

RESULTS

LatentVelo infers cell fate trajectories

In Figure 2, we demonstrate LatentVelo on a pancreatic endocrinogenesis dataset showing differentiation of endocrine

CellPress
OPEN ACCESS

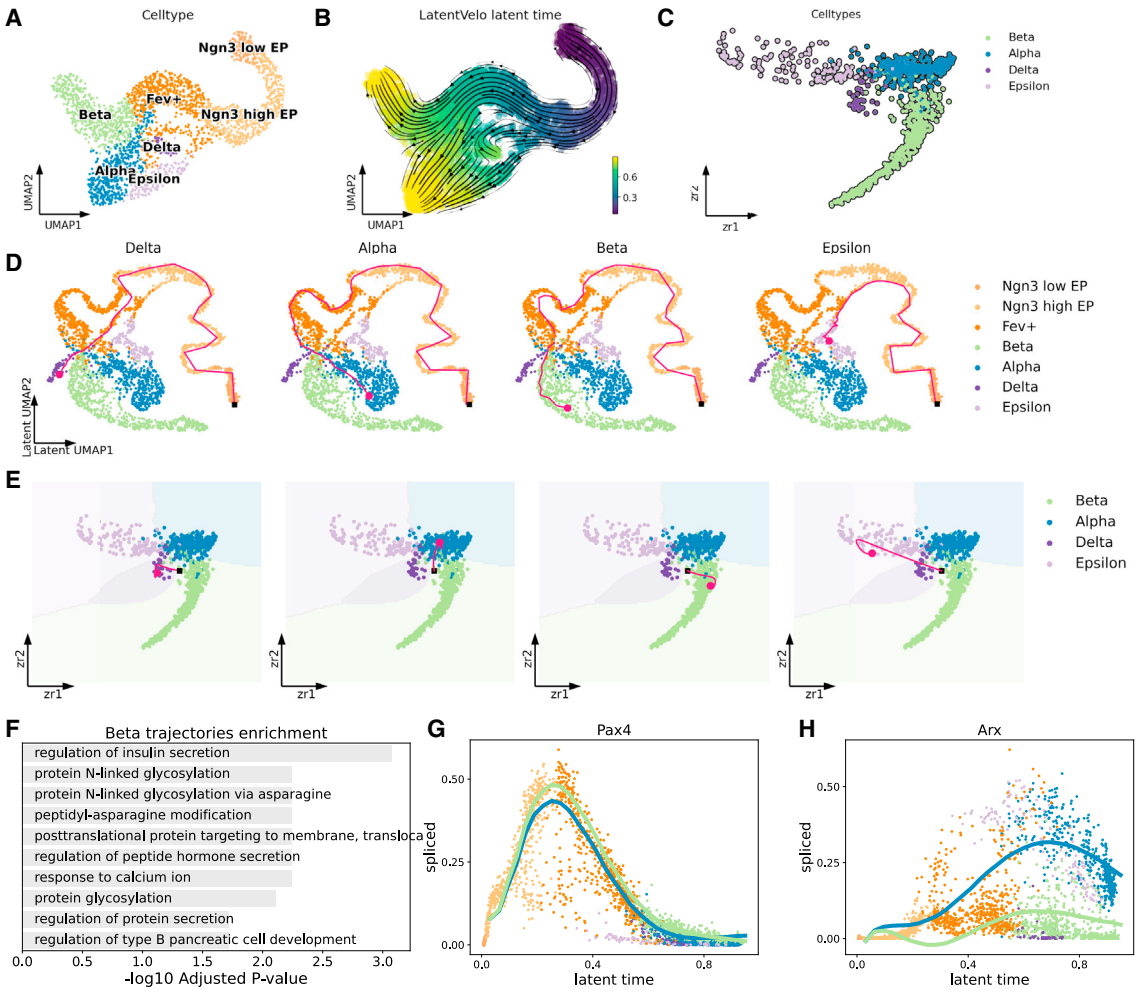


Figure 2. LatentVelo infers latent cell dynamics

(A) UMAP plot showing the differentiation of endocrine progenitors into 4 terminal states: alpha, beta, delta, and epsilon cells.

(B) Velocity and latent time inferred by LatentVelo, indicating the direction of differentiation.

(C) Inferred latent regulatory states in LatentVelo, highlighting the distinct dynamics for the terminal states.

(D) Latent trajectories inferred by LatentVelo for 4 different cells, corresponding to a delta, epsilon, beta, or alpha cell. Plot is shown for a UMAP representation of the spliced latent state. The black square represents the inferred initial state $\mathbf{z}_0$.

(E) Scatterplot of the terminal states for z_{r1} vs. z_{r2}. Dynamics start from the initial state $\mathbf{z}_0$ at the black square (the learned initial state) and evolve with the estimated h and follow the magenta path until terminating at the magenta circles at the estimated $\hat{t}$.

(F) Gene Ontology analysis on the beta cell trajectories.

(G) Alpha and beta cell trajectories for Pax4.

(H) Alpha and beta cell trajectories for Arx.

progenitors into 4 terminal states: alpha, beta, delta, and epsilon cells.[26] The latent time and velocities estimated by LatentVelo show the differentiation of progenitors into the terminal cell states (Figure 2B). We visualize the 2-dimensional latent regulatory state $\mathbf{z}_r$, which controls the dynamics of each cell (Figure 2C). The 4 terminal cell types cluster into separate regions, showing that LatentVelo learns distinct dynamics for these 4 terminal states. In contrast, traditional RNA velocity methods learn the same parameters for all cell types.

We choose 4 cells, each belonging to one of the 4 terminal states, and plot the inferred dynamics for these cells (Figure 2D). The dynamics start from the initial cell state $\mathbf{z}_0$ at time 0, which is learned together with the dynamics and autoencoder. In each of these trajectories, the encoder of the model infers the conditioning variable $\mathbf{h}$, which determines the dynamics of $\mathbf{z}_r$, which regulates the dynamics of a given cell to fall along a particular lineage. Then, starting from the initial state $\mathbf{z}_0$ and conditioning the dynamics on $\mathbf{h}$, we solve the ODE for each of the trajectories to the final time $\hat{t}$. These trajectories more clearly delineate the differentiation of the 4 terminal cell types in comparison to 2b. In particular, the development of epsilon cells is separated and is more cleanly seen as a separate terminal state.

The dynamics of the latent regulatory state $\mathbf{z}_r$ demonstrate the branching of the dynamics (Figure 2E). $\mathbf{z}_r$ evolves from the initial

state (black square) to the respective regions representing the 4 terminal cell types.

To confirm that these inferred trajectories recover the known biology, we conduct a Gene Ontology analysis of differentially expressed genes along the inferred beta cell trajectories, finding that the processes of regulation of insulin secretion and of regulation of beta cell development are enriched (Figure 2F). Additionally, we plot the inferred trajectories for the spliced expression of the Pax4 (Figure 2G) and Arx (Figure 2H) genes, showing that beta cells follow a trajectory with higher Pax4, while alpha cells follow a trajectory with lower Arx, in the progenitor cells. Note that only the terminal state of these trajectories are fit to the data ($\mathbf{z}(\hat{t})$)—the trajectory is a prediction of the model. This is an expected result due to the opposing actions of Pax4 and Arx during alpha and beta cell bifurcation,[27] which has also been found in other RNA velocity analyses.[6]

LatentVelo is similar yet distinct in this aspect of inferring trajectories to recent approaches that utilize RNA velocity dynamics in further analyses. CellRank[4] uses a cell-to-cell transition matrix, which can be constructed with RNA velocity, to extract initial states, terminal states, and absorption probabilities of cells transitioning to a particular terminal state. The difference with LatentVelo is that we infer the past trajectory rather than the probability of a future particular fate. Dynamo[5] uses RNA velocity estimates to interpolate a smooth velocity field, from which various differential geometric quantities can be extracted, and the velocity field can be used to simulate trajectories for a specific initial starting state.

Given velocity estimates from any RNA velocity method and a sufficiently accurate approximation of the vector field, Dynamo is able to produce similar trajectories as inferred by LatentVelo. We use Dynamo with the LatentVelo latent space and velocities in Figure S1. We find that the approximate velocity field has trouble generating trajectories by solving the ODE backward from terminal cells but that the least-action paths generated with Dynamo generally agree with LatentVelo.

Inferring these trajectories is more difficult than simply estimating velocity direction. In Figure S1, we show on an intestinal organoid dataset[28] that estimated velocities can show expected cell-type transitions (visually on UMAP and quantitatively with cross-boundary directedness [CBDir] score), while inferred trajectories are poor—the final state of the trajectory $\mathbf{z}(\hat{t})$ does not match the directly encoded state from the data $\hat{\mathbf{z}}$. We also show that this can be improved by increasing the dimensions of the latent space to include more information. This procedure can generally be used to determine the required dimension of the latent space—we can increase the dimension until trajectories terminate close to the directly encoded state from the encoder.

LatentVelo infers early separation of trajectories for reprogrammed and dead-end cells

To demonstrate LatentVelo's ability to learn biologically meaningful features in its latent representation, we use a dataset of mouse embryonic fibroblasts (MEFs) reprogramming toward induced endoderm progenitors (iEPs)[29] (Figure 3). In this experiment, overexpression of key transcription factors drives fibroblasts to potentially undergo reprogramming or to develop into a dead-end state. Cell tagging is used to identify longitudinal trajectories of reprogramming. On this dataset, scVelo incorrectly shows differentiation from reprogrammed to dead-end cells and shows no route to reprogramming[4] (Figure 3B).

In contrast to scVelo, the velocity estimated by LatentVelo correctly shows no differentiation from reprogrammed to dead-end cells and indicates transitions to both dead-end and reprogrammed cells (Figure 3C), and the estimated latent time aligns with the experimental time course (Figures 3A and 3C).

Observing the regulator parameter z_r, we see the LatentVelo infers two distinct regimes of dynamics: negative z_r, where cells undergo reprogramming, and positive z_r of dead-end cells that do not (Figures 3D and 3E). This highlights that the regulation parameter z_r can describe different "modes" of the dynamics and here identifies the differences in reprogrammed vs. dead-end cells.

Reprogramming takes a complex path through the data, and the reprogramming trajectory is not clearly distinguished from the dead-end trajectory in the tSNE plot. However, in contrast, it becomes clearly apparent in the UMAP plot of the spliced latent space inferred by LatentVelo shown (Figures 3F–3I). Here, we see the latent space cleanly separates the reprogrammed/dead-end cells, generating a latent representation informative of the reprogramming dynamics. This shows that the latent representation of the dynamics inferred by LatentVelo is biologically meaningful. In Figure S2, we also show in the pancreas, retina, bone marrow, hematopoiesis, intestinal organoid, and hindbrain datasets that, in general, the regulatory parameter $\mathbf{z}_r$ clearly identifies lineages in this same way, highlighting lineage-dependent dynamics that are not modeled in some other RNA velocity approaches like scVelo.

The z_r vs. experimental time and latent time shows a clear separation for reprogrammed/dead-end cells (Figure 3J). We also train a logistic regression classifier on z_r (effectively just scaling z_r to represent a probability) and show a receiver operating characteristic (ROC) curve (Figure 3K), demonstrating that it is predictive of reprogrammed cells. This shows that the single parameter z_r inferred by the model corresponds to a meaningful biological feature: the reprogramming outcome.

This analysis with LatentVelo suggests that the reprogramming fate decision occurs early on in the process, as early as 6 or 9 days (seen at 6 days, but there are very few cells tagged as reprogrammed at 6 days). This was also suggested in the original analysis by Biddy et al.[29] Since LatentVelo is able to infer these early reprogrammed fates without the cell tagging information, this suggests that the relevant information for reprogramming is available within the scRNA-seq data at early times and can be extracted with an analysis of the RNA velocity dynamics with LatentVelo.

