diffGEK: Differential Gene Expression Kinetics

Notes on use

Introduction

diffGEK is a method designed to uncover differences in transcriptional kinetics in single cell differentiation trajectories, provided that there are at least two different biological conditions. By transcriptional kinetics we mean the rates of transcription, splicing and degradation, as classically defined by established RNA velocity tools such as (La Manno et al., 2018; Bergen et al., 2020). As in the cited papers, we aim at inferring the kinetics based on the following ODE system:

$$\begin{aligned} \dot{U}_g(t) &= \alpha_g(t) - \beta_g(t)\, U_g(t) \\ \dot{S}_g(t) &= \beta_g(t)\, U_g(t) - \gamma_g(t)\, S_g(t) \end{aligned} \tag{1}$$

where $U_g(t)$ and $S_g(t)$ are the number of unspliced and spliced counts for gene g at pseudotime t, respectively; $\alpha_g(t)$, $\beta_g(t)$ and $\gamma_g(t)$ are the kinetic parameters (transcription, splicing and degradation rate respectively) for gene g at pseudotime t. Compared to the other papers. however, we modelled the kinetic parameters in a novel way. First, the parameters are gene specific. Second, they are a function of the cellular state, i.e., of the pseudotime. Since this would imply a big number of unknown variables to estimate, we further assumed that the parameters can be modelled with natural cubic splines.

Once the kinetic rates are computed, we need a procedure to decide whether they are differential or not among the two biological conditions over the whole trajectory of interest. To this aim, we consider 8 possible models, M1-M8, as in the corresponding Table 1.

For each gene, we need to fit all 8 models and then rank the models to decide which one fits the data better, relying at the same time on the smallest possible number of parameters (k in the table). This is achieved upon comparing the Aic information criterion index. Once the model is picked, we can conclude that the corresponding genes has differential kinetics or not, and which of the three kinetics are different.

	α	β	γ	k
M1	same	same	same	32
M2	any	same	same	40
M3	same	any	same	40
M4	same	same	any	40
M5	any	any	same	48
M6	any	same	any	48
M7	same	any	any	48
M8	any	any	any	56

Table 1: Schematic of the eight possible models used to fit the data. ”same” means that the corresponding rate is forced to be the same across conditions. ”any” means that it can be different. α: transcription rate; β: splicing rate; γ: degradation rate; k: number of model parameters (eight spline nodes per rate, four initial conditions and four estimated errors).

Requirements

To obtain unspliced counts, users need to run specific tools such as velocyto (La Manno et al., 2018).

For our paper, we used cellRank (Lange et al., 2022) to select trajectories, but the trajectory can be selected with any tool.

To prepare the input for the MATLAB pipeline, diffGEK requires a pre-analysis performed via scVelo (Bergen et al., 2020), that imputes the spliced and unspliced counts.

After the input has been prepared, the MATLAB pipeline for fitting the model starts. Throughout the pipeline, we used the same scheme and similar codes as in Pseudodynamics (Fischer et al., 2019), and thus diffGEK requires the same software as these tools: AMICI and PESTO. AMICI solves the models in the background with c++, and thus requires a c++ compiler.

Note that diffGEK, like Pseudodynamics, is a collection of scripts and not a tool to install. The scripts have to be adapted to the specific experimental design, and the model executable needs to be recompiled on the machine used to perform the analysis.

Prepare input

As mentioned above, diffGEK works on a single cell differentiation trajectory; this could be for example stem cells producing a certain lineage. It also relies on spliced and uspliced count information, which the user has to compute prior to the analysis (usually with the velocyto tool, (La Manno et al., 2018)).

The first step is to create a single cell landscape and then subset the trajectory of interest. In our paper, we either took an erythropoiesis trajectory as computed by the authors of the respective paper (Isobe et al., 2023), or computed the myelopoiesis trajectory of a published dataset (Liu et al., 2021) by means of the CellRank tool (Lange et al., 2022). We showed how we did it in the notebook `myelopoiesis/run_models/prepare_landscape.ipynb`. Note, however, that the landscape creation/trajectory selection can be achieved with any tool of preference.

