Statistics Project: RNA Velocity

Najwa Laabid

January 31, 2021

- ☐ Project overview
- ☐ Biological background
- ☒ Velocity modeling
- ☒ Post velocity
- ☐ Experiments
- ☐ Final report - Reviewed/Approved

Contents

1 Velocity Modeling

RNA velocity can be inferred by modeling the splicing kinetics using spliced and unspliced counts of genes from sc-RNA-seq data. Transcriptional dynamics can be divided to three stages: induction/repression, splicing, and degradation. We assume that every gene goes through these three stages during its transcription, with gene-specific corresponding rates $\alpha^{(k)}$, β, and γ. $\alpha^{(k)}$ would take on additional state-specific values to reflect the induction/repression (and intermediate stages) of genes signaling cellular differentiation.

If we assume these rates to be time-independent (save for $\alpha^{(k)}$ which is state-, therefore time-dependent), we get the gene-specific rate equations shown in (1), which capture the change of observed mRNA quantity over time:

$$\begin{aligned} \frac{du(t)}{dt} &= \alpha^{(k)}(t) - \beta u(t) \\ \frac{ds(t)}{dt} &= \beta u(t) - \gamma s(t) \end{aligned} \tag{1}$$

Velocity in this case is defined as the time derivative of spliced mRNA: $v(t) = \frac{ds(t)}{dt}$.

While there seems to be a consensus on the splicing kinetics model, different computational methods have been proposed for inferring the kinetic rates involved. This section covers the models proposed in scVelo, namely: steady-state model, dynamical model and stochastic model.

1.1 Steady-state Model

The steady-state model, introduced in [1], assumes that at the end of the induction/repression phase, the quantity of spliced mRNA would stabilize over (a small lapse of) time, i.e. RNA velocity $v(t) = \frac{ds(t)}{dt} = 0$. This observation turns the kinetics equations to a linear combination between unspliced and spliced mRNA:

$$u(t) = \frac{\gamma}{\beta}, s(t) = \gamma' s(t) \tag{1}$$

where γ' is the steady-state ratio of unspliced to spliced mRNA, indicating where mRNA synthesis and degradation is in balance.

In practice, this simplification requires induction/repression to last sufficiently long for the spliced/unspliced counts to reflect this phenomenon. If this is indeed true, we can get the data corresponding to this stage in the upper/lower quantiles of the counts for induction/repression respectively. By performing

linear regression on these quantiles, we can get an estimation of γ', and consider velocities to be deviations from the steady-state ratio:

$$v(t) = u_i - \gamma' s_i \tag{2}$$

γ' can be solved analytically using: $\gamma' = \frac{u^T s(t)}{||s||^2}$, where u and s are vectors of size-normalized unspliced and spliced counts for a particular gene from the extreme quantiles. An improvement on the model takes into account basal transcription through an offset o, leading to estimates of $\gamma' = \frac{cov(u,s)}{var(s)}$ and $o = \bar{u} - \bar{s}\gamma'$ where $\bar{u}$ and $\bar{s}$ are the means of u and s respectively.

This method avoids estimating absolute values of kinetic rates and resolving time assignments of the observations, while recovering decent estimates of the trajectory of cell development. Implicitly, however, the model assumes that all genes have a common splicing rate $\beta = 1$, since it disregards specific rate values. Furthermore, the assumption of an observable steady state is often violated in practice, particularly by transient cell populations. scVelo proposed a dynamical model to solve these issues.

1.2 Dynamical Model

In this section, we present the mathematics behind scVelo's dynamical model. We start by verifying the analytical solution presented in the paper to equation 1. The procedure follows the explanations presented in [2] and can be found in appendix A.

Next, we present the process followed to infer the value of the parameters of the equation, namely the transcriptional rates and the latent cell-specific time.

1.2.1 Parameter Inference

We want to estimate the parameters $\theta = (\alpha^{(k)}, \beta, \gamma)$ at different transcriptional states $k \in \{1, 0, ss_1, ss_0\}$. Another way to look at the problem is to estimate the phase trajectory (expressed by $\hat{x}(t) = (\hat{u}(t), \hat{s}(t))$) that best describes the observed quantities $u(t)$ and $s(t)$. This is done using maximum likelihood, with the following assumptions/definitions:

- **Residuals**: we consider the residuals of the observations $x(t)$ to the estimated trajectory $\hat{x}(t)$ as signed Euclidean distances (also known as Deming residuals): $e_i = sign(s_i - \hat{s}_{t_i}).||x_i^{obs} - x_{t_i}(\theta)||$. Laplacian residuals [1] are also an option.
- **Constant variance**: gene-specific σ constant across cells within one transcriptional state.
- **Idependence**: the observations are independent and identically distributed.

[1] Using least absolute (instead of squared) residuals.

- **Normality**: $e_i \ N(0, \sigma^2)$.