We sample 25 trajectories for dead-end and reprogrammed cells with latent time $\hat{t}$ (Figure 3M). These trajectories diverge toward reprogrammed and dead-end states, with no significant transitions from reprogrammed to dead-end cells as there are in scVelo. To perform a similar analysis with scVelo, we use CellRank[4] to infer the terminal states from the estimated velocities of scVelo. We use the velocity kernel, which computes a transition matrix based on the estimated velocities. Using the scVelo velocities, only a single terminal state is obtained in

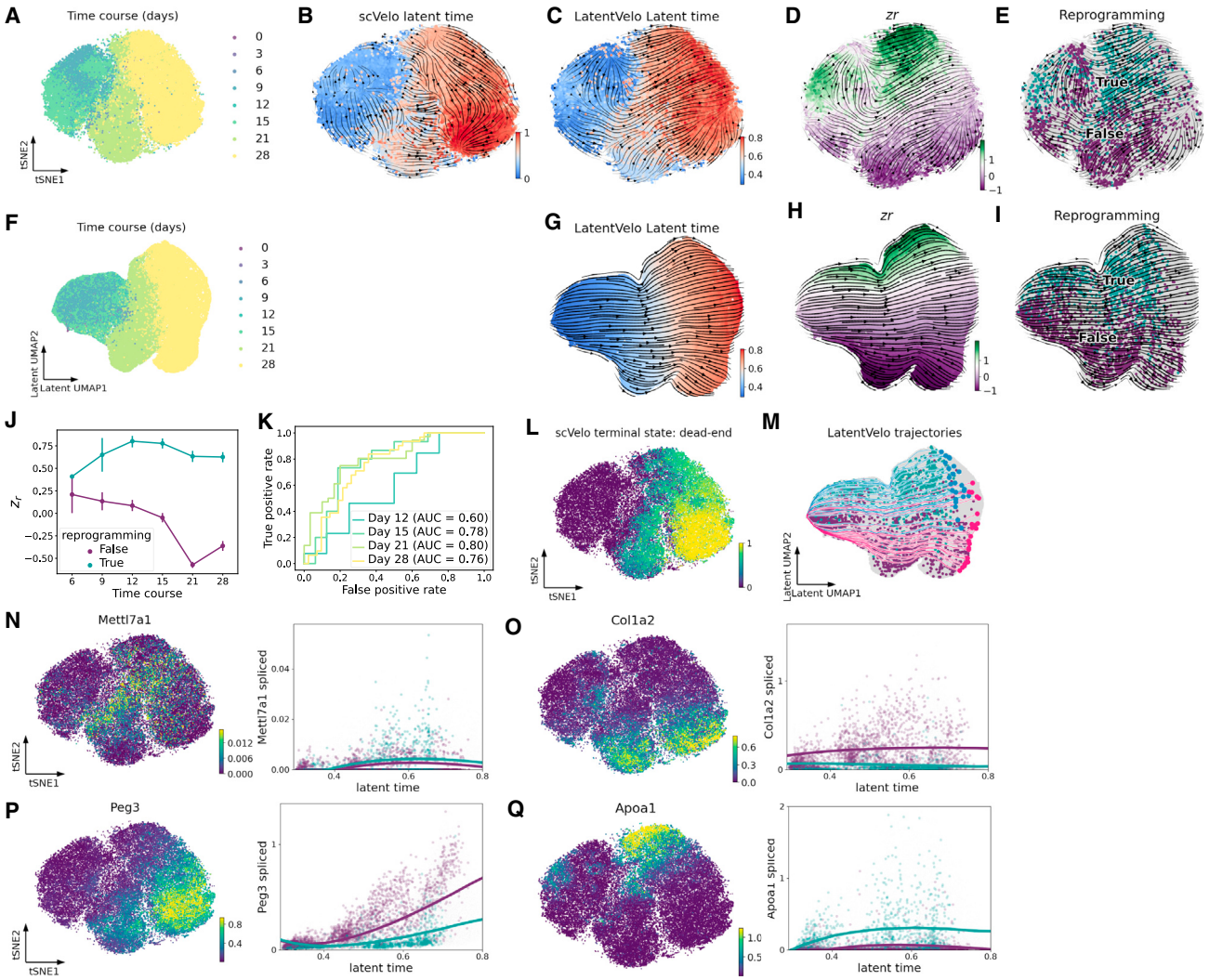


Figure 3. Reprogramming dynamics of mouse fibroblasts

(A–E) tSNE representation of mouse embryonic reprogramming data.[29]

(A) tSNE plot colored by experiment time course.

(B) scVelo velocity and latent time.

(C) LatentVelo velocity and latent time.

(D) LatentVelo regulatory parameter z_r.

(E) Reprogramming outcome (turquoise: reprogrammed, purple: dead end) from the experiment.

(F–I) UMAP representation of the latent spliced latent state $\mathbf{z}_s$, shown with (F) the experimental time course, (G) LatentVelo velocity and latent time, (H) regulatory parameter z_r, and (I) reprogramming outcome.

(J) Mean z_r vs. experimental time point. Errorbars are bootstrapped 95% confidence intervals.

(K) ROC curve from a 1-dimensional logistic regression showing z_r is predictive of reprogramming outcome.

(L) Original tSNE embedding colored by the predicted fate probabilities by CellRank[4] using the scVelo velocities with CellRank's velocity kernel. scVelo only infers a terminate state at the dead-end cells.

(M) Example latent trajectories of the dynamics inferred by LatentVelo are shown on the UMAP representation of the spliced latent state, showing the divergence of dead-end and reprogrammed cell trajectories. 25 dead-end and 25 reprogramming trajectories with $\hat{t} > 0.6$ are randomly sampled.

(N–Q) Mean trajectories for 4 differentially expressed genes.

Figure 3L, corresponding to dead-end cells. This is due to scVelo incorrectly inferring transitions from reprogrammed cells to dead-end cells.[4] CellRank offers a method of improving this estimate by combining the velocity kernel with other kernels, but here, we are only interested in the velocity estimates of scVelo.

We show that these trajectories reproduce the expected results in 4 genes that are up-regulated for either reprogramming or dead-end cells (Figure 3N–3Q). While the model only fits the terminal state of the trajectories to the data ($\mathbf{z}(\hat{t})$), this demonstrates that the entire inferred trajectories agree with the expected result.

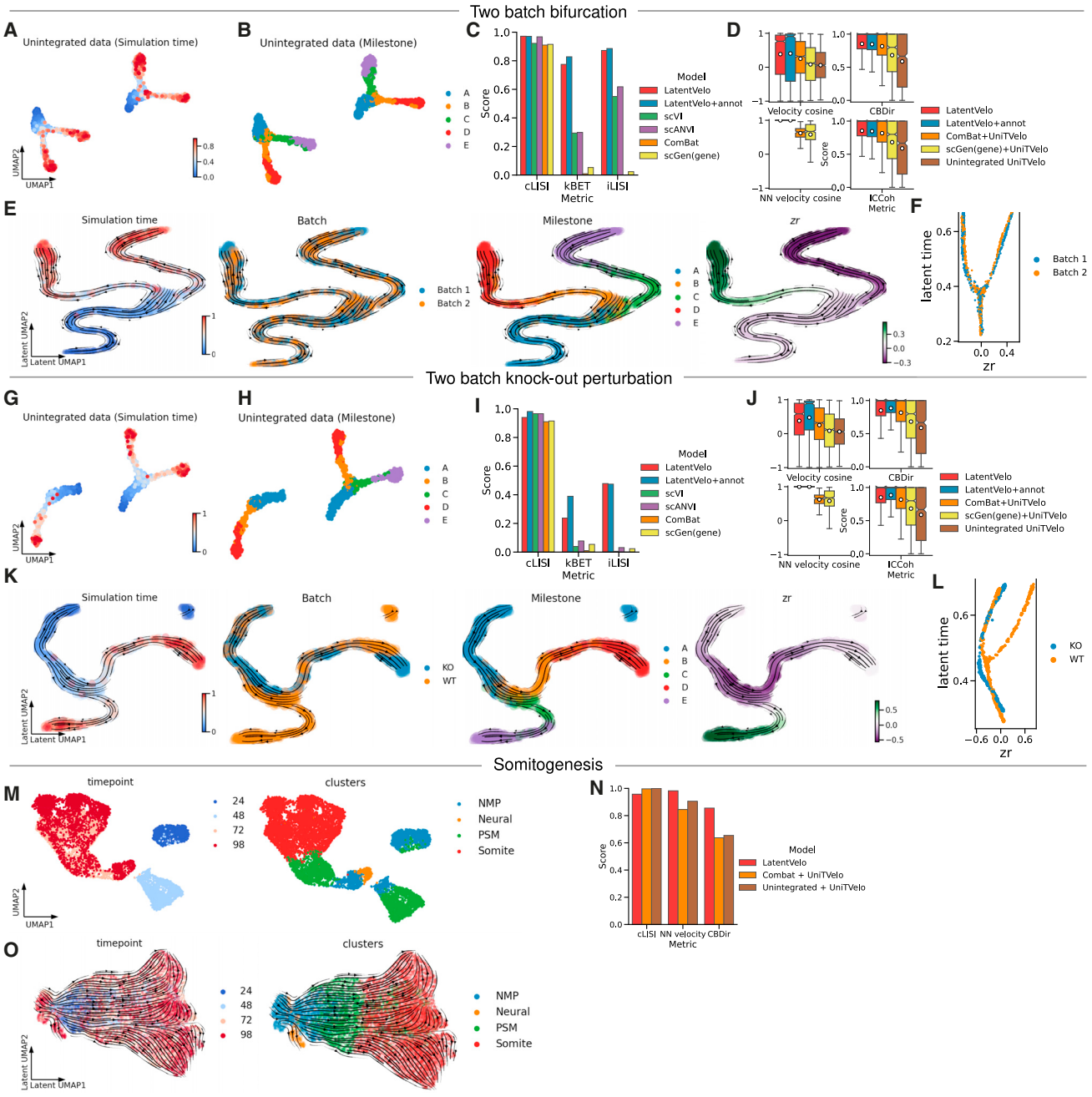


Figure 4. Batch effect correction of dynamics with LatentVelo

(A) Unintegrated data of a two batch bifurcation, simulated with dyngen. Simulation time shows the direction of differentiation.

(B) Milestones define similar clusters of cells between batches and directions tested with CBDir scores.

(C) Batch correction (kBET and iLISI) and biological cluster conservation (cLISI) scores for batch correction of cell states (not velocity).

(D) Velocity scores for the models. Nearest-neighbor (NN) velocity cosine similarity scores measure batch correction of velocity, and the velocity cosine similarity and CBDir scores measure velocity accuracy.

(E) Latent space UMAP with cell-type annotations (LatentVelo+annot) showing simulation time, batch ID, milestone annotations, and latent regulatory state z_r of spliced latent states at $\hat{t}$.

(F) z_r vs. latent time for the two batches, showing batch independence of the regulatory variable.

(G) Unintegrated data simulated with dyngen. Gene-module knockout has been performed for one of the batches (labeled WT [wild type] and KO [knockout]).

(H) Milestone cells show that the C (green) and E (purple) lineages are removed by the knockout.

(I) Batch correction metrics for cell states.

(J) Velocity metrics, showing scores for batch correction and accuracy of velocity.

(legend continued on next page)

Latent space dynamics correct for batch effects in RNA velocity and cell states

Batch correction for RNA velocity is unaddressed by current approaches.[13] Since typical RNA velocity methods operate on gene space, high-performing batch correction methods[30] that typically output a latent embedding or corrected nearest-neighbor graph will not work with typical RNA velocity methods. Another challenge of correcting batch effects in RNA velocity is the need to simultaneously correct unspliced and spliced RNA counts. In this section, we show that LatentVelo can perform batch correction of states and dynamics.

Since LatentVelo learns dynamics on a latent space, it naturally enables batch correction by learning a shared latent space for the multiple batches. This works similarly to existing VAE batch correction methods, such as scVI and scANVI.[15,31] By including batch IDs in the input of the encoder and decoder and using batch-independent latent dynamics, all batch variation can be handled by the encoder and the decoder.

To demonstrate LatentVelo's ability to learn dynamics in a latent space that are independent of batch effects, we use two different simulated datasets from dyngen[25] with strong batch effects (Figure 4). First, we use a simulated dataset with two batches of a bifurcation. The two batches are seen in a UMAP plot of the unintegrated data (Figures 4A and 4B), and the shown trajectory milestones are used as comparable clusters of cells between the batches for benchmarking. We evaluate biological cluster conservation (cell-type local inverse Simpson's Index [cLISI]) and batch correction (k-nearest-neighbor batch effect test [kBET], integration local inverse Simpson's index [iLISI])[30] of the latent cell states (Figure 4C). Our model is the only method that scores highly on all metrics, even when compared with these methods specifically designed for batch correction of gene expression states.