Next, we need to create the input for diffGEK, which will be achieved in two steps. The first consist in creating a count matrix for each condition and for the spliced/unspliced counts separately. The tables, which follow the scanpy convention cells x genes, also have a column for pseudotime. In the first step, we rely on the scVelo framework to create imputed counts to feed into the model. This step is necessary because the unspliced counts are very noisy. The foundamental steps are:

- load an anndata object containing row counts, a UMAP landscape, and the layers "spliced" and "unspliced"
- run `scv.pp.filter_and_normalize` to filter low quality genes, lognormalize the layers and select HVGs. We recommend to keep the number of genes low, max 2000, because noisy genes (particularly noisy unspliced counts) affect the results at both the imputation and the later fitting step.
- run `scv.pp.moments` to input counts.
- save a dataframe per each group (spliced/unspliced, WT/mutant) and the list of genes used.

We showed how to do it in the repo file:
`run_models/prepare_input.ipynb` (from now on we refer to the codes for the erythropoiesis dataset, but the myelopoisis folder is structured in exactly the same way). Note, however, that in the paper analysis we added an extra set of genes taken from tradSeq for the

purpose of comparison with the HVGs. The user doesn't need to add other genes, unless they want to.

For the second step, we switch to MATLAB, because the fitting procedure will require MATLAB codes. The code to run is:
`run_models/create_input.m`. This code creates a MATLAB object (`gene_name.mat`) per each gene, containing the input for the fit (pooled counts at specific pseudotime intervals, which can be modified from the hardcoded ones, see the following section). The input is ready.

Fitting procedure

We now need to fit each gene with each of the 8 models explained above. The first thing to do is to create executables that will generate the model predictions. To do so, we need to first have a look at the files of the type: `model_building/kinetics_modeln_syms.m`, where $n = 1 - 8$. These files contain all the information on how the ODE model is structured. Particularly:

- p is the vector of parameters: 8 for the transcription rate for the WT condition (corresponding to the splicing nodes); 8 for the splicing rate for the WT condition; 8 for the degradation rate for the WT condition; 8 for the transcription rate for the mutant condition if different from WT in the corresponding model; 8 for the splicing rate for the mutant condition if different from WT in the corresponding model; 8 for the degradation rate for for the mutant condition if different from WT in the corresponding model; 4 for the initial conditions; 4 for the estimated errors. The number of spline nodes can be any, but it has to be manually modified in all these files,
- t_all is the set of time intervals at which to compute the mean spliced/unspliced values. It can be changed by the user (both the number of intervals, and the length, which can be different per each interval), but it has to be consistent with the above mentioned `create_input.m` code.
- splines calculation; no need to change anything.
- ODE system; no need to change anything.

To compile such files, use the provided `kinetics_wrap.m` function. This will create 8 more auxiliary files (no need to change anything) and 8 `.mex` executables.

We now need to run the fitting procedure. This is by far the most computationally intense step. Each gene has to be run independently, and requires little memory, but many genes and many models have to be run; for this reason we recommend to parallelize the analysis as much as possible. For each gene, we recommend to start from 200-300 initial conditions, which translates in a few minutes to finish one gene per each model. Note that the hardcoded version uses 3000 initial conditions because we wanted to be sure to found the global minimum, but we realised it was not necessary a posteriori. To tune this parameter, modify the part: `optionsMultistart.n_starts = 3000` in the files:
`run_models/main_modeln.m`, $n = 1 - 8$.

The fit maximizes a likelihood function that is provided in the auxiliary files:
`run_models/llKin_model1.m`.

Users don't need to change these files unless they want to change the likelihood function, or they have changed any structural parameter in the
`model_building/kinetics_modeln_syms.m` files. Note that if you change the likelihood you will also need to change the gradient (the derivative of the likelihood function with respect to the parameters).

After the fitting procedure is done, we will have 8 * number of genes files in the folder data_output. Each file contains the parameters and the loglikelihood values corresponding to each trial, but sorted in such a way that the first values correspond to the best possible model (maximum likelihood).

Analyse results

We can now perform the analysis the results, which is separated in 4 steps for convenience.

- The file `stats/c1_analyse_genes.m` collects the best fit parameters and computes the likelihood as weighted by the regularization term (as previously estimated by the simulations procedure; see section below).

- The file `stats/c2_write_aic.m` computes the corrected Aikaike information criterion index (Aic) for each model.

