Introduction to Doblin:a step-by-step through the pipeline

David Gagné-Leroux

2025-05-17

Table of Contents

1.0 *Doblin* Package

The *Doblin* package serves as a comprehensive toolkit designed for preliminary analysis of abundance time series. Its primary purpose is to extract dominant behaviours from abundance time series of a biological system. These "dominant behaviours", called *clones*, provide a better understanding of clonal dynamics within evolving cell populations. They can be used to approximate the main levels of fitness in a population, the time at which a cell line acquired its first beneficial mutation, and develop some intuition about interactions that happen between clones. *Doblin* is specifically designed to manage dominant and persistent barcode lineages generated through *Next Generation Sequencing* (NGS).

The present document is a step-by-step guide intended to facilitate the utilization of *Doblin*. The following sections are chronologically organized and follow the same order as the tool. The code snippets provided in the subsequent pages originate from the main.R script.

1.1 Overview

Note: We assume that users have already extracted the barcodes from their sequencing data.

The initial step of *Doblin* involves offering visualizations of the provided input data. Users have the option to select between a logarithmic-scale model and/or a linear-scale model for visualizing their dataset as a time series. It's noteworthy that plotting linear-scale models can be computationally intensive, potentially requiring several minutes to generate. Additionnally, users have the option to visualize the evolution of their dataset's diversity.

As *Doblin* is tailored to focus on dominant and persistent barcodes, data filtering is employed to retain only the barcodes associated with lineages surpassing a specified minimum frequency and persisting for a predefined duration. Subsequently, barcode lineages demonstrating similar population dynamics are assumed to share a common ecological and evolutionary background. They are thus grouped within the same cluster to form a clone. To quantify the similarity between pairs of barcode lineages, a distance metric based on either Pearson correlation or Dynamic Time Warping (DTW) is used. The resulting pairwise-distance matrix facilitates hierarchical clustering of barcode lineages. An optimal clustering strategy aims to avoid merging barcode lineages with disparate dynamics while ensuring distinctiveness among groupings. The culmination of the analysis entails computing LOESS curves for each resulting cluster, thereby encapsulating the primary clonal lineages.

1.2 Barcode extraction from the FASTQ/FASTA file

This subsection is specifically for users with sequencing data who haven't yet perform barcode extraction. Prior to analysis, the user must identify and extract the barcode sequences from his raw sequencing data. *Bartender* is a useful tool to process barcode data (https://github.com/LaoZZZZZ/bartender-1.1).

Once the barcodes have been extracted, it is primordial to format the data as follows:

- Input file format: A csv file containing the barcode extraction results over **3 columns**: ID, Time, Reads.

  ID: Consensus sequence that identifies a group of barcodes.

  Time: Integer representing the time at which the data was measured.

  Reads: Number of barcodes counted at a given time for a given consensus sequence.

1.3 Installation

1. Open Terminal
2. Change the current working directory to the location where you want to clone the *Doblin* repository.
3. Type `git clone`, and then paste *Doblin*'s URL.
   `git clone https://github.com/DavidGagneLeroux/doblin`
4. Press **Enter** to create your local clone.
5. In Terminal, set your working directory to the `doblin/` folder.

1.4 Command line

Commad line:

```
Rscript ./demo/main.R -t [MIN_FREQUENCY] -o [OUTPUT_DIR] -n [INPUT_FILE_NAME] -i [INPUT_FILE]
-c [TIME_CUTOFF]
```

Here's an example of how to use the command line:

```
Rscript ./demo/main.R -t 0.0005 -o ~/Documents/doblin/ -n test
-i ~/Documents/input.csv -c 12
```

Where

-t: Minimum frequency above which barcodes are assigned colors [default: 0.0005]. The barcodes that do not reach the minimum frequency are colored in grey. This argument is used when plotting the dynamics.

-o: Output directory [default: current working directory].

-n: Input file name.

-i: Input file.

-c: Minimum duration, in terms of time points, for which lineages must persist to be eligible for clustering. If a lineage exists for fewer time points than the specified threshold "-c", it won't be included in the clustering analysis.

2.0 The pipeline

2.1 Data processing

From the outset, the user must choose whether or not to generate a visualization of his dataset:

```
Processing the command line...
Step 0: Processing CSV file...
Do you want to plot the dynamics of your dataset?(y/n): y
```

As previously outlined, the provided input file consists of three columns: ID, Time, and Reads. Upon processing the input data, we transform it into a long-format dataframe to recover the initial, final, mean, and maximum frequencies of each barcode lineage. This restructuring enables the identification of dominant barcodes. Accordingly, we extract the top N barcodes, with N preset to 1000, based on their maximum frequencies. Among the top N barcodes, those that have reached a minimum frequency "-t" are assigned colors. The remaining barcodes are simply assigned the color grey.