Under these conditions, the likelihood for a particular gene is [2]:

$$L(\theta) = \frac{1}{(\sqrt{2\pi}\sigma)^n} exp(-\frac{1}{2}\sum_{i}^{n} \frac{||x_i - x_{t_i}(\theta)||^2}{\sigma^2}) \tag{1}$$

We would like the find the values of θ that maximize the likelihood of the sample of x_is. To do so, we need to couple of each observation x_i with the corresponding state-dependent equations/rates. This requires estimating the latent-time of the cells that would put them in the right phase trajectory. To perform this double estimation, we can use Expectation-Maximization (EM) framework, which would iterate between finding the optimal parameters θ and the latent parameters of state and time.

In general, EM is used in partial-data problems, i.e. problems where we do not have access to all the data necessary for estimating the parameters. This is usually the case when we have latent variables, and are therefore unable to observe a crucial aspect of the phenomenon being modeled. In this case, EM offers a numerical approach to estimating both sets of parameters: the observable and un-observable. To do so, we assume that we have the values of one set (expectation) and derive the other (maximization), then switch the positions of the derived and assumed sets iteratively until we're satisfied with the values given to both.

For the problem at hands, we start by initializing the values of θ from estimates of the steady-state model as follows:

- $\beta = 1$, from the assumption of the steady-state model.
- $\gamma = \frac{u^T s}{||s||^2}$, from the equation of the steady-state model given in section 1.1.
- $k = 1$ for positive steady-state velocity (signaling induction state) and 0 for negative velocity (for repression state).
- $\alpha = 0$ for $k = 0$ and $max(s)$ for $k = 1$.

Then we derive the values of latent-time for each cell. This can be done using maximum-likelihood. However, to increase the computational speed, the authors proposed an approximation using an explicit formula, derived from a manipulation of the spliced counts equation. The formula and its deduction process can be found in Appendix C. This method is said to be less accurate than ML estimation, but the gain in speed-up justifies the shortcut. In the last iteration of EM, time is estimated optimally using likelihood.

[2]The equation given in the paper is taking a 1/nth of the original likelihood, as explained by the authors in. a Github issue [3].

In the maximization step, we consider the time values to be correct, and estimate θ that would maximize the likelihood given in 1. In this step, we use a derivative-free numerical method for estimation, known as downhill simplex method or Nelder-Mead method.

It is worth noting at this stage that the likelihood given in equation 1 serves to evaluate the goodness of fit of a single gene on the phase trajectory, which identifies putative drivers of the underlying process.

1.3 Stochastic Model

The stochastic model exploits the relationship between unspliced and spliced mRNA abundances as well as their *covariation*. To do so, we use the second-order moment of velocity. The procedure goes as follows.

First we consider transcription, splicing and degradation to be probabilistic events representing the change in u_t and s_t over a small time interval dt. The likelihood for every event is given below:

transcription: $P(u_{t+dt} = u_t + 1, s_{t+dt} = s_t) = \alpha dt$

splicing: $P(u_{t+dt} = u_t - 1, s_{t+dt} = s_t + 1) = \beta u_t dt$

degradation: $P(u_{t+dt} = u_t, s_{t+dt} = s_t - 1) = \gamma s_t dt$

One way to understand these probabilities is to consider three distinct pairs of changes of u_t and s_t, corresponding to the three biological phenomena/events. Each of these phenomena happens to be associated with a fixed rate, representing the change in quantity per second. By multiplying said rate by the interval dt, we end up with a probability of the event occuring within the new interval.

Next, we consider the uncentered moment of the bivariate distribution of u and s, $< u_t^l s_t^k >= E[u_t^l s_t^k]$. The time derivative of this moment in light of the probabilities is shown in equation 1.

$$\begin{aligned}\frac{d < u_t^l s_t^k >}{dt} =& < \alpha((u_t + 1)^l s_t^k - u_t^l s_t^k) > + < \beta u_t((u_t - 1)^l (s_t + 1)^k - u_t^l s_t^k) > \\ &+ < \gamma s_t(u_t^l (s_t - 1)^k - u_t^l s_t^k) >\end{aligned} \tag{1}$$

From equation 1 we can get differential equations describing the first-order and second-order of u_t and s_t as well as the interaction between them through $u_t s_t$. Said equations can be derived by giving specific values to l and k in equation 1.

More specifically, we consider the first and second-order equations of velocity as shown below, by taking $l = 0$ and $k = 1$ or $k = 2$ respectively:

$$\begin{aligned}\frac{d<s>}{dt} &= \beta u_t - \gamma <s_t> \\ \frac{d<s^2>}{dt} &= \beta <u_t> +2\beta <u_t s_t> +\gamma <s_t> -2\gamma <s_t^2>\end{aligned} \tag{2}$$

At the steady-state, we can set $\frac{d<s>}{dt} = 0$ and $\frac{d<s^2>}{dt} = 0$, leading to the following system of steady-state equations:

$$\begin{pmatrix} \langle u_t \rangle \\ \langle u_t \rangle + 2\langle u_t s_t \rangle \end{pmatrix} = \gamma' \begin{pmatrix} \langle s_t \rangle \\ 2\langle s_t^2 \rangle - \langle s_t \rangle \end{pmatrix} + \epsilon' \tag{3}$$