To evaluate velocity (Figure 4D), we use the CBDir score, which measures velocity directions on the boundary between cell types, and the inner-cluster coherence (ICCoh) score, which measures the coherence of neighboring velocities (STAR Methods section LatentVelo), both of which were used by UniTVelo and VeloAE.[16,17] Batch correction of velocity is evaluated by comparing the cosine similarity of velocities for neighboring cells in different batches (nearest-neighbor [NN] velocity cosine similarity). We compare LatentVelo with UniTVelo used on the unintegrated data or UniTVelo used with a previously suggested method for batch correction of RNA velocity[32,33] (STAR Methods section ComBat and scGen RNA velocity batch effect correction). This approach requires the use of any generic batch correction method that outputs corrected gene expression matrices, and so we use this method with ComBat[34] (as was originally suggested) and scGen[35] (an auto encoder method). We found that each of these methods fail to properly integrate the batches and have worse performance on velocity metrics than LatentVelo. Note, we use UniTVelo due to the poor performance of scVelo on the synthetic datasets (see Figure 6).

LatentVelo's latent space accurately integrates the two batches and velocities following the direction of differentiation (Figure 4E). Both batches are visually integrated, in addition to the quantitative metrics (Figures 4C and 4D). The regulatory parameter $z_r(\widehat{t})$ shows that LatentVelo accurately describes the lineage dynamics with distinct values of z_r on each lineage, despite the two batches. Adding cell-type annotations to the latent space improves batch correction (LatentVelo+annot).

We demonstrate our approach on two simulated batches of a bifurcation but now perform a gene-module knockout on one of the batches, eliminating the C (green) and E (purple) milestone cells in this batch (Figures 4G and 4H). Quantitative metrics show that LatentVelo is able to integrate the two batches (Figures 4I and 4J). The UMAP representation of the spliced latent states visually shows the integration of the batches in the joint latent space (Figure 4K). LatentVelo cleanly disentangles the batch effects from the perturbation.

LatentVelo is the only model consistently performing strongly for both velocity estimation and batch correction of cell states and velocities for these simulated datasets. Additionally, our model performs even better at batch correction of gene expression than the batch correction methods scVI, scANVI, ComBat, and scGen(gene). This is visualized in Figure S3. LatentVelo is able to achieve this better performance for batch correction by utilizing the splicing dynamics to not just position similar cells together but to also align them in time according to the dynamics. All model hyperparameters remain at their defaults for these batch correction tests. We use default parameters for scVI, scANVI, scGen, ComBat, and UniTVelo.

In experimental data, batch correction is often made more difficult with batches from different experimental times. To demonstrate LatentVelo in this scenario, we use a segmentoid dataset,[36] an organoid model of somitogenesis. No clear trajectory is seen with the time points, and cell types are not clustered between time points (Figure 4M). In the original analysis, RNA velocity was inferred with scVelo on each time point separately. With LatentVelo, we infer velocity with all time points combined, to integrate the time points.

In the LatentVelo latent space UMAP, cell types are clustered, and velocities follow the expected transitions (Figure 4O). Time points are not fully mixed, as they are not developmentally equivalent, and LatentVelo orders the cells according to latent developmental time. The result is a clear trajectory with the cell-type transitions. We quantitatively evaluate the cell-type transitions with CBDir scores and the batch correction of velocity (Figure 4N), showing that LatentVelo performs better than the comparison methods of UniTVelo using Combat for integration or UniTVelo on the original unintegrated data.

(K) UMAP representation of LatentVelo+annot spliced latent states.
(L) z_r vs. latent time for the two batches, showing batch independence of the regulatory variable.
(M) Original UMAP embedding of a segmentoid, showing 4 experimental time points and cell-type clusters.
(N) Integration and velocity metrics after integration with LatentVelo or comparison methods.
(O) LatentVelo latent space UMAP plots with time points and cell clusters.

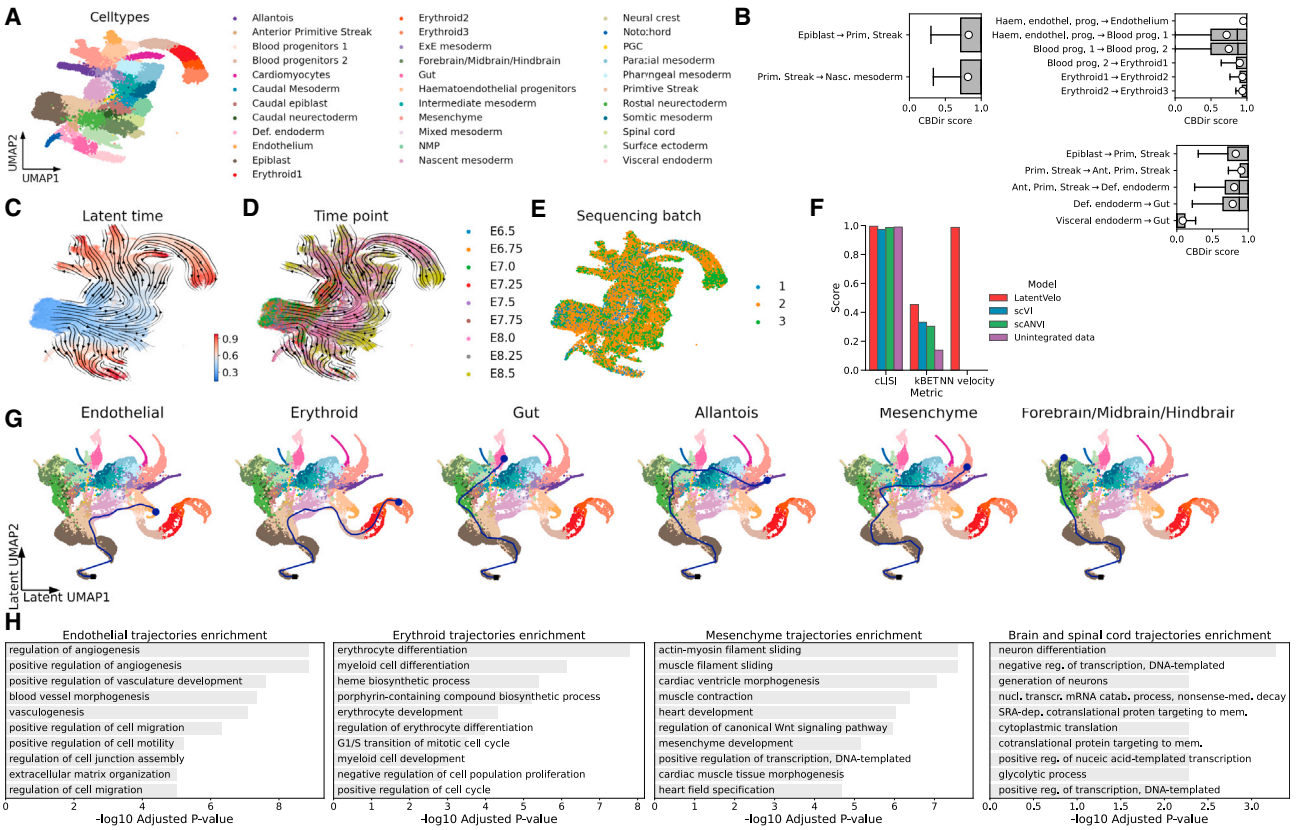


Figure 5. Inferring multiple lineages in mouse gastrulation

(A) UMAP plot showing cell types during mouse gastrulation.

(B) Some CBDir cell-type transition scores estimated on the inferred latent space, highlighting the inferred transitions.

(C and D) LatentVelo latent time (C) and experimental time points (D) with inferred LatentVelo velocity.

(E and F) Sequencing batch of the experiment (E) and batch correction scores (F).

(F) Integration of sequencing batches.

(G) Inferred latent trajectories for examples of endothelial, erythroid, gut, allantois, mesenchyme, and forebrain/midbrain/hindbrain cells.

(H) Gene Ontology analysis of differentially expressed genes along the inferred trajectories. We show the top 10 terms.

LatentVelo infers cell fate trajectories in large multi-lineage systems

In the pancreas dataset in Figure 2 and the fibroblast reprogramming dataset in Figures 3 and S2, we demonstrated that LatentVelo can learn dynamics in datasets with multiple lineages. However, these datasets had at most 3–4 lineages. We now test LatentVelo on a large-scale many-lineage system.

We use a mouse gastrulation dataset (Figure 5), showing development from pluripotent epiblast cells into the ectodermal, mesodermal, and endodermal progenitors of major organs.[37] Note, this is the full dataset from where the erythroid cells from Figure S9 come from (seen in the top right of the UMAP plot in Figure 5A with the same colors). On this large scale dataset, we have increased the dimension of the latent $\mathbf{z}_s$ and $\mathbf{z}_u$ states to 70 and used a 6-dimensional $\mathbf{z}_r$.

Putative developmental directions can be seen by observing the later time point cells (embryonic day 8.5 [E8.5]; Figure 5D), and our model's velocity and latent time estimates generally agree with these time points (Figure 5C). We also highlight some CBDir transition scores between cell types, showing the model infers transitions from epiblast to mesoderm, the development of epithelial and erythroid cells, and the development of the gut from endoderm cells (Figure 5B). One particular transition that we do not see is the convergence of visceral endoderm cells and definitive endoderm cells to gut cells that was originally identified[37]; we only see the transition from definitive endoderm cells to gut cells. We suspect this is due to LatentVelo assuming a single initial state $\mathbf{z}_0$.

This dataset consists of 3 sequencing batches. We use these to evaluate LatentVelo's ability to do batch correction (Figures 5E and 5F). LatentVelo outperforms scVI, scANVI, and the original unintegrated data at mixing together the batches (kBET score), and velocities are highly aligned between the batches (NN velocity cosine similarity).

We infer trajectories for a sample of endothelial, erythroid, gut, allantois, mesenchyme, and forebrain/midbrain/hindbrain cells (Figure 5G). These trajectories give a more clear visualization of the dynamics as compared to the velocity field on the gene-space UMAP plot, where disentangling the many occurring transitions is difficult. Inferred trajectories start as epiblast cells and

transition to the primitive streak, then branch to the variety of cell types.

To validate, we compute differentially expressed gene along these inferred trajectories and perform a Gene Ontology (GO) analysis (Figure 5H). Along the endothelial trajectories, we find GO terms relating to vasculature development, including angiogenesis and vasculogenesis, as expected. Along the erythroid trajectories, we find erythrocyte and myeloid cell development terms enriched, as expected, and heme and porphyrin biosynthesis, which are critical components of erythroid function. Along the mesenchyme trajectories, we see a variety of terms relating to muscle function and development, including canonical Wnt signaling, which is critical for the differentiation of mesenchymal stem cells,[38] as expected. Along the brain and spinal cord trajectories (specifically, the forebrain, midbrain, hindbrain, and spinal cord), we see terms relating to neuron differentiation and generation, as expected. Overall, the trajectories inferred by LatentVelo involve genes that are consistent with known biology.

This dataset demonstrates LatentVelo's ability to learn dynamics on large multi-lineage systems with many different cell types.

Benchmarking transitions on a variety of synthetic and real datasets

In the above sections, we performed a detailed analysis on a number of datasets. We now benchmark LatentVelo on a variety of both synthetic and real datasets to demonstrate robustness.

We primarily compare our model with the two modes of scVelo: dynamical and stochastic.[11] Other recent RNA velocity methods have been developed, such as UniTVelo,[19] DeepVelo,[18] VeloVAE,[19] and VeloVI.[39] We do not compare with these methods due to their recency and the variety of hyperparameters needed to be adjusted to fairly compare the methods.

We use dyngen[25] to generate synthetic datasets of 5,000 cells, each with a variety of developmental structures: linear (51 genes), bifurcation (65 genes), trifurcation (81 genes), and a binary tree (89 genes). We use the cosine similarity between ground-truth and estimated velocities to compare the models. LatentVelo performs much better than scVelo dynamical and stochastic modes (Figures 6A–6C) on these synthetic datasets. We also test our model on increasing/decreasing transcription, splicing, and degradation rates vs. time with simulations from the scVelo linear differential equations in Figure S6.

We evaluate 10 real datasets with known trajectory directions and compare them with scVelo stochastic and dynamical modes (Figures 6D and 6E). We evaluate these velocities with our modified CBDir score (STAR Methods section LatentVelo). Our model consistently scores well for the CBDir scores, above scVelo. In particular, LatentVelo is able to model feature such as a transcriptional boost, which has presented problems for LatentVelo (Figure S2; bone marrow and erythroid gastrulation datasets).[12,13] Overall, lower valued scores (but above scVelo) are seen in scNT, mouse hematopoiesis, and hindbrain (GABA, glial) datasets, where velocity follows the expected directions but noisy cell-type boundaries lower the CBDir scores. CBDir relies on accurate discrete annotations of cell types, which can cause issues on the boundaries between cell types. Additionally, this metric does not look at the strength of transitions, only the direction of velocity.

