Approach Description

Metadata collection

The first step of our pipeline is to collect and generate all the necessary metadata. We extract the state ('progenitor', 'effector', 'terminal exhausted', 'cycling', 'other') and condition (unperturbed or knocked-out gene) columns from the data provided. We then calculate the vector of cell state proportions for each knocked-out gene and unperturbed condition of our training dataset.

Deseq analysis

We performed differential expression analysis, using PyDeseq2 [1], by comparing the original cell counts of each state, e.i. progenitor vs effector, using the complete gene dataset. This analysis provided us with 679 unique genes with 2-log fold change between the states. We reduce the sparse matrix columns to 679 columns corresponding to the genes found. We provide a pickle file with the results of the time consuming differential expression analysis.

Fill zero-values

The next step of our pipeline is to fill the zero values of the dataset. To do that, we find the mean value of the non-zero values of each gene in each state. This means that for each gene we have 4 mean values. We replace the non-zero values accordingly. We have created a dictionary to skip the time consuming part of calculating each value (states_dictionary)

Model training

We reduce the rows of the dataset to 300 rows per state in order to balance every class. We remove from the dataset the columns that correspond to the knocked-out genes. We use Fast.AI [2] for the creation of the regression model. We perform multi-output regression using the 5-cell state proportions as the target. Using the default settings, we find an optimum learning rate and we fit our network.

Validation and test datasets creation

We downloaded the STRING Mus musculus protein network data (full network, scored links between proteins) [3] and used it to infer associations between the held-out knockout genes and the knockout genes in the training set. We selected the associations using the “Combined Score” criterion. For each one of the held-out genes in the extracted gene set, we calculated the weighted by their distribution mean expression using the gene expression values of the associated knockout genes and then calculated the mean expression of all genes in the gene set. We then created the expression vector of the training/held-out gene we are interested in. From the extracted gene set, the held-out genes are not included in the analysis. This vector was used to predict the distribution of the held-out gene.

[1] https://github.com/owkin/PyDESeq2
[2] https://www.fast.ai/
[3] https://string-db.org/cgi/download?sessionId=bxfb4cC22kLB&species_text=Mus+musculus