From here we can get an estimate for γ' involving the second-order moment of s_t and u_t. Taking $s' = \begin{pmatrix} \langle s_t \rangle \\ 2\langle s_t^2 \rangle - \langle s_t \rangle \end{pmatrix}$ and $u' = \begin{pmatrix} \langle u_t \rangle \\ \langle u_t \rangle + 2\langle u_t s_t \rangle \end{pmatrix}$, the steady state ratio γ' can be derived using generalized least squares as follows:

$$\gamma' = \left(s'^T \Omega^{-1} s'\right) s'^T \Omega^{-1} u' \tag{4}$$

This new estimate of γ' provides an accuracy rivaling with the dynamical model, at the reduced computational cost of the steady-state model. The paper also mentioned exploring the use of higher-order moments with the dynamical model as future work.

1.4 Prior Information for Insufficiently Observed Kinetics

It can happen that for a given gene only a small portion of the overall dynamics is disclosed at the end of the process. We say that the splicing kinetics are *insufficiently observed*. In plotting, this case appears as a straight line rather than a curve in the unspliced to spliced phase diagram. Velocity models are then challenged to 'guess' velocity vectors surrounding the line. There are two possible causes of this situation: 1) the gene's upregulation/downregulation only happens at the extremes of a developmental process, or 2) we only observe a small time frame of the underlying dynamical process (due to the process being extremely fast for instance).

From a mathematical perspective, the situation translates to an identifiability issue resulting in two local optima, which can only be solved with additional information. One way to provide this information is through a regularization term added to the negative log-likelihood of the dynamical model: $l(\theta)+\lambda R(t,\theta)$. The regularization term penalizes observations with a latent time assigned earlier than the root.

$$R(\theta) = \frac{1}{n}\sum_{i}^{n} 1_{\{t_i<t_j\}} ||x_i^{obs} - x_o^{obs}||^2 \tag{1}$$

We identify root cells from other informative genes in the data in scenario 1, or from experimental knowledge such as the first time point or a known progenitor population in scenario 2. The genes with partial kinetics can be detected by their R^2 value exceeding a given threshold, for example: 0.95.

2 Post-velocity

This section is dedicated to the computations involving velocity used in the plotting and study of cell trajectories.

2.1 Transition Probabilities

Assuming velocity to accurately capture the dynamics of cell differentiation, we can compute the probability of transitioning from cell i to cell j as the difference between the predicted change in the spliced mRNA of cell i (i.e., the velocity of cell i) and the actual change of gene expression necessary to reach cell j, $\sigma_{ij} = s_j - s_i$. Since the two entities of interest are vectors, we apply cosine difference to compare them, leading to equation (1):

$$\pi_{ij} = cos(\phi_{ij}, v_i) = \frac{\phi_{ij}^T v_i}{||\phi_i j||||v_i||} \tag{1}$$

This metric only measures difference in direction, and ranges from -1 (opposite direction) to 1 (identical direction), passing through 0 (orthogonal = most dissimilar). Computing this measure for multiple cells results in a similarity matrix known as the velocity graph. This graph is only computed for cells belonging to the same knn graph neighborhood.

Next we can apply an exponential kernel to the cosine correlations to get transition probabilities. Aggregated into a transition matrix, these probabilities give the Markov chain of the differentiation process. This matrix is also known as a transport map, and serves to get the descendant or ancestor (using transpose) population of a group of cells.

2.2 Gene-shared Latent Time

We aim to resolve cell-specific times to position the cells on their developmental timelines. This time is deduced from computing a shared time scale across genes. The following bullet points detail these steps:

1. **Normalize well-fitted genes**: for each cell, we consider the genes best-fitted in the previous velocity computation, with a likelihood ≥ 0.1. We normalize their latent-time to a common normalized time scale.
2. **Identify root cells**: the root cells correspond to the beginning of the differentiation process. We reveal them through stationary states μ^* satisfying: $\mu^* = \mu^* \tilde{\pi}^T$, which correspond to the left eigenvectors of $\tilde{\pi}^T$ with a value of 1.
3. **P-quantile of time increments**: for every root cell found earlier, we compute the p-quantile gene-time increments across all genes. The percentage p maximizes the correlation between the resulting latent time

course and its convolution across the local neighborhood of cells, thus increasing the robustness and consistency of the estimation. Working with a quantile accounts for the expected increasing in cell number (adapts for the non-uniform density of cells). Optimal values of p vary between 20% and 30%.

4. **Cell-specific time as a mean**: we obtain the latent time for cell i as the mean across all root cells: $t_i =< t_{i,o} >$. To increase robustness, we can regress the latent time against its neighborhood convolution, and replace inconsistent time points with their convolutions.

2.3 Visualizations

Embeddings are low-dimensional spaces where we can translate high-dimensional vectors [**?**]. The translation maintains connections between vectors in their original space (e.g., word embeddings preserving semantics) while allowing for visualizations and descriptive data analysis not possible in high dimensions. In cell data, embeddings allow plotting cell trajectories to gain a visual appreciation of cell movement and the overall structure of cell data [**?**].