We use the ICCoh score to evaluate velocity coherence. At a maximum value of 1, neighboring cells have the same velocity direction. LatentVelo achieves high values for all datasets for this metric.

We evaluate LatentVelo's ability to model the dynamics of separate lineages with the regulatory parameter $\mathbf{z}_r$. We train a simple logistic regression classifier on $\mathbf{z}_r$ to predict separate lineages, obtaining classification accuracy between 0.9 and 1 for all datasets (shown in Figure S2). This shows that the lineages are clearly separated and modeled with distinct dynamics.

For these synthetic and real datasets, we keep all model hyperparameters at their default values, except the dimension of $\mathbf{z}_r$ and the dimension of $\mathbf{h}$. The dimension of $\mathbf{z}_r$ is by default set based on the number of expected lineages, and we set $\mathbf{z}_r$ to be the number of expected lineages minus one, except if we only expect 1 lineage where we use 1 dimension. We have found this to be a good heuristic. The dimension of $\mathbf{h}$ is set to be the same as $\mathbf{z}_r$, unless there is poor agreement between $\mathbf{z}(\widehat{t})$ and $\widehat{\mathbf{z}}$, and then it is increased (alternatively, the dimension of the latent state can be increased; see Figure S4). Regularization strength is set to the default value 0.1 for all synthetic datasets and real datasets. For the single-cell metabolically labeled new RNA tagging sequencing (scNT-seq) dataset, the data are very noisy, so we restrict the model to only a small subset of velocity genes (STAR Methods section regularizing splicing direction). However, still more work needs to be done exploring these settings of LatentVelo, as well as the settings of other models.

Since all of the parameters of the neural networks in LatentVelo need to be initialized and since training is stochastic, there is some variation in fits. For most datasets, this is small; however, for some random seeds, the velocities are in wrong directions. A similar problem was encountered with UniTVelo and was addressed by initializing the latent time with a diffusion pseudotime based on a specified known root cell type or by fitting the model multiple times and selecting the one that is consistent with any prior knowledge.[17] We mitigate this problem on the mouse gastrulation, fibroblast reprogramming, and hindbrain GABA datasets by including a correlation in the loss function between inferred latent time and experimental time points for the first few epochs of training (see STAR Methods section incorporating experimental time points or root cells). We have found that incorrect velocities result from initialization, and this can be corrected early by simply influencing the correct direction with experimental time points. This also is a problem on the scNT, intestinal organoid, and bone marrow datasets, where experimental time points are not available. In these cases, the correct direction must be chosen out of multiple random seeds, or the "root" cells need to be specified with prior biological knowledge (see STAR Methods section incorporating experimental time points or root cells).

In cases where the incorrect velocity field is a nearly fully reversed version of the expected velocity field, we believe that this issue occurs due to a lack of genes showing cycling induction to repression behavior. The result is monotonic gene dynamics, where the direction is ambiguous. These problems occur on datasets where scVelo also has problems. In

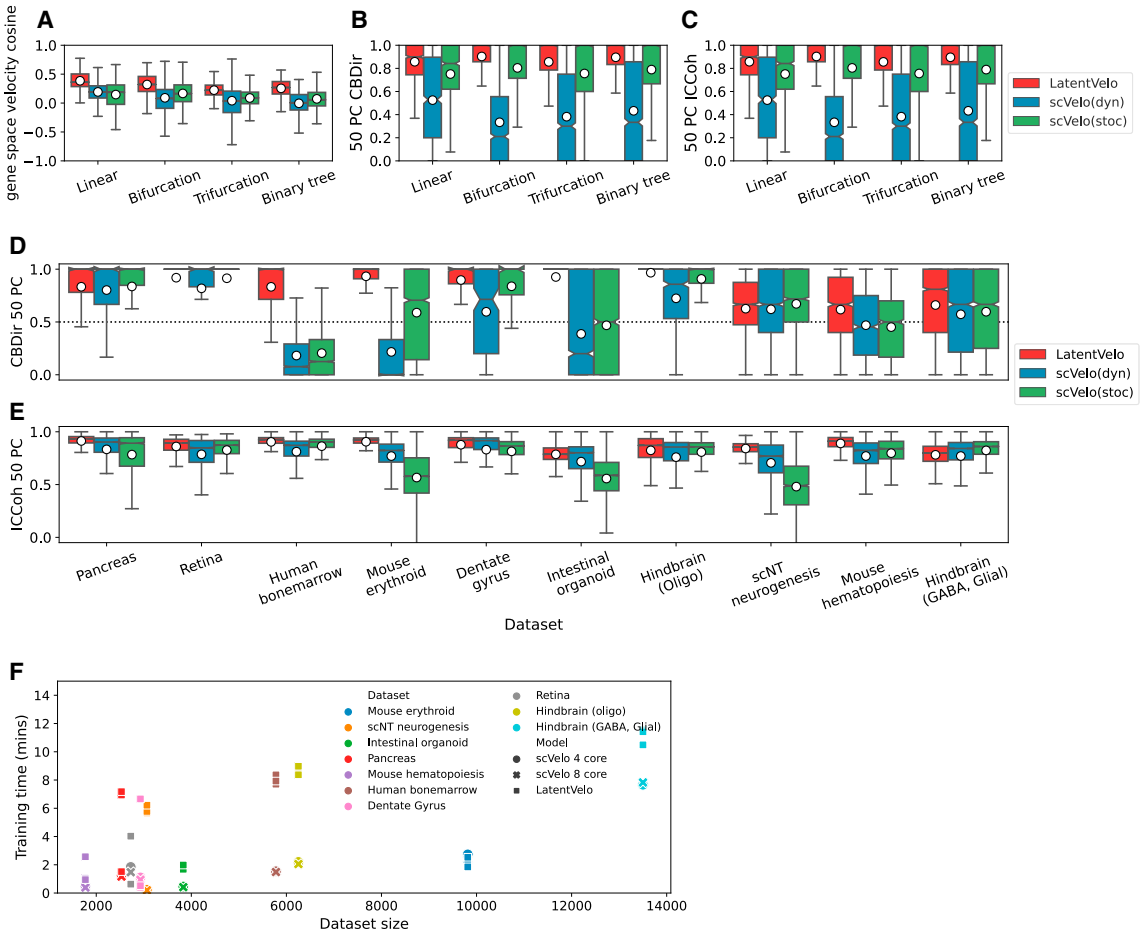


Figure 6. Quantitative benchmarking on synthetic and real data

(A–C) Quantitative metrics for the synthetic datasets.

(A) Cosine similarity between estimated and exact simulated velocity on the full gene space.

(B and C) Cross-boundary directedness (CBDir) score on the space of 50 principal components (PCs) (B) and inner-cluster coherence score (ICCoh) on the space of 50 PCs (C).

(D and E) Quantitative metrics for real datasets. (D) CBDir and (E) ICCoh scores on the space of 50 PCs. Higher scores for all metrics are better.

(F) Runtime comparison between LatentVelo and scVelo with 4 or 8 cores. LatentVelo is run with a GPU and run 3 times with different seeds due to the stochasticity in training.

LatentVelo, we offer the ability to specify root cells or use experimental time points to correct for this issue.

We have performed a robustness test of different values of hyperparameters in Figure S6 for the pancreas, bone marrow, retina, dentate gyrus, and synthetic bifurcation datasets. While the model generally performs well for many different hyperparameter settings, for some random seeds, velocities are incorrect (and not simply reversed). We have introduced a method of detecting some of these issues by examining individual gene unspliced vs. spliced expression plots (STAR Methods section regularizing splicing direction).

DISCUSSION

Previous RNA velocity methods have inferred velocity on gene space, with one exception being VeloAE, which only modeled steady-state dynamics.[16] LatentVelo embeds gene expression into a latent space, with the time progression of gene expression described by latent dynamics. Since the latent embedding and latent dynamics are trained together, this allows us to infer dynamics-informed embeddings of gene expression, which is a dimensional reduction of cell states, informed by the dynamics of the system. Shown with the fibroblast reprogramming dataset,[29] the UMAP representation of this latent space allows clear visualization of the separate trajectories taken by reprogrammed or dead-end cells. This opens up a new method of studying and characterizing the dynamics of cell differentiation, through the features represented in this latent space.

Interestingly, we have found that scVelo stochastic mode, which computes RNA velocity based on a linear regression between unspliced and spliced RNA at an assumed steady state, performed much better than scVelo dynamical mode. We believe this highlights the issues with the simple model of transcription in scVelo dynamical mode. The recent methods DeepVelo,[18]

UniTVelo,[17] VeloVAE,[19] and our approach LatentVelo all have different approaches to addressing this issue.

Since LatentVelo uses a VAE, we can sample to generate uncertainty estimates of latent times and velocities. This is shown in Figure S4, where uncertainty is largest near branching between multiple cell types. This method of uncertainty estimation is different from the method of scVelo,[11] which used the consistency of the velocity estimates of neighboring cells. Recent work has extensively explored these types of uncertainty estimates from variational Bayesian models of RNA velocity.[39,40] In Figure S4, we show that on a dataset of mature peripheral blood mononuclear cells (PBMCs),[41] LatentVelo infers large uncertainty in the velocities, as expected in this case of no differentiation dynamics. Further indicating no trajectories on this dataset, the latent space is totally unstructured, and latent times all are similar.

LatentVelo can be easily extended to multi-omics. In the STAR Methods section integrating multi-omic data with ATAC-seq and in Figure S5, we show an example of this with combined scRNA-seq and assay for transposase-accessible chromatin sequencing (ATAC-seq). This extension is done by adding a new variable to the structured latent dynamics $\mathbf{z}_c$, corresponding to the latent representation of chromatin accessibility, and modeling the regulation of chromatin by $\mathbf{z}_r$ and the regulation of transcription by $\mathbf{z}_c$ and $\mathbf{z}_r$. This demonstrates a key feature of LatentVelo: the ability to build general structured latent dynamics. This is presented as an example; we leave the more detailed analysis of multi-omics systems and exploration of other forms of structure in the dynamics to future work.

LatentVelo addresses some of the challenges raised in a recent review paper of RNA velocity[13]: multi-modal omics (STAR Methods section integrating multi-omic data with ATAC-seq), multi-variate dynamics, batch correction, lineage-/time-dependent rates, and implicit gene selection by embedding in the latent space. Two challenges not addressed are (1) stochastic dynamics and (2) normalization. We see potential ways to address these challenges with simple modifications to our model: (1) for stochastic dynamics, we can replace the latent ODEs in our model with stochastic differential equations (SDEs),[42] and (2) we can approach normalization in a similar way as scVI,[31] including normalization factors as latent variables to infer. Modeling SDEs instead of ODEs is a clear next direction for LatentVelo; in particular, by using a Bayesian approach to inferring the latent SDE,[42] we can simultaneously learn the prior SDE describing the full population of cells as well as the conditional SDE describing the dynamics of each cell conditioned on a particular branch (as we have done with the parameter $\mathbf{h}$). This would allow the use of the model for perturbed inputs—estimating future developmental trajectories for individual cells rather than just the past trajectories of cells as we have done here. Existing SDE models do not learn the latent embedding as LatentVelo does, nor do they incorporate splicing dynamics.[43]

Limitations of the study

One limitation of LatentVelo is the assumption of a single initial latent state from which the trajectory of all cells start from and that each observed cell can be reached by a continuous trajectory from this initial state. LatentVelo may encounter issues in situations with sparse cell clusters with few observed transient cells connecting trajectories. In these cases the model still infers generally accurate velocity directions for a given cell (Figure 6), but latent trajectories do not start from expected initial states. For example, on the dentate gyrus dataset,[11] we see the expected transitions on the granule lineage, radial glia to astrocyte transitions, and oligodendrocyte progenitors to oligodendrocyte transitions, but LatentVelo estimates the disconnected microglia cluster as having the lowest latent times, so inferred trajectories all incorrectly start from this cluster.