- The file `erythropoiesis/stats/c3_write_statistic_aic.m` picks the best model (smaller Aic), and writes a recap file for all genes together on the file `statistic_aic.txt`. The analysis could also finish here, as the classification of the

genes has been performed. The file provides a list with two columns: the first columns is a sequence 1 - number of genes, and the second column shown a number from 1 to 8, representing the best corresponding model. If the user wants to use our codes to create some summer statistic of the selected models, they can use the codes present in the folder `create_figures`.

- However, if one wants to create plots to show the goodness of the fits and the genes' trend along pseudotime, the file `stats/c4_write_parameters.m` writes a file where all the estimated parameters are written in the same file: `fit_parameters.txt`.

'Samples' mode

In case of matched replicates, diffGEK can also be run in 'samples' mode.

In this cases, each replicate is treated as a different dataset, but the parameters have to be shared among the replicates. This analysis showed, in our simulations, to spot more differential genes, but it also has more parameters (more initial conditions) and the convergence is slower.

The scheme to follow to run the analysis in sample mode can be found in the folder: `samples_mode`. Note, however,that the provided code are hardcoded for the number of samples corresponding to three. For any other number of samples, the codes have to be changed and recompiled. In principle, also the two regularization parameters (see the following section) have to be re-optimized for each number of replicates (we have only tested n = 3, as in our dataset).

Simulations

In our fitting procedure we leverage two regularization parameters: ρ and μ. The function of ρ is to keep the splines less wobbly; μ regularizes the estimated error. We estimated these parameters via the codes present in the folder `in_silico`. The scheme is similar to that used for the real data datsets, but with a few extra features.

- Since these are simulations, we need to create the ground truth. To this aim, we run the codes in the folder `generate_data`. Each code simulates 100 genes, for a total of 800 simulations, for each of which a `.mat` object is stored, that keeps track of the ground truth parameters.

- Then each gene is refitted with each model, for a total of 64000 fits for each regularization parameter.
- Similarly, the folder `stats` contains the files to produce the stats for each combination of the regularization parameters. From the comparison of these results to the ground truth, the best regularization parameters are picked.
- An extra file, `plot_parameters_compare_ground_truth.m`, is provided to plot the simulations and compare the estimated parameters compared to the ground truth.

References

Bergen, V., Lange, M., Peidli, S., Wolf, F. and Theis, F. (2020). Generalizing RNA velocity to transient cell states through dynamical modeling. Nature Biotechnology *38*, 1–7.

Fischer, D. S., Fiedler, A. K., Kernfeld, E. M., Genga, R. M. J., Bastidas-Ponce, A., Bakhti, M., Lickert, H., Hasenauer, J., Maehr, R. and Theis, F. J. (2019). Inferring population dynamics from single-cell RNA-sequencing time series data. Nature Biotechnology *37*, 461–468.

Isobe, T., Kucinski, I., Barile, M., Wang, X., Hannah, R., Bastos, H. P., Chabra, S., Vijayabaskar, M., Sturgess, K. H., Williams, M. J., Giotopoulos, G., Marando, L., Li, J., Rak, J., Gozdecka, M., Prins, D., Shepherd, M. S., Watcham, S., Green, A. R., Kent, D. G., Vassiliou, G. S., Huntly, B. J., Wilson, N. K. and Göttgens, B. (2023). Preleukemic single-cell landscapes reveal mutation-specific mechanisms and gene programs predictive of AML patient outcomes. Cell Genomics , 100426.

La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., Lidschreiber, K., Kastriti, M. E., Lönnerberg, P., Furlan, A., Fan, J., Borm, L. E., Liu, Z., van Bruggen, D., Guo, J., He, X., Barker, R., Sundström, E., Castelo-Branco, G., Cramer, P., Adameyko, I., Linnarsson, S. and Kharchenko, P. V. (2018). RNA velocity of single cells. Nature *560*, 494–498.

Lange, M., Bergen, V., Klein, M., Setty, M., Reuter, B., Bakhti, M., Lickert, H., Ansari, M., Schniering, J., Schiller, H. B., Pe'er, D. and Theis, F. J. (2022). CellRank for directed single-cell fate mapping. Nature Methods *19*, 159–170.

Liu, Y., Gu, Z., Cao, H., Kaphle, P., Lyu, J., Zhang, Y., Hu, W., Chung, S. S., Dickerson, K. E. and Xu, J. (2021). Convergence of oncogenic cooperation at single-cell and single-gene levels drives leukemic transformation. Nature Communications *12*.