Multiple solutions exist for dimension reduction. The most famous of all is t-distributed stochastic neighbor embedding (t-SNE), which is currently available in multiple implementations, including SK-Learn's TSNE function, openTSNE and MulticoreTSNE [**?**]. Recent developments in the field lead to newer, more efficient solutions, such as Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP), which was the method of choice in scVelo.

ScVelo's embedding makes use of the transition matrix π (see section **??**) to translate velocities onto the embedding. Considering every cell i is described by vector s_i, we can define the difference between the embeddings of two cells as:

$$\tilde{\delta_{ij}} = \frac{\tilde{s_j} - \tilde{s_i}}{||\tilde{s_j} - \tilde{s_i}||} \tag{1}$$

An embedding formula can be seen as the expected displacement with respect to the transition matrix:

$$v_i = \mathbb{E}[\delta_i] = \sum_{j \neq i} (\tilde{\pi}_{ij} - \frac{1}{n}) \delta_{ij} \tag{2}$$

where $\frac{1}{n}$ is an adjustement for the non-uniform density of points in the embedding.

2.4 Testing for Different Kinetics Regimes

Different kinetics' regimes appear due to difference in cell types and lineages [4]. Even when cells come from a homogeneous population, transcription pro-

cesses like alternative splicing or polyadenylation, or modulations degradation, can make a difference in the kinetic structure of the cells [4]. The likelihood based framework detects these regimes through a likelihood ratio test isolating clusters not sufficiently explained by a single model for the overall dynamics. The tests are carried out by fitting every cell type with an independent model, and evaluating if this substitution improves the likelihood significantly.

$$LR = -2ln\frac{sup_{\theta}L_{\theta}}{sup_{\theta'}L_{\theta'}} \tag{1}$$

with θ and θ' belong to different kinetic models. For efficiency reasons an orthogonal regression is considered as an alternative model. The cell type furthest away from the learned trajectory is considered for the independent fitting. Other cell types may be added to the new model if it improves the likelihood ratio further. The ratio can be tested for significance considering its χ^2.

2.5 Validation Metrics

The goal here is two-fold: evaluate the velocity vector field and the contribution of a selection of genes to the overall inferred dynamics.

The coherence can be measured through a *consistency score*, defined for every cell c_i as the mean correlation of its velocity v_i with velocities from neighboring cells: $c_i = \langle corr(v_i, v_j)\rangle_i$.

The contribution can be captured through a *reconstructability score*, defined as the median correlation of outgoing transitions from cell i to all cells it can potentially transition to. To get this metric, we compute a transition graph π including all genes and another graph π' including only the genes of interest. The score can be computed as shown in 1.

$$r = median_i corr(\pi_i, \pi'_i) \tag{1}$$

Appendices

Appendix A Analytical Solutions to the Kinetics Equations

This appendix presents a solution to the differential equations used in the dynamical model (see section ***).

A.1 Analytical solution to the equation of unspliced mRNA

We start by finding a solution to the equation shown below.

$$\frac{du(t)}{dt} = \alpha^{(k)}(t) - \beta u(t) \tag{1}$$

This is known as a linear first order differential equation. It is one of the simplest differential equations that has a closed form solution. What follows is the detailed version of the standard process of solving such equations. The goal of the process is to derive an expression for $u(t)$ as a function of t. This means, informally, that we need to 'remove' the derivative term from the equation.

We start by reordering the terms of the equation to the form shown below:

$$\frac{du(t)}{dt} + \beta u(t) = \alpha^{(k)}(t) \tag{2}$$

We define a function $A(t)$, s.t.: $A(t)\beta = A'(t)$

We multiply both sides of (2) by $A(t)$:

$$A(t)\frac{du(t)}{dt} + A(t)\beta u(t) = A(t)\alpha^{(k)}(t) \tag{3}$$

We replace $A(t)\beta$ by $A'(t)$:

$$A(t)\frac{du(t)}{dt} + A'(t)u(t) = A(t)\alpha^{(k)}(t) \tag{4}$$

At this point the left-side of the equation appears to be the derivative of the product of two functions, $u(t)$ and $A(t)$. We substitute accordingly.

$$(A(t)u(t))' = A(t)\alpha^{(k)}(t) \tag{5}$$

Now we integrate both sides of the equation:

$$\begin{aligned} \int (A(t)u(t))'dt &= \int A(t)\alpha^{(k)}(t)dt \\ A(t)u(t) + c &= \int A(t)\alpha^{(k)}(t)dt \\ A(t)u(t) &= \int A(t)\alpha^{(k)}(t)dt - c \end{aligned} \tag{6}$$

Since c is an unknown constant, it can absorb the minus sign in front of it (both writings of c will lead to the same final equation, with values for c of opposite signs).

$$\begin{aligned} A(t)u(t) &= \int A(t)\alpha^{(k)}(t)dt + c \\ u(t) &= \frac{\int A(t)\alpha^{(k)}(t)dt + c}{A(t)} \end{aligned} \tag{7}$$

We arrive at the final solution of the equation in its general form. Given a function $A(t)$ satisfying the assumptions given before, we can find a closed-form expression of $u(t)$ using equation (7). Next we try to find a specific expression for function $A(t)$.