While we have shown LatentVelo is accurate on many datasets, for some, it requires additional hyperparameter adjustment by increasing the dimension of the latent states, using the cell-type annotated model, or restricting to only a subset of genes. Indeed, all recent improvements to RNA velocity also have multiple settings and hyperparameters, in addition to preprocessing steps.[14] While we have discussed some heuristics to choosing the hyperparameters of LatentVelo, more work needs to be done in this area.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - LatentVelo
  - Gene-space velocities
  - Regularizing splicing direction
  - Incorporating cell-type annotations
  - Incorporating experimental time points or root cells
  - RNA velocity and latent time uncertainty
  - Batch correction
  - ComBat and scGen RNA velocity batch effect correction
  - scVI and scANVI batch correction
  - Comparison velocity models
  - Differential expression and gene ontology analysis
  - Integrating multi-omic data with ATAC-seq
  - Datasets
- QUANTIFICATION AND STATISTICAL ANALYSIS
  - RNA velocity metrics
  - Batch correction metrics
  - Analyzing model fits and hyperparameters

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.crmeth.2023.100581.

ACKNOWLEDGMENTS

We thank Eric Johnson and Dominic Skinner for reading and providing feedback on the manuscript. S.F. received funding from a University of Toronto

Data Sciences Institute Postdoctoral fellowship. The authors received funding from a University of Toronto Medicine by Design grant.

AUTHOR CONTRIBUTIONS

S.F. and S.G. conceptualized the model. S.F. implemented the model and performed the analyses. S.F., S.G., and M.M. wrote and reviewed the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: January 11, 2023
Revised: June 16, 2023
Accepted: August 18, 2023
Published: September 13, 2023

REFERENCES

1. Deconinck, L., Cannoodt, R., Saelens, W., Deplancke, B., and Saeys, Y. (2021). Recent advances in trajectory inference from single-cell omics data. Curr. Opin. Struct. Biol. *27*, 100344.
2. Saelens, W., Cannoodt, R., Todorov, H., and Saeys, Y. (2019). A comparison of single-cell trajectory inference methods. Nat. Biotechnol. *37*, 547–554.
3. Ding, J., Sharon, N., and Bar-Joseph, Z. (2022). Temporal modelling using single-cell transcriptomics. Nat. Rev. Genet. *23*, 355–368.
4. Lange, M., Bergen, V., Klein, M., Setty, M., Reuter, B., Bakhti, M., Lickert, H., Ansari, M., Schniering, J., Schiller, H.B., et al. (2022). CellRank for directed single-cell fate mapping. Nat. Methods *19*, 159–170.
5. Qiu, X., Zhang, Y., Martin-Rufino, J.D., Weng, C., Hosseinzadeh, S., Yang, D., Pogson, A.N., Hein, M.Y., Hoi Joseph, M.K., Wang, L., Grody, E.I., et al. (2022). Mapping transcriptomic vector fields of single cells. Cell *185*, 690–711.e45.
6. Zhang, Z., and Zhang, X. (2021). Inference of high-resolution trajectories in single-cell RNA-seq data by using RNA velocity. Cell Rep. Methods *1*, 100095.
7. Gupta, R., Cerletti, D., Gut, G., Oxenius, A., and Claassen, M. (2020). Cytopath: Simulation based inference of differentiation trajectories from RNA velocity fields. Preprint at bioRxiv. https://doi.org/10.1101/2020.12.21.423801.
8. Chen, Z., King, W.C., Hwang, A., Gerstein, M., and Zhang, J. (2022). DeepVelo: Single-cell transcriptomic deep velocity field learning with neural ordinary differential equations. Preprint at bioRxiv. https://doi.org/10.1101/2022.02.15.480564.
9. Liu, R., Pisco, A.O., Braun, E., Linnarsson, S., and Zou, J. (2022). Dynamical systems model of RNA velocity improves inference of single-cell trajectory, pseudo-time and gene regulation. J. Mol. Biol. *434*, 167606.
10. La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., Lidschreiber, K., Kastriti, M.E., Lönnerberg, P., Furlan, A., et al. (2018). RNA velocity of single cells. Nature *560*, 494–498.
11. Bergen, V., Lange, M., Peidli, S., Wolf, F.A., and Theis, F.J. (2020). Generalizing RNA velocity to transient cell states through dynamical modeling. Nat. Biotechnol. *38*, 1408–1414.
12. Barile, M., Imaz-Rosshandler, I., Inzani, I., Ghazanfar, S., Nichols, J., Marioni, J.C., Guibentif, C., and Göttgens, B. (2021). Coordinated changes in gene expression kinetics underlie both mouse and human erythroid maturation. Genome Biol. *22*, 197.
13. Bergen, V., Soldatov, R.A., Kharchenko, P.V., and Theis, F.J. (2021). RNA velocity—current challenges and future perspectives. Mol. Syst. Biol. *17*, e10282.
14. Gorin, G., Fang, M., Chari, T., and Pachter, L. (2022). Rna velocity unraveled. PLoS Comput. Biol. *18*, e1010492.
15. Xu, C., Lopez, R., Mehlman, E., Regier, J., Jordan, M.I., and Yosef, N. (2021). Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. Mol. Syst. Biol. *17*, e9620.
16. Qiao, C., and Huang, Y. (2021). Representation learning of RNA velocity reveals robust cell transitions. Proc. Natl. Acad. Sci. USA *118*, e2105859118.
17. Gao, M., Qiao, C., and Huang, Y. (2022). UniTVelo: temporally unified RNA velocity reinforces single-cell trajectory inference. Nat. Commun. *13*, 6586.
18. Cui, H., Maan, H., and Wang, B. (2022). DeepVelo: Deep learning extends RNA velocity to multi-lineage systems with cell-specific kinetics. Preprint at bioRxiv. https://doi.org/10.1101/2022.04.03.486877.
19. Gu, Y., Blaauw, D., and Welch, J.D. (2022). Bayesian inference of rna velocity from multi-lineage single-cell data. Preprint at bioRxiv. https://doi.org/10.1101/2022.07.08.499381.
20. Qian, L. (2022). sctour: a deep learning architecture for robust inference and accurate prediction of cellular dynamics. Preprint at bioRxiv. https://doi.org/10.1101/2022.04.17.488600.
21. Huang, Y., and Sanguinetti, G. (2021). BRIE2: computational identification of splicing phenotypes from single-cell transcriptomic experiments. Genome Biol. *22*, 251.
22. Diederik, P. (2014). Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, 2nd International Conference on Learning Representations, ICLR 2014 (Conference Track Proceedings). *Banff, AB, Canada, April 14-16, 2014*.
23. Jimenez Rezende, D., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume.
24. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D.K. (2018). Neural ordinary differential equations. In Advances in Neural Information Processing Systems, *volume 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds. (Curran Associates, Inc.).
25. Cannoodt, R., Saelens, W., Deconinck, L., and Saeys, Y. (2021). Spearheading future omics analyses using dyngen, a multi-modal simulator of single cells. Nat. Commun. *12*, 3942–3949.
26. Bastidas-Ponce, A., Tritschler, S., Dony, L., Scheibner, K., Tarquis-Medina, M., Salinno, C., Schirge, S., Burtscher, I., Böttcher, A., Theis, F.J., et al. (2019). Comprehensive single cell mRNA profiling reveals a detailed roadmap for pancreatic endocrinogenesis. Development *146*, dev173849.
27. Collombat, P., Mansouri, A., Hecksher-Sorensen, J., Serup, P., Krull, J., Gradwohl, G., and Gruss, P. (2003). Opposing actions of arx and pax4 in endocrine pancreas development. Genes Dev. *17*, 2591–2603.
28. Battich, N., Beumer, J., de Barbanson, B., Krenning, L., Baron, C.S., Tanenbaum, M.E., Clevers, H., and van Oudenaarden, A. (2020). Sequencing metabolically labeled transcripts in single cells reveals mRNA turnover strategies. Science *367*, 1151–1156.
29. Biddy, B.A., Kong, W., Kamimoto, K., Guo, C., Waye, S.E., Sun, T., and Morris, S.A. (2018). Single-cell mapping of lineage and identity in direct reprogramming. Nature *564*, 219–224.
30. Luecken, M.D., Büttner, M., Chaichoompu, K., Danese, A., Interlandi, M., Mueller, M.F., Strobl, D.C., Zappia, L., Dugas, M., Colomé-Tatché, M., and Theis, F.J. (2022). Benchmarking atlas-level data integration in single-cell genomics. Nat. Methods *19*, 41–50.
31. Lopez, R., Regier, J., Cole, M.B., Jordan, M.I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. Nat. Methods *15*, 1053–1058.
32. Ranek, J.S., Stanley, N., and Purvis, J.E. (September 2022). Integrating temporal single-cell gene expression modalities for trajectory inference and disease prediction. Genome Biol. *23*, 186.
33. Hansen, K.D. (2021). Batch Effects in Scrna Velocity Analysis. https://www.hansenlab.org/velocity_batch.

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

35. Lotfollahi, M., Wolf, F.A., and Theis, F.J. (2019). scgen predicts single-cell perturbation responses. Nat. Methods *16*, 715–721.

36. Miao, Y., Djeffal, Y., De Simone, A., Zhu, K., Lee, J.G., Lu, Z., Silberfeld, A., Rao, J., Tarazona, O.A., Mongera, A., et al. (2022). Reconstruction and deconstruction of human somitogenesis in vitro. Nature *614*, 500–508.

37. Pijuan-Sala, B., Griffiths, J.A., Guibentif, C., Hiscock, T.W., Jawaid, W., Calero-Nieto, F.J., Mulas, C., Ibarra-Soria, X., Tyser, R.C.V., Ho, D.L.L., et al. (2019). A single-cell molecular map of mouse gastrulation and early organogenesis. Nature *566*, 490–495.

38. Lefebvre, V., and Bhattaram, P. (2010). Vertebrate skeletogenesis. Curr. Top. Dev. Biol. *90*, 291–317.

39. Adam, G., Weiler, P., Lotfollahi, M., Klein, D., Hong, J., Streets, A., Theis, F.J., and Nir, Y. (2022). Deep generative modeling of transcriptional dynamics for rna velocity analysis in single cells. Preprint at bioRxiv. https://doi.org/10.1101/2022.08.12.503709.

40. Qian, Q., Bingham, E., La Manno, G., Langenau, D.M., and Pinello, L. (2022). Pyro-velocity: Probabilistic rna velocity inference from single-cell data. Preprint at bioRxiv. https://doi.org/10.1101/2022.09.12.507691.

41. Zheng, G.X.Y., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., et al. (2017). Massively parallel digital transcriptional profiling of single cells. Nat. Commun. *8*, 14049.

42. Li, X., Ting-Kam Leonard, W., Chen, R.T.Q., and Duvenaud, D. (2020). Scalable gradients for stochastic differential equations. In International Conference on Artificial Intelligence and Statistics.

43. Grace Hui Ting Yeo; Saksena, S.D., and Gifford, D.K. (2021). Generative modeling of single-cell time series with PRESCIENT enables prediction of cell trajectories with interventions. Nat. Commun. *12*, 1–12.

44. Nestorowa, S., Hamey, F.K., Pijuan Sala, B., Diamanti, E., Shepherd, M., Laurenti, E., Wilson, N.K., Kent, D.G., and Göttgens, B. (2016). A single-cell resolution map of mouse hematopoietic stem and progenitor cell differentiation. Blood *128*, e20–e31.

45. Lo Giudice, Q., Leleu, M., La Manno, G., and Fabre, P.J. (2019). Single-cell transcriptional logic of cell-fate specification and axon guidance in early-born retinal neurons. Development *146*, dev178103.

46. Hochgerner, H., Zeisel, A., Lönnerberg, P., and Linnarsson, S. (2018). Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell RNA sequencing. Nat. Neurosci. *21*, 290–299.

47. Zeisel, A., Hochgerner, H., Lönnerberg, P., Johnsson, A., Memic, F., van der Zwan, J., Häring, M., Braun, E., Borm, L.E., La Manno, G., et al. (2018). Molecular architecture of the mouse nervous system. Cell *174*, 999–1014.e22.

48. Vladoiu, M.C., El-Hamamy, I., Donovan, L.K., Farooq, H., Holgado, B.L., Sundaravadanam, Y., Ramaswamy, V., Hendrikse, L.D., Kumar, S., Mack, S.C., et al. (2019). Childhood cerebellar tumours mirror conserved fetal transcriptional programs. Nature *572*, 67–73.

49. Setty, M., Kiseliovas, V., Levine, J., Gayoso, A., Mazutis, L., and Dana, Pe'er (2019). Characterization of cell fate probabilities in single-cell data with palantir. Nat. Biotechnol. *37*, 451–460.

50. Qiu, Q., Hu, P., Qiu, X., Govek, K.W., Cámara, P.G., and Wu, H. (2020). Massively parallel and time-resolved rna sequencing in single cells with scnt-seq. Nat. Methods *17*, 991–1001.

51. Wolf, F.A., Angerer, P., and Theis, F.J. (2018). SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. *19*, 15.

52. Gayoso, A., Lopez, R., Xing, G., Boyeau, P., Valiollah Pour Amiri, V., Hong, J., Wu, K., Jayasuriya, M., Mehlman, E., Langevin, M., et al. (2022). A python library for probabilistic analysis of single-cell omics data. Nat. Biotechnol. *40*, 163–166.

53. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. *12*, 2825–2830.

54. Fang, Z., Liu, X., and Peltz, G. (2023). GSEApy: a comprehensive package for performing gene set enrichment analysis in python. Bioinformatics *39*, btac757.

55. Ricky, T., and Chen, Q. (2018). torchdiffeq. https://github.com/rtqichen/torchdiffeq.

56. Blei, D.M., Kucukelbir, A., and McAuliffe, J.D. (2017). Variational inference: A review for statisticians. J. Am. Stat. Assoc. *112*, 859–877.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Pancreatic endocrinogenesis	Bastidas-Ponce et al. 2019[26]	GSE132188
Mouse hematopoiesis	Nestorowa et al. 2016[44]	GSE81682
Mouse retina development	Lo Giudice et al. 2019[45]	GSE122466
Dentate Gyrus development	Hochgerner et al. 2018[46]	GSE95753
Intestinal organoid	Battich et al. 2020[28]	GSE128365
Mouse hindbrain (oligo)	Zeisel et al. 2018[47]	SRP135960
Mouse hindbrain (GABA, Glial)	Vladoiu et al. 2019[48]	GSE118068
Mouse gastrulation	Pijuan-Sala et al. 2019[37]	E-MTAB-6967
Human bone marrow	Setty et al. 2018[49]	ERP120467
scNT-seq neuron KCl stimulation	Qiu et al. 2020[50]	GSE141851
Mouse embryonic fibroblast reprogramming	Biddy et al. 2018[29]	GSE99915
Somitogenesis segmentoid	Miao et al. 2022[36]	GSE195467
PBMC	Zheng et al. 2017[41]	SRP073767
Embryonic mouse brain	10x Genomics	https://www.10xgenomics.com/resources/datasets/fresh-embryonic-e−18-mouse-brain-5-k-1-standard-1-0-0
Software and algorithms		
LatentVelo	This manuscript	https://github.com/Spencerfar/LatentVelo https://doi.org/10.5281/zenodo.8248696
scanpy (version 1.9.1)	Wolf et al. 2018[51]	https://github.com/theislab/scanpy
scVelo (0.2.4)	Bergen et al. 2020[11]	https://github.com/theislab/scvelo
scvi-tools (0.15.0)	Gayoso et al. 2022[39,52]	https://scvi-tools.org/
UniTVelo	Gao et al. 2022[17]	https://github.com/StatBiomed/UniTVelo
dyngen (version 1.0.5)	Cannoodt et al. 2021[25]	https://github.com/dynverse/dyngen
CellRank (version 1.5.1)	Lange et al. 2022[4]	https://github.com/theislab/cellrank
scgen (version 2.1.0)	Lotfollahi et al. 2019[35]	https://github.com/theislab/scgen
scib (version 1.0.3)	Luecken et al. 2022[30]	https://github.com/theislab/scib
pytorch (version 1.11.0)	PyTorch Foundation	https://pytorch.org/
scikit-learn (version 1.1.1)	Pedregosa et al. 2011[53]	https://scikit-learn.org/
gseapy (version 0.10.8)	Fang et al. 2022[54]	https://github.com/zqfang/GSEApy

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Spencer Farrell (spencer.farrell@utoronto.ca).

Materials availability

This study did not generate new unique reagents.

Data and code availability

- The datasets that are analyzed within the current study are publicly available. The accession numbers are listed in the key resources table.
- LatentVelo is publicly available as a Python package on Github (https://github.com/Spencerfar/LatentVelo). Notebooks for running all of the analysis performed in this manuscript are available. The DOI is listed in the key resources table.

- Any addition information that may be required to reanalyze the data reported in this paper is available from the lead contact upon request.

METHOD DETAILS

LatentVelo

We take spliced and unspliced counts per cell as input and embed into a latent space $\widehat{\mathbf{z}}$ with an encoder neural network. A separate encoder is used for spliced and unspliced counts, partitioning the latent space as $\widehat{\mathbf{z}} = (\widehat{\mathbf{z}}_s, \widehat{\mathbf{z}}_u)$. Our model is trained as a Variational Autoencoder (VAE), where we use a standard normal prior on the latent space $\widehat{\mathbf{z}} \sim \text{Normal}(0, 1)$. We also use an encoder to estimate a latent developmental time with a logit-normal prior $t \sim \text{LogitNormal}(0, 1)$.

Dynamics on the latent space are described by neural ordinary differential equations,

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}(\mathbf{z}(t), \mathbf{h}), \qquad \text{(Equation 12)}$$

where $\mathbf{f}$ is a neural network describing the velocity field of the dynamics. The spliced component of this velocity field represents latent RNA velocity. The structure of $\mathbf{f}$ is decomposed into the 3 components described in Figure 1, $\mathbf{f} = (\mathbf{f}_u, \mathbf{f}_s, \mathbf{f}_r)$.

These dynamics are coupled to the auto-encoder by matching the ODE solution $\mathbf{z}(\widehat{t})$ at time $\widehat{t}$ with the encoded latent state $\widehat{\mathbf{z}}$. The ODE is solved with the dopri5 solver from the torchdiffeq package.[55]

We use an approximate posterior factorized as $q(\widehat{\mathbf{z}}, \widehat{t}|\mathbf{x}) = q(\widehat{t}|\widehat{\mathbf{z}})q(\widehat{\mathbf{z}}|\mathbf{x})$, where $\mathbf{x} = (\mathbf{s}, \mathbf{u})$ are the observed spliced and unspliced counts. Separate encoders for spliced and unspliced latent states are used, such that $q(\widehat{\mathbf{z}}|\mathbf{x}) = q(\widehat{\mathbf{z}}_s|\mathbf{s})q(\widehat{\mathbf{z}}_u|\mathbf{u})$. Additionally, we also use separate decoders, $p(\mathbf{s},\mathbf{u}|\mathbf{z}) = p(\mathbf{s}|\mathbf{z})p(\mathbf{u}|\mathbf{z})$. Since the data are very noisy, we find that we cannot use a count-based distribution such as a negative-binomial, so instead use a Gaussian with mean functions $\mu(\mathbf{z}) = (\mu(\mathbf{z}_s), \mu(\mathbf{z}_u))$ with smoothed and normalized counts following the scVelo preprocessing procedure.[11]

The latent dynamics in Equation 12 are solved until $\widehat{t}$ for each cell. The solution $\mathbf{z}(\widehat{t})$ is matched to the encoded state $\widehat{\mathbf{z}}$ by assuming $(\widehat{\mathbf{z}} - \mathbf{z}(\widehat{t}))$ follows a Gaussian distribution, where we also fit the standard deviation σ_z.

Our model is trained with the loss:

$$L = -\mathbb{E}_{\widehat{\mathbf{z}},\widehat{t}\sim q}[\text{logNormal}(\mathbf{x}|\mu(\widehat{\mathbf{z}}), \sigma)] + \text{KL}(q\|p) - \mathbb{E}_{\widehat{\mathbf{z}},\widehat{t}\sim q}[\text{logNormal}(\mathbf{x}|\mu(\mathbf{z}(\widehat{t})), \sigma) - \text{logNormal}(\widehat{\mathbf{z}}|\mathbf{z}(\widehat{t}), \sigma_z)]. \qquad \text{(Equation 13)}$$

Expectations are computed by sampling $\widehat{\mathbf{z}}$, $\widehat{t}$, and $\mathbf{h}$ from the encoder, and solving the latent dynamics for $\mathbf{z}(\widehat{t})$. In STAR Methods sections XII C and XII E we discuss further terms in this loss function regularizing the trajectory direction.

The first two terms of this loss are the standard negative evidence lower bound of a VAE,[22,23] representing the expectation of the negative log likelihood with a Gaussian, and the KL divergence between the posterior $q(\widehat{\mathbf{z}}, \widehat{t}|\mathbf{x})$ and the prior, $\text{Normal}(0, 1) \times \text{LogitNormal}(0, 1)$. The second last term is the loss for the decoded solution of the dynamics $\mathbf{z}(\widehat{t})$. The last term penalizes the distance between the encoded latent state $\widehat{\mathbf{z}}$ and the latent state estimated by the dynamics $\mathbf{z}(\widehat{t})$ – matching the latent dynamics to the encoded states.

By default, we use a 20-dimensional latent state for each observed component (spliced, unspliced) and use ELU activations throughout. We follow VeloAE[16] and use a encoder structured as an initial fully connected dense neural network with 1 hidden layer (of size 25 by default) as the first part, and then use two graph convolution layers using a 30 nearest-neighbor graph computed on the 30 principle components (same as was used for smoothing) as the similarity graph for the second part. This enables the encoder to use information about nearest neighbors when computing the latent embedding. In the case of batch correction, we do not use the graph convolution layers and instead just use a fully connected dense neural network with 2 hidden layers.

The encoders for $\mathbf{h}$ and $\widehat{t}$ are also graph convolutional networks with 2 layers of graph convolutions. The differential equation derivative functions $\mathbf{f}_u$ and $\mathbf{f}_r$ have a single hidden layer of size 25. $\mathbf{f}_s$ is linear in $\mathbf{z}_s$ and $\mathbf{z}_u$. By default we also use a linear decoder, and use separate linear decoders per batch for batch correction (with the option for a fully connected neural network). The dimension of $\mathbf{z}_r$ is chosen based on the expected number of branches. For one or two expected branches, we use 1 dimensions, for 3 expected branches we use 2 dimensions. In general, we take the dimension of $\mathbf{z}_r$ to be 1 less than the number of expected branches. Figure S7 shows a diagram of these architectures.

We use 90% of the data for training, and use the other 10% as a validation set to monitor training progress. We train for 50 epochs, and use the model at the epoch with lowest mean-squared error on the validation set. By default we train with the Adam optimizer with a learning rate of 0.01 and a batch size of 100. In large datasets (e.g., mouse gastrulation or Fibroblast reprogramming), we increase the batch size to 1000. In scenarios with exploding gradients resulting in failed training, we use gradient clipping to stabilize training. We increase the KL divergence weight in the VAE from 0 to 1 linearly over the first 25 epochs.

The model is implemented in *pytorch* utilizing the *torchdiffeq* package for neural ODEs,[24,55] which critically allows computing gradients of the inferred latent time-points $\widehat{t}$ in the solver, rather than marginalizing over $\widehat{t}$ by integrating. We found the marginalization approach to be challenging.

Gene-space velocities

While the main focus of Latentvelo is to infer latent dynamics, we can also infer the dynamics of single genes. We compute the velocity of single genes in LatentVelo by transforming the latent space velocities into gene-space velocities by utilizing the decoder: $\dot{\mathbf{s}} = J[D_s](\mathbf{z}_s)\mathbf{f}_s(\mathbf{z}_s,\mathbf{z}_u)$, where $J[D_s](\mathbf{z}_s)$ is the Jacobian of the spliced decoder evaluated at $\mathbf{z}_s$ and $\mathbf{f}_s$ is the latent space velocity. However since the high-dimensional gene space is compressed into a low-dimension latent space, we expect the dynamics of only the best reconstructed genes in the autoencoder to have their dynamics be well modeled.

Regularizing splicing direction

The linear ODEs describing transcription, splicing, and degradation are,

$$\dot{u_g}(t) = \alpha_g(t) - \beta_g u_g(t) \quad \text{(Equation 14)}$$

$$\dot{s_g}(t) = \beta_g u_g(t) - \gamma_g s_g(t) \quad \text{(Equation 15)}$$

per gene g. These equations enforce the casual relationship of splicing between unspliced and spliced RNA. Generalizing, this same effect can be achieved with $\frac{\partial \dot{s_g}}{\partial u_g} > 0$ and $\frac{\partial \dot{s_g}}{\partial s_g} < 0$.