We know that:

$$\begin{aligned} A(t)\beta &= A'(t) \\ \frac{A'(t)}{A(t)} &= \beta \end{aligned} \tag{8}$$

The left-side of the equation looks like the derivative of a compound logarithmic function. We substitute accordingly and integrate both sides.

$$\begin{aligned} ln'(A(t)) &= \beta \\ \int ln'(A(t))dt &= \int \beta dt \\ ln(A(t)) + p &= \beta t + m \\ ln(A(t)) &= \beta t + p \end{aligned} \tag{9}$$

Note that we 'merged' the integration constants p and m in one constant p. Here again the numerical expressions of the solution before and after the modification will be equivalent. Next we take the exponential of both sides to find the final

expression of $A(t)$. We introduce a simplification of constants once more by replacing e^p with k.

$$\begin{aligned} ln(A(t)) &= \beta t + p \\ A(t) &= e^{\beta t + p} \\ A(t) &= e^{\beta t} e^p \\ A(t) &= k e^{\beta t} \end{aligned} \tag{10}$$

Now we replace the final expression of $A(t)$ in equation (7) to get:

$$u(t) = \frac{\int k e^{\beta t} \alpha^{(k)}(t) dt + c}{k e^{\beta t}} \tag{11}$$

We note here that the equation has two constants, k and c. To avoid finding a numerical value for both, we merge them into one constant as shown below. We also rewrite the equation to get rid of the fraction.

$$\begin{aligned} u(t) &= \frac{\int e^{\beta t} \alpha^{(k)}(t) dt + \frac{c}{k}}{e^{\beta t}} \\ u(t) &= \frac{\int e^{\beta t} \alpha^{(k)}(t) dt + c}{e^{\beta t}} \\ u(t) &= e^{-\beta t} \int e^{\beta t} \alpha^{(k)}(t) dt + e^{-\beta t} c \end{aligned} \tag{12}$$

What is left at this stage is to solve the integral $\int e^{\beta t} \alpha^{(k)}(t) dt$. Here it is important to note that scVelo's dynamical model assumes that $\alpha^{(k)}(t)$ takes constant values at different states k. We can therefore consider it a constant for the purposes of this integration.

$$\begin{aligned} \int e^{\beta t} \alpha^{(k)}(t) dt &= \alpha^{(k)} \int e^{\beta t} dt \\ &= \frac{\alpha^{(k)}}{\beta} (e^{\beta t} + p) \\ &= \frac{\alpha^{(k)}}{\beta} e^{\beta t} + \frac{\alpha^{(k)}}{\beta} p \\ &= \frac{\alpha^{(k)}}{\beta} e^{\beta t} + p \end{aligned} \tag{13}$$

We plug the result in equation (12).

$$
\begin{aligned}
u(t) &= e^{-\beta t}(\frac{\alpha^{(k)}}{\beta}e^{\beta t} + p) + e^{-\beta t}c \\
&= \frac{\alpha^{(k)}}{\beta} + pe^{-\beta t} + ce^{-\beta t} \\
&= \frac{\alpha^{(k)}}{\beta} + e^{-\beta t}(p + c) \\
&= \frac{\alpha^{(k)}}{\beta} + e^{-\beta t}c \\
&= ce^{-\beta t} + \frac{\alpha^{(k)}}{\beta}
\end{aligned} \tag{14}
$$

Using the initial conditions of the model: $u(0) = u_0$, we can find the value of c as follows:

$$
\begin{aligned}
u(0) &= ce^{-\beta 0} + \frac{\alpha^{(k)}}{\beta} \\
u_0 &= c + \frac{\alpha^{(k)}}{\beta} \\
c &= u_0 - \frac{\alpha^{(k)}}{\beta}
\end{aligned} \tag{15}
$$

Substituting in (14), we get the final solution given in the paper:

$$
\begin{aligned}
u(t) &= (u_0 - \frac{\alpha^{(k)}}{\beta})e^{-\beta t} + \frac{\alpha^{(k)}}{\beta} \\
&= u_0e^{-\beta t} - \frac{\alpha^{(k)}}{\beta}e^{-\beta t} + \frac{\alpha^{(k)}}{\beta} \\
&= u_0e^{-\beta t} - \frac{\alpha^{(k)}}{\beta}e^{-\beta t} + \frac{\alpha^{(k)}}{\beta} \\
&= u_0e^{-\beta t} + \frac{\alpha^{(k)}}{\beta}e^{-\beta t} + \frac{\alpha^{(k)}}{\beta} \\
&= u_0e^{-\beta t} + \frac{\alpha^{(k)}}{\beta}(1 - e^{-\beta t})
\end{aligned} \tag{16}
$$