To further constrain the direction of splicing to accurately represent the biology (unspliced to spliced), we need to enforce $\frac{\partial \dot{s_g}}{\partial u_g} > 0$ and $\frac{\partial \dot{s_g}}{\partial s_g} < 0$ for each gene g. However, enforcing this constraint for general forms of the velocity $\dot{s}$ is difficult, since we need to compute the Jacobian, which is a matrix of size $2N^2$, where N is the number of genes. Computing this would require back-propagation through the entire model back to the input, which is computationally infeasible. Instead, we can use a much faster approach to enforce this direction by using the correlation between velocity and spliced and unspliced counts.

We weakly regularize by the correlation between gene-space velocity and input data (similar to DeepVelo[18]),

$$\lambda_{su}\text{corr}(\dot{\mathbf{s}}, \mathbf{u}) + \lambda_{ss}\text{corr}(\dot{\mathbf{s}}, -\mathbf{s}). \quad \text{(Equation 16)}$$

By default we take $\lambda_{su} = \lambda_{su} = 0.1$. The goal with this term is to weakly regularize the direction, rather than match the strict linear dependence seen in other models. A similar form of regularization is also done in DeepVelo.[18] To compute gene-space velocity we compute the time-derivative of the transformation with the decoder, $\dot{\mathbf{s}} = J[\mu_s](\mathbf{z}_s)\dot{\mathbf{z}}_s$, where $J[\mu_s]$ is the Jacobian of the spliced decoder. Similarly we can compute the unspliced gene-space velocity, $\dot{\mathbf{u}} = J[\mu_u](\mathbf{z}_u)\dot{\mathbf{z}}_u$. Note, the Jacobian here is taken with respect to the latent space which has a smaller dimension, and this is a Jacobian vector product, which are easily computable by backpropagation in comparison to the Jacobian discussed above. This regularization can be done per celltype, only including the same type cells in the correlation.

We only apply this regularization to genes that show significant splicing dynamics. These "velocity genes" are identified in a similar way to UniTVelo and scVelo.[17] We fit a linear regression between spliced and unspliced data per gene, then only select genes with R^2 score above 0.05 and below 0.95. In cases of genes with very high R^2, all cells lie on a straight line in the u vs. s plane, showing no splicing dynamics. In cases of genes with very low R^2, cells are scattered uniformly in the u vs. s plane, showing no splicing dynamics. Genes with an extreme ratio of standard deviations between unspliced and spliced were also filtered so that $0.3 \leq \sigma_u/\sigma_u \leq 3$, given that this may be the result of error in reading unspliced counts.

Incorporating cell-type annotations

We follow scANVI,[15] and incorporate cell-type annotations by modifying the prior. We introduce the new latent state $\widehat{\mathbf{w}}$, and use the new approximate posterior,

$$q(\widehat{\mathbf{z}}, \widehat{\mathbf{w}}, \widehat{t}|\mathbf{x}, c) = q(\widehat{\mathbf{z}}|\mathbf{x})q(\widehat{\mathbf{w}}|c, \widehat{\mathbf{z}})q(\widehat{\mathbf{t}}|\widehat{\mathbf{z}}). \quad \text{(Equation 17)}$$

We place a standard normal prior on $\mathbf{w}$, and model $\widehat{\mathbf{z}}$ and $\widehat{t}$ as functions of $\mathbf{w}$ and the cell-type annotations c instead of placing a standard normal/logitnormal priors; $p(\widehat{\mathbf{z}}|\widehat{\mathbf{w}}, c) = \text{Normal}(f_{\mu,z}(\widehat{\mathbf{w}}, c), f_{\sigma,z}(\widehat{\mathbf{w}}, c))$ and $p(\widehat{t}|\widehat{\mathbf{w}}, c) = \text{LogitNormal}(f_{\mu,t}(\widehat{\mathbf{w}}, c), f_{\sigma,t}(\widehat{\mathbf{w}}, c))$.

The effect of this modification is to allow the priors on $\widehat{\mathbf{z}}$ and $\widehat{t}$ to vary with cell-type, preventing over-regularizing and eliminating biologically relevant cell-type clusters for the sake of matching the restrictive prior.

Incorporating experimental time points or root cells

We can include a regulation term to enforce a positive correlation between inferred latent time and experimental time when available, $\lambda_t\text{corr}(\widehat{t}, t_{e,i})$, where $t_{e,i}$ is the experimental time point for the ith cell. The weight of this correlation can be linearly decayed over the first few epochs to fix reversed velocities.

In scenarios where no experimental time-points are known, a "root" celltype can be used in reversed velocity scenarios. In these cases we can regularize the estimated latent time $\widehat{t}$ to be minimal for the specified root celltype.

RNA velocity and latent time uncertainty

To estimate uncertainty in latent time, we sample from the inferred posterior distribution $q(\hat{t}|\hat{\mathbf{z}}_u, \hat{\mathbf{z}}_s)$ and compute the standard error of the mean.

To estimate uncertainty in latent RNA velocity, we sample from the inferred posterior distribution of states $q(\hat{\mathbf{z}}_s, \hat{\mathbf{z}}_u|\mathbf{s}, \mathbf{u})$ and compute latent RNA velocity for each sample $\mathbf{v} = \mathbf{f}_s(\hat{\mathbf{z}}_s, \hat{\mathbf{z}}_u)$. We then compute the average cosine similarity of all pairs from these samples as the consistency of velocity,

$$c_v = \mathbb{E}_{i>j \text{ samples}}[\cos(\mathbf{v}_i, \mathbf{v}_j)]. \quad \text{(Equation 18)}$$

However, we note that variational Bayesian approximations underestimate uncertainty,[56] and so here we only interpret these uncertainty estimates to be *relative* estimates of uncertainty.

Batch correction

To perform batch correction, we include the batch ID into the encoder and decoder. For the encoder, this is of the form of a one-hot vector – a vector with length equal to the number of batches containing all zeros except at the index corresponding to the particular batch. For a feedforward neural network decoder we also use this one-hot vector. For the linear decoder we use a separate linear decoder per batch.

The latent dynamics remain independent of the batch ID. The priors on the latent space and latent time are also independent of the batch ID. Both of these conditions encourage the latent space to be independent of the batch ID, putting all handling of distinct batches in the encoder and decoder.

ComBat and scGen RNA velocity batch effect correction

We follow the approach taken by Hansen and Ranek et al.[32,33] to compare our approach for batch effect correction of RNA velocity. Since traditional RNA velocity methods require cells in gene-space, only batch correction methods that return a corrected gene matrix can be used. Here we use ComBat[34] and scGen (using the corrected gene output).[35] ComBat is run from the *scanpy* package.[51]

Since we need to simultaneously correct spliced and unspliced counts, batch correction is performed on the sum of these counts. We define the sum matrix M and the ratio matrix R,

$$M = S + U, \quad \text{(Equation 19)}$$

$$R = \frac{S}{S+U}. \quad \text{(Equation 20)}$$

Batch correction is performed on M. For ComBat, we first $\log(1+x)$ transform this matrix. For scGen, we normalize then $\log(1+x)$ transform.

After batch correcting M to get the corrected matrix $\tilde{M}$, we invert these transforms and then multiply R or $1-R$ to recover corrected spliced and unspliced matrices.

$$S_{\text{corrected}} = \tilde{M}R, \quad \text{(Equation 21)}$$

$$U_{\text{corrected}} = \tilde{M}(1-R). \quad \text{(Equation 22)}$$

Then RNA velocity is estimated as before with the same pre-processing steps.

scGen is run with default settings, except we set the dimension of the latent state to be the same as our model, 20.

scVI and scANVI batch correction

scVI and scANVI are run from the scvi-tools package.[52] We run scVI with a negative binomial gene-likelihood, and the same latent dimension as our model, 20. scANVI is trained by starting with the pre-trained scVI model and training for an additional 100 epochs.

Comparison velocity models

scVelo stochastic and dynamical modes are run with default settings. UniTVelo is run with the defaults IROOT = *None* and R2_ADJUST = *True*.

When we compute the transition matrix to embed RNA velocity in 50 dimensional PC space or 2 dimensional UMAP or tSNE for plotting, we use scVelo's function scvelo.tl.velocity_graph with default settings.

Differential expression and gene ontology analysis

To perform differential expression analysis, we first select cells at the particular terminal cell states. We then use LatentVelo to predict the inferred trajectories to these terminal cells. Then along these trajectories predicted by LatentVelo, we select the closest cells

within the dataset based on euclidean distance in the latent space $(\mathbf{z}_u, \mathbf{z}_s, \mathbf{z}_r)$. We then use a Wilcoxon rank-sum one-sided test comparing the cells along the trajectory of interest to the others. We select genes with p values below 10^{-5}. We then order these genes by the absolute value of their Wilcoxon rank-sum test statistic, then take the top 250 differentially expressed genes as input for gene ontology (GO) analysis.

To preform GO analysis, we use the enrichr function from the gseapy python package[54] with the "GO_Biological_Process_2021" gene set. We select GO terms with adjusted p values below 0.05, and plot the top 10 terms.

For the pancreas dataset, we select later time trajectories (latent time > 0.3, starting from Fev+ cells). For the mouse gastrulation dataset, select latent times above 0.25 (eliminating epiblast cells). Choosing these later latent time cells removes the common genes seen in all trajectories within the early progenitor cells.

Integrating multi-omic data with ATAC-seq

LatentVelo can be generalized to multiomics by adjusting the structure in the latent space. We include chromatin accessibility with a new latent variable $\mathbf{z}_c$. In this model, chromatin accessibility effects transcription of unspliced RNA $\mathbf{z}_u$, and chromatin dynamics are regulated by $\mathbf{z}_r$. We still also allow the direct regulation of transcription by other methods than just chromatin accessibility by retaining the connection between $\mathbf{z}_u$ and $\mathbf{z}_r$. The diagram in Figure S5 shows the new structure of the latent variables. The form of the dynamics for this model are:

$$\frac{d\mathbf{z}_c(t)}{dt} = \mathbf{f}_c(\mathbf{z}_c(t), \mathbf{z}_r(t)) \quad \text{(Equation 23)}$$

$$\frac{d\mathbf{z}_u(t)}{dt} = \mathbf{f}_u(\mathbf{z}_u(t), \mathbf{z}_c(t), \mathbf{z}_r(t)) \quad \text{(Equation 24)}$$

$$\frac{d\mathbf{z}_s(t)}{dt} = \mathbf{f}_s(\mathbf{z}_u(t), \mathbf{z}_s(t)), \quad \text{(Equation 25)}$$

$$\frac{d\mathbf{z}_r(t)}{dt} = \mathbf{f}_r(\mathbf{z}_s(t), \mathbf{z}_r(t), \mathbf{h}), \quad \text{(Equation 26)}$$

$$\mathbf{h} = \mathbf{f}_h(\widehat{\mathbf{z}}_s, \widehat{\mathbf{z}}_u). \quad \text{(Equation 27)}$$

In addition to these equations, we adapt the correlation regularization on gene-space for the unspliced dynamics, and now use the regularization,

$$\lambda_{su}\text{corr}(\dot{\mathbf{s}}, \mathbf{u}) + \lambda_{ss}\text{corr}(\dot{\mathbf{s}}, -\mathbf{s}) + \lambda_{uc}\text{corr}(\dot{\mathbf{u}}, \mathbf{c}), \quad \text{(Equation 28)}$$

to enforce the direction of transition based on chromatin accessibility in addition to splicing. We use the same default $\lambda_{su} = \lambda_{ss} = \lambda_{uc} = 0.1$

Datasets

For each dataset we select genes with at least 20 cells with non-zero unspliced and spliced counts, and from these genes select the top 2000 highly variable genes using scVelo preprocessing. We apply the transformation $log(1 + x)$ before computing principle components and smooth data by averaging over the 30 nearest neighbors in 30 dimensional principal component space with the scVelo function scvelo.pp.moments. Input variables to the model are then scaled to standard deviation 1. For each dataset, we use the provided celltype annotations used in the original publication.

Dyngen synthetic datasets

We generate synthetic datasets with dyngen.[25] For each dataset in Figure 6, we simulate 5000 cells. We set 15 target genes and 15 housekeeping genes, and use the number of transcription factors required for the trajectory backbone. We use the backbones "linear", "bifurcating", "binary tree" with 2 modifications, and "trifurcating". We set $\tau = 0.01$, census_interval = 1, and use 100 simulations. For the CBDir metric, we use the milestones defined by dyngen as transitions. We set the dimension of $\mathbf{z}_r$ to be 1 on the linear and bifurcation datasets, and 2 on the trifurcation and binary tree datasets. We set the dimension of $\mathbf{h}$ to be 1 on the linear and bifurcation datasets, and 3 on the trifurcation and binary tree datasets. For the synthetic batch correction datasets (bifurcation and gene module knockout), we set the dimension of z_r to be 1 and the diumnension of h to be 1.