A.2 Analytical solution to the equation of spliced mRNA

This time we try to find a formula for estimating $s(t)$ knowing that:

$$\frac{ds(t)}{dt} = \beta u(t) - \gamma s(t) \tag{1}$$

We plug in the expression of $u(t)$ found earlier:

$$\begin{aligned} \frac{ds(t)}{dt} &= \beta u_o e^{-\beta t} + \alpha^{(k)}(1 - e^{-\beta t}) - \gamma s(t) \\ \frac{ds(t)}{dt} + \gamma s(t) &= \beta u_o e^{-\beta t} + \alpha^{(k)}(1 - e^{-\beta t}) \end{aligned} \tag{2}$$

This time we skip the initial steps and start from the formula for the general solution of the first-order linear differential equation. Let us consider first the right-side of the equation to be equal to a function $g(t)$, $\frac{ds(t)}{dt} + \gamma s(t) = g(t)$.

$$\begin{aligned} s(t) &= \frac{\int A(t)g(t)dt + c}{A(t)}, \\ s.t., A(t) &= e^{\int \gamma dt} = pe^{\gamma t} \end{aligned} \tag{3}$$

Now we substitute $g(t)$ and $A(t)$ for their values and simplify the constants:

$$\begin{aligned} s(t) &= \frac{\int pe^{\gamma t}(\beta u_o e^{-\beta t} + \alpha^{(k)}(1 - e^{-\beta t}))dt + c}{pe^{\gamma t}} \\ &= \frac{\int e^{\gamma t}(\beta u_o e^{-\beta t} + \alpha^{(k)}(1 - e^{-\beta t}))dt + c}{e^{\gamma t}} \\ &= e^{-\gamma t} \int e^{\gamma t}(\beta u_o e^{-\beta t} + \alpha^{(k)}(1 - e^{-\beta t}))dt + e^{-\gamma t}c \end{aligned} \tag{4}$$

Solving the integral gives:

$$\begin{aligned} \int e^{\gamma t}(\beta u_o e^{-\beta t} + \alpha^{(k)}(1 - e^{-\beta t}))dt &= \int \beta u_o e^{(\gamma-\beta)t} + e^{\gamma t}\alpha^{(k)} - \alpha^{(k)}e^{(\gamma-\beta)t}dt \\ &= \frac{\beta u_o}{\gamma - \beta}e^{(\gamma-\beta)t} + \frac{\alpha^{(k)}}{\gamma}e^{\gamma t} - \frac{\alpha^{(k)}}{\gamma - \beta}e^{(\gamma-\beta)t} + m \end{aligned} \tag{5}$$

We substitute in (4):

$$
\begin{aligned}
s(t) &= e^{-\gamma t}(\frac{\beta u_o}{\gamma-\beta}e^{(\gamma-\beta)t} + \frac{\alpha^{(k)}}{\gamma}e^{\gamma t} - \frac{\alpha^{(k)}}{\gamma-\beta}e^{(\gamma-\beta)t} + m) + e^{-\gamma t}c \\
&= \frac{\beta u_o}{\gamma-\beta}e^{-\beta t} + \frac{\alpha^{(k)}}{\gamma} - \frac{\alpha^{(k)}}{\gamma-\beta}e^{-\beta t} + me^{-\gamma t} + e^{-\gamma t}c \\
&= \frac{\beta u_o}{\gamma-\beta}e^{-\beta t} + \frac{\alpha^{(k)}}{\gamma} - \frac{\alpha^{(k)}}{\gamma-\beta}e^{-\beta t} + ce^{-\gamma t} \\
&= ce^{-\gamma t} + \frac{\alpha^{(k)}}{\gamma} + e^{-\beta t}(\frac{\beta u_0 - \alpha^{(k)}}{\gamma-\beta})
\end{aligned}
\tag{6}
$$

We find the value of c, using the initial condition of $s(t)$: $s(0) = s_0$:

$$
\begin{aligned}
s(0) &= ce^{-\gamma 0} + \frac{\alpha^{(k)}}{\gamma} + e^{-\beta 0}(\frac{\beta u_0 - \alpha^{(k)}}{\gamma-\beta}) \\
s(0) &= c + \frac{\alpha^{(k)}}{\gamma} + \frac{\beta u_0 - \alpha^{(k)}}{\gamma-\beta} \\
c &= s_0 - \frac{\alpha^{(k)}}{\gamma} - \frac{\beta u_0 - \alpha^{(k)}}{\gamma-\beta}
\end{aligned}
\tag{7}
$$

We plug in the value of c in (6) to get:

$$
\begin{aligned}
s(t) &= (s_0 - \frac{\alpha^{(k)}}{\gamma} - \frac{\beta u_0 - \alpha^{(k)}}{\gamma-\beta})e^{-\gamma t} + \frac{\alpha^{(k)}}{\gamma} + e^{-\beta t}(\frac{\beta u_0 - \alpha^{(k)}}{\gamma-\beta}) \\
&= s_0 e^{-\gamma t} + \frac{\alpha^{(k)}}{\gamma}(1 - e^{-\gamma t}) + (\frac{\alpha^{(k)} - \beta u_0}{\gamma-\beta})(e^{-\gamma t} - e^{-\beta t})
\end{aligned}
\tag{8}
$$

We note here that t in the equations is valid within one conditional state, i.e. t $= \tau - \tau_0^{(k)}$ where τ is the absolute time of the process.

Appendix B Setups for Experiments

B.1 Setting up a remote server

B.1.1 `SSH` on Windows (optional)

While **ssh** is a standard networking tool included on most UNIX-based operating systems (Linux, MacOs, etc), it was not available on Windows until Windows 10. The steps below explain how to activate the existing installation. For older versions of Windows, suggestions for third-party tools are given afterwards.

To enable **ssh** on windows, open a Powershell as **Administrator** and run the following:

- To show the name and state of OpenSSH on your system:

```
Get-WindowsCapability -Online | ? Name -like 'OpenSSH
    ↪ *'
```

- To add the **ssh** server component:

```
Add-WindowsCapability -Online -Name OpenSSH.Server
    ↪ ~~~~0.0.1.0
```

- To start the server after successfully enabling it:

```
Start-Service sshd
```

- The server will start silently. To check that it is indeed running, you can use:

```
Get-Service sshd
```

- You can have the **ssh** server (**sshd**) start on boot by running:

```
Set-Service -Name sshd -StartupType 'Automatic'
```

For additional information on security settings for the server, see [5].

After starting the server, close the admin Powershell and open a regular terminal window to run the **ssh** commands mentioned later in this guide.

If you have an older Windows version, you can use third-party tools like MobaXTerm [6], Git tools [7], and PuTTY [8].

B.1.2 Running a Jupyter notebook on `cs0`

The idea of this work is to run a 'browser-less' notebook on a remote server (e.g. **cs0**), and forward the interface to your local machine. The steps given below should help you achieve that, following the tutorial proposed in [9]. It is

assumed that Jupyter environment is available on your local machine and the remote server of interest. [3]

The set-up proceeds in two steps: (1) firing a notebook on the remote server and (2) forwarding the port where the notebook is running on the remote server to your local machine. This guide assumes no familiarity with networking concepts and/or language, such that some of its explanations might appear trivial to experienced users.

Starting a notebook on `cs0` server First we need to access the server. It is assumed that you already have a user in both `cs` and `cs0` servers (you can request access from designated faculty members if that's not the case).

- Access `cs.uef.fi` from your local machine by running:

```
$ ssh username@cs.uef.fi
```

  Enter your password when prompted. If all goes well, you should be on a terminal shell in the remote server.

- From `cs` server, `ssh` to the `cs0` server using the same command and the corresponding login credentials:

```
$ ssh username0@cs0
```

- Now in the `cs0` server, launch a Jupyter notebook at a specific port and without browser access:

```
$ jupyter notebook --no-browser --port=XXXX
```

  The default port is 8888, but any other (suitable) port should do. .

- At this stage, you should get a url with a numerical token to access the notebook from a browser. An example url can look like this:

```
http://localhost:XXXX/?token=#####
```

  Save the token somewhere for further use.

Port forwarding for `cs` and `cs0` Now we want to forward the traffic on port `XXXX` in `cs0` to a specific port on our local machine. [4]. Since accessing `cs0` is only possible through the `cs` server, you need to forward the port from `cs0` to `cs`, then from `cs` to your local machine.

- From a new terminal tab, login to `cs` and run the command:

[3] At the time of writing this tutorial, Jupyter is installed in `cs0` for all clients.
[4] This process is called *port forwarding*. You can read more about it in [10]

```
$ ssh -4 -NfL localhost:YYYY:localhost:XXXX
    ↪ username0@cs0
```

where: `-N` suppresses the execution of a remote command, `-f` requests the ssh command to go to the background before execution, `-L` requires an input in the form of local_socket:remote_socket, and -4 forces `ssh` to use IPv4 [5].

If the command is successful, you will not get any output in the terminal.

- From your local machine, run the same command with the corresponding parameters:

```
$ ssh -4 -NfL localhost:ZZZZ:localhost:XXXX username@cs
    ↪ .uef.fi
```

- Now you can access the notebook interface at:

```
http://localhost:ZZZZ/tree
```

Note the use of the port specified for your **local** machine in the last port forwarding command. At this stage, Jupyter will ask you to provide the token you generated in the remote machine earlier. Once you provide that, you should get an the usual Jupyter notebook interface.

B.1.3 Troubleshooting

Below are some common issues that might happen in this process along with terminal commands to investigate them.

Choosing a port You might want to use an explicit port at any stage of this setup. In this case, you need to know which ports are available for use, using the command:

```
$ netstat -lnptu
```

where:

`-n/--numeric`	don't resolve service names
`-l/--listening`	display listening sockets
`-t/--tcp`	display only TCP sockets
`-u/--udp`	display only UDP sockets

[5] `ssh` might try to resolve localhost using an IPv6 address format and fail. I am not sure what triggers this behavior, but using this flag seems to solve the issue [11].