To generate the simulated data with batch effects with dyngen, cell kinetic parameters are sampled from the same distributions in different realizations with the same gene regulatory networks, as suggested in the original dyngen paper.[25]

scVelo linear model synthetic datasets

We use the scVelo linear differential equations to simulate datasets with time-varying alpha (transcription rate), beta (splicing rate), gamma (degradation rate). We simulate 500 cells with 30 genes. By default, we set $\alpha = 5, \beta = 0.5, \gamma = 0.5$. For time-varying rates, we select 5 genes to be time-dependent. We set the dimension of $\mathbf{z}_r$ to be 1 on these datasets.

Pancreatic endocrinogenesis

Mouse pancreatic cells sampled at E15.5.[26] The initially cycling population is removed to focus on differentiation into terminal cell types. This dataset is downloaded from the CellRank package.[4] The transitions tested with CBDir are (Ngn3 low EP → Ngn3 high EP), (Ngn3 high EP → Fev+), (Fev+ → Delta), (Fev+ → Beta), (Fev+ → Epsilon), (Fev+ → Alpha). We set the dimensions of $\mathbf{z}_r$ and $\mathbf{h}$ to be 3 and on this dataset, and use a latent dimension of size 30. We use the celltype annotated model for this dataset.

Mouse hematopoiesis

Data is from,[44] and processed data is downloaded from https://zenodo.org/record/6110279.[32] The transitions tested with CBDir are (LTHSC → MPP), (MPP → LMPP), (MPP → CMP), (CMP → GMP), (CMP → MEP). We set the dimension of $\mathbf{z}_r$ and $\mathbf{h}$ to be 1 and 2 on this dataset.

Mouse retina development

Data is from,[45] downloaded from the Kharchenko lab http://pklab.med.harvard.edu/peterk/review2020/examples/retina/. The transitions tested with CBDir are (Neuroblast → PR), (Neuroblast → AC/HC), (Neuroblast → RGC). We set the dimension of $\mathbf{z}_r$ and $\mathbf{h}$ to be 2 and 2 on this dataset.

Dentate gyrus development

Mouse Dentate Gyrus development,[46] at two time points P12 and P35 downloaded from the scVelo package.[11] The transitions tested with CBDir are (OPC → OL), (Radial Glia-like → Astrocytes), (Neuroblast → Granule immature). We set the dimensions of $\mathbf{z}_r$ $\mathbf{h}$ to be 3 and 4 on this dataset.

Intestinal organoid

Data from,[28] downloaded from dynamo.[5] The transitions tested with CBDir are (Stem cells → TA cells), (Stem cells → Goblet cells), (Stem cells → Tuft cells), (TA cells → Enterocytes). We set the dimensions of $\mathbf{z}_r$ and $\mathbf{h}$ to be 2 and 3 on this dataset. We use the celltype annotated model for this dataset.

Mouse hindbrain (oligo)

Data of mouse hindbrain oligodendrocyte lineage from,[47] downloaded from http://pklab.med.harvard.edu/ruslan/velocity/oligos/. The transitions tested with CBDir are (COPs → NFOLs), (NFOLs → MFOLs). We set the dimension of $\mathbf{z}_r$ and $\mathbf{h}$ to be 1 and 2 on this dataset. We use the celltype annotated model for this dataset.

Mouse hindbrain (GABA, glial)

Data of mouse hindbrain with the differentiation of GABAergic interneuron and glial cells,[48] procesed data is downloaded from DeepVelo.[18] The transitions tested with CBDir are (Neural stem cells → progenitors), (Proliferating VZ progenitors → VZ progenitors), (VZ progenitors → Gliogenic progenitors), (VZ progenitors → Differentiating GABA interneurons), (Differentiating GABA interneurons → GABA interneurons). We set the dimensions of $\mathbf{z}_r$ and $\mathbf{h}$ to be 1 and 2 on this dataset. We use the celltype annotated model for this dataset.

Mouse erythroid

Erythroid lineage of mouse gastrulation.[37] Downloaded from the scVelo package.[11] The transitions tested with CBDir are (Blood progenitors 1 → Blood progenitors 2), (Blood progenitors 2 → Erythroid1), (Erythroid1 → Erythroid2), (Erythroid2 → Erythroid3). We set the dimension of $\mathbf{z}_r$ and $\mathbf{h}$ to be 1 and 1 on this dataset.

Human bone marrow

Data from,[49] downloaded with scVelo.[11] The transitions tested with CBDir are (HSC 1 → CLP), (HSC 1 → Mega), (HSC 1 → Ery 1), (Ery 1 → Ery 2), (HSC 1 → HSC 2), (HSC 2 → Precursors), (HSC 2 → Mono 2), (Mono 2 → Mono 1), (Precursors → DCs). We set the dimension of $\mathbf{z}_r$ and $\mathbf{h}$ to be 2 and 2 on this dataset.

scNT-seq neuron KCl stimulation

Cortical neurons are stimulated with potassium chloride (KCl) for 0, 15, 30, and 60 min. Data from,[50] downloaded from https://github.com/wulabupenn/scNT-seq using single-cell metabolically labeled new RNA tagging sequencing (scNT-seq). The transitions tested with CBDir are the times (0 → 15), (15 → 30), (30 → 60), (60 → 120). We set the dimension of $\mathbf{z}_r$ and $\mathbf{h}$ to be 1 and 1 on this dataset. We also restrict to only including the velocity genes in the likelihood.

Mouse embryonic fibroblast reprogramming

Reprogramming of mouse embryonic fibroblasts into induced endoderm progenitor cells.[29] Data was downloaded with the CellRank package.[4] We set the dimension of $\mathbf{z}_r$ to be 1 on this dataset. We use the celltype annotated model on this dataset.

Mouse gastrulation

Mouse gastrulation including all progenitors of major organs.[37] Downloaded from the scVelo package.[11] We subset to 20000 cells selected randomly for faster training. We set the dimensions of $\mathbf{z}_r$ and $\mathbf{h}$ to be 6 and 7 on this dataset. We use the celltype annotated model for this dataset. We use the experimental times to regularize the velocity direction.

Segmentoid

Somitogenesis organoid[36] with 4 different timepoints at 24, 48, 72, and 98 h. Downloaded from the https://github.com/PourquieLab/Miao_Djeffal_2022. The transitions tested with CBDir are (NMP → Neural), (NMP → PSM), (PSM → Somite). We follow the

preprocessing in the original analysis. We set the dimensions of $\mathbf{z}_r$ and $\mathbf{h}$ to be 2 and 2 on this dataset. We set NMP cells as the root cells, which are the most prevalent celltype at the earliest timepoint.

PBMC

PBMC dataset,[41] downloaded with scVelo.[11] We follow the preprocessing in the original analysis. We randomly sample 30000 cells from the data for faster training. We set the dimensions of $\mathbf{z}_r$ and $\mathbf{h}$ to be 1 and 2 on this dataset. We use a latent state size of 50.

Embryonic mouse brain

Data downloaded from 10X https://www.10xgenomics.com/resources/datasets/fresh-embryonic-e-18-mouse-brain-5-k-1-standard-1-0-0. We followed the pre-processing from MultiVelo.[20] We set the dimension of $\mathbf{z}_r$ to be 2 on this dataset.

QUANTIFICATION AND STATISTICAL ANALYSIS

RNA velocity metrics

When ground truth velocities are known (synthetic data), we compare estimated velocities by computing the cosine similarity on gene space and 50 principle components.

When ground truth velocities are not known, we use known cell-type transitions with the Cross-Boundary Directedness metric.[16,17] This score measures the velocity direction of cells on the boundary between two cell-types.

Additionally, we modify the score to be more robust to noisy boundaries between cell-types by only checking that the direction of the velocity of a cell is directed toward any cell of the expected cell-type, rather than any specific cell.

Our updated version quantifies the probability that a cell is likely to transition in the specified direction. Therefore, we can interpret scores above 0.5 to indicate a likely transition in that direction. Note that since this score depends on well-defined boundaries, noisy boundaries can lower the score.

The score is calculated as

$$\mathrm{CBDir}(A \to B, \text{cell } i) = \frac{1}{B \cap \mathcal{N}(i)} \sum_{j \in B \cap \mathcal{N}(i)} \delta[\cos(\mathbf{v}_i, \mathbf{s}_{ij})], \quad \text{(Equation 29)}$$

where $\mathbf{s}_{ij} = (\mathbf{s}_i - \mathbf{s}_j)/\mathrm{sign}(\mathbf{s}_i - \mathbf{s}_j)$ and $\mathcal{N}(i)$ is the neighborhood of cell i. Since the boundaries between cells can be noisy in high-dimensional gene space, this score is computed on a lower-dimensional space. In previous work this score was computed on a 2D UMAP embedding, here we compute this score on 50 principle components or the LatentVelo's latent space.

We also use the In-Cluster Coherence (ICCoh) metric[16,17] to evaluate the coherence of velocities within a cluster or cell-type. This score is computed per cell by the average cosine similarity of neighboring cells in the same cluster/celltype,

$$\mathrm{ICCoh}(A, \text{cell } i) = \frac{1}{A \cap \mathcal{N}(i)} \sum_{j \in A \cap \mathcal{N}(i)} \cos(\mathbf{v}_i, \mathbf{v}_j). \quad \text{(Equation 30)}$$

Similarly, we also use 50 principle components for this score. A similar consistency score was also used with scVelo[11] and DeepVelo.[18]

Note that while we have used 50 principle components to evaluate cell-type transitions because we can do a common benchmark between all methods, a natural embedding to use when evaluating cell-type transitions with LatentVelo is just the latent embedding inferred by the model.

Since the exact velocities in the Dyngen simulations are noisy, we average over 100 nearest neighbors when computing the gene space cosine similarity between estimated and exact velocities. Otherwise, the whiskers in Figure 6 for velocity cosine similarity extend the entire plot due to noise in the simulation. Separately plotting these "incorrectly" predicted cells shows their simulated velocity going opposite the direction of differentiation. For this reason, we compare with local averages of velocity.

Batch correction metrics

To evaluate batch correction we use the k-nearest-neighbor batch effect test (kBET) and integration local inverse Simpson's Index (iLISI) metrics.[30] We also use the cell-type local inverse Simpson's Index (cLISI) metric to evaluate biological cluster conservation,[30] ensuring that cell-type clusters are not erased by the batch correction. These metrics are computed with the *scib* package.[30]

kBET evaluates the batch composition of the nearest neighbors of a cell, which should match the overall batch composition for a particular cell-type for good batch correction. iLISI and cLISI measure the nearest-neighbor graph structure, evaluating batch mixing (iLISI) or cell-type separation (cLISI).

To evaluate batch correction of RNA velocity, we measure the cosine similarity of nearest neighbor cells in different batches.

Analyzing model fits and hyperparameters

Since the initialization, training, and selection of a training/test set for the model are all stochastic, we will not get the same trained model every time. Additionally, there are many hyperparameters that effect model fits. While choosing the model with the lowest loss value can eliminate some poor fits, there are cases where the model has a poor fit with a low loss value.

To address this issue, we analyze the unspliced vs. spliced plot of individual key genes. We expect the velocity of these genes should generally align with the flow of the data in the unspliced vs. spliced space. We fit a smoothing spline to the unspliced vs. spliced data to estimate the direction of the flow. We then compute the tangent line to these splines,

$$\mathbf{T}(s) = \frac{\left(v_s, \frac{df(s)}{ds}\right)}{\sqrt{v_{ss}^2 + \frac{df(s)^2}{ds}}}, \quad \text{(Equation 31)}$$

where $f(s)$ is the smoothing spline, and v_s is the spliced velocity. We then compute the cosine similarity between $\mathbf{T}$ and the unspliced-spliced space velocity (v_s, v_u). Since the spline does not specify the positive or negative direction, we compute the average absolute value of the cosine similarity. We only consider key genes where the R^2 score between the data and spline is low, to ensure that the spline is well fit. For example, this removes cycling genes that are poorly fit by the spline.

For velocities that align with the direction of the spline, scores will be near 1, misaligned velocities will have a lower score with a minimum of 0. We use this approach to evaluate model fits while varying model hyperparameters in Figure S6. While some fits we can discard due to a high loss value, for others we must do this gene-level analysis.