Busy port This can happen on any of the machines involved in this setup. It is likely to occur when the connection to the notebook is lost somehow, but the tunnel set up at that particular machine (`cs` for instance) is still running.

One way to solve the issue is to see what process is using the port of interest and kill it explicitly. To get the process id use:

```
$ lsof -i:XXXX
```

with `XXXX` being your port of interest. You can run the command in `sudo` mode on your local machine to include processes owned by users other than yourself.

Once you have the process ID, you can kill the process using:

```
$ kill process_id
```

Trouble with SSH Sometimes the connection is not established for unclear reasons. In this case, it is helpful to add the flag '`-v`' to `ssh` connection commands to get a verbose output for debugging purposes.

B.1.4 Advanced set-up

This section presents a few advanced tricks to save some time when connecting to the server, and ease managing open terminal tabs on the machines involved.

Screen Multiplexing The goal is to multiplex a terminal window to either show multiple windows on the same screen, or hide some from your screen without compromising their content/execution (see [12] for more information). Once installed (through `apt-get` for instance), you can run it by typing `tmux` on any linux terminal.

'config' File Servers' domain names are a mouthful. Instead of typing, `username@cs.uef.fi` every time you want to access the server, you can define an alias for your connection in a `config` file under `ssh` directory. A possible configuration for one server can look like this:

```
Host cs
    User username
    Hostname cs.uef.fi
```

Now to log in to `cs.uef.fi`, you can simply type `ssh cs`.

You can also define the configuration of `cs0` to include `cs` as a proxy server.

```
Host cs0
    User username
    Hostname cs0
    ProxyJump cs
```

This way, whenever you try to connect to `cs0` from your local machine, the computer will connect to `cs` automatically before. Finally, if you're using a MacBook, you can add these lines to save your credentials to Mac's key chain:

```
Host *
    UseKeychain yes
```

Overall, my `config` file looks like this:

```
Host *
    UseKeychain yes

Host cs
    User username
    Hostname cs.uef.fi

Host cs0
    User username
    Hostname cs0
    ProxyJump cs
```

Next we will see how to use `ssh` keys to skip server login, so that `ssh cs0` is all you need to type to access `cs0`.

SSH key if you don't want to be asked to provide your username/password each time you log in to `cs`/`cs0`, you can generate an ssh key locally and send it to both servers to fast-forward your login [13]. To do so, follow the steps below:

- **Generate an ssh key** on your local machine using the command:

  ```
  $ ssh-keygen -t rsa
  ```

  If the command is successful, you should see two new files, `id_rsa` and `id_rsa.pub`, appear under the hidden '`.ssh`' folder in your home directory. [6]

- **Copy ssh key to the sever of interest**, which in this case would be `cs`. To do so, run the command:

  ```
  $ ssh-copy-id username@cs.uef.fi
  ```

- **Try to log in** using the command `ssh cs0` from your local machine. You should be able to reach the server without any additional input or intermediate steps.

[6] To access the ssh folder on a mac, you can use `open` followed by `open .ssh`.

Coming soon - bash functions This section will be dedicated to showing how to define bash functions to set the port forwarding and start Jupyter notebook with shorter commands.

References

[1] G. La Manno, R. Soldatov, A. Zeisel, E. Braun, H. Hochgerner, V. Petukhov, K. Lidschreiber, M. E. Kastriti, P. Lönnerberg, A. Furlan, J. Fan, L. E. Borm, Z. Liu, D. van Bruggen, J. Guo, X. He, R. Barker, E. Sundström, G. Castelo-Branco, P. Cramer, I. Adameyko, S. Linnarsson, and P. V. Kharchenko, "RNA velocity of single cells," *Nature*, vol. 560, no. 7719, pp. 494–498, 2018.

[2] P. Dawkins, "Linear differential equations," Nov 2020.

[3] N. Laabid, "Likelihood equation in paper (eq. (7))," September 2020.

[4] "Generalizing RNA velocity to transient cell states through dynamical modeling," *bioRxiv*, p. 820936, 2019.

[5] D. Kinghorn, "How to use ssh client and server on windows 10," May 2019.

[6] "Mobaxterm." Avaialble at `https://mobaxterm.mobatek.net/`.

[7] "Git." Avaialble at `https://git-scm.com/download/win`.

[8] "Putty." Avaialble at `https://www.putty.org/`.

[9] L. Miranda, "Running a jupyter notebook from a remote server," January 2018.

[10] Wikipedia contributors, "Port forwarding — Wikipedia, the free encyclopedia," 2020. [Online; accessed 24-September-2020].

[11] F. Boender, "Ssh port forwarding: bind: Cannot assign requested address," September 2014.

[12] D. Mckay, "How to use tmux on linux (and why it's better than screen)," May 2020.

[13] E. Sverdlov, "How to set up ssh keys," June 2012.