The user can choose between plotting the dynamics on a logarithmic scale, a linear scale or both. **Users should be aware that plotting their dynamics on a linear scale can be a heavy task requiring several minutes of processing time.**

```
Step 1: Plotting the dynamics...
1.1 Reshaping input file into long-format dataframe...
1.2 Retrieving the first 1000 barcodes with the highest maximum frequencies...
1.3 Assigning colors to lineages having reached the minimum frequency threshold
among the 1000 most dominant barcoded lines...
```

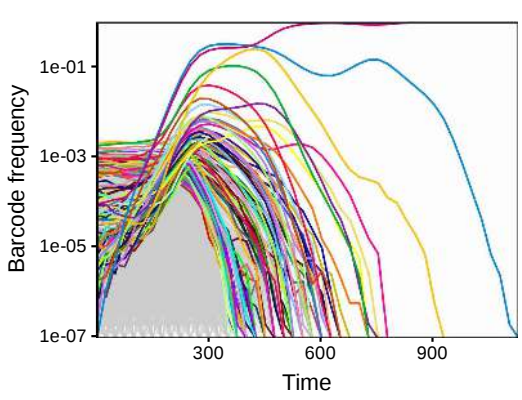
```
Do you want to plot a log-scale model, a linear-scale model or both?
  (logarithmic/linear/both): both
Plotting in progress...
Rendering linear-scale area plot. This may take a few minutes...
```

2.2 Visualizing the dynamics of the dataset

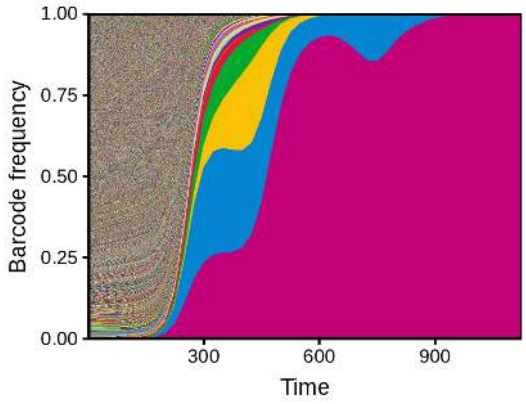
Plotting the dynamics on logarithmic and linear scales provides a comprehensive view of the dataset. As previously indicated, we allocated a distinct color to lineages surpassing a specified minimum frequency, determined by the `-t` argument in the command line (default: 0.0005). For example, the color "magenta [#c20078]" remains assigned to a specific barcode lineage irrespective of whether it appears on the logarithmic or linear scale. No other barcode shares the same color. Maintaining consistency in the color palette across both plots facilitates comparison.

In Figure 1 -a) and -b), barcode lineages from a simulated time series are depicted on both logarithmic and linear scales. The lineages in both plots are arranged based on their maximum frequency. Figure 1-a) illustrates the dynamics on a logarithmic scale, where the dominant lineages are those reaching higher frequencies. In this context, the dominant lineage is visually identified by the color magenta.

Figure 1-b) presents a visualization of the dynamics on a linear scale. Here, dominant lineages are displayed from the bottom of the panel, while lineages with lower frequencies are successively stacked towards the top. It is evident that the magenta lineage occupies the largest area, underscoring its dominance in the dataset.



(a) Visualization of the dynamics on a logarithmic scale



(b) Visualization of the dynamics on a linear scale

Figure 1: Visualizing the dynamics of the dataset

2.3 Diversity

```
Do you want to plot the diversity of your dataset?(y/n): y
2.1 Calculating the diversity...
2.2 Plotting the diversity...
```

The simplest way to quantify the diversity of barcoded lineages in a population is to count the number of unique barcodes observed at a particular time point. However, if lineages differ widely in frequency, then this measure may not be very informative and will suffer from substantial sampling bias (since very low-frequency barcodes will be under-sampled). A more general approach is to quantify the diversity of barcodes using the effective diversity index.

When q = 0, the index simply counts the absolute diversity in the sample, i.e. the total number of unique barcode lineage. This measure is equivalent to the species richness used in ecological studies. When q = 1, the index weights each barcode lineage by its frequency. This measure is equivalent to the exponential of the Shannon entropy H. When q $\rightarrow \infty$, the index is equal to the reciprocal of the proportional abundance of the most common barcode lineages. Thus, only the higher-frequency lineages contribute to the value of this index. By comparing the diversity index across these three orders, we can describe the complex dynamics of the barcode composition over time. Figure 2 illustrates the temporal evolution of our dataset's diversity. As observed in Figures 1-a) and -b), our dataset undergoes a biological "sweep" characterized by the dominance of a single lineage over others. Figure 2 corroborates this observation by depicting a reduction in dataset diversity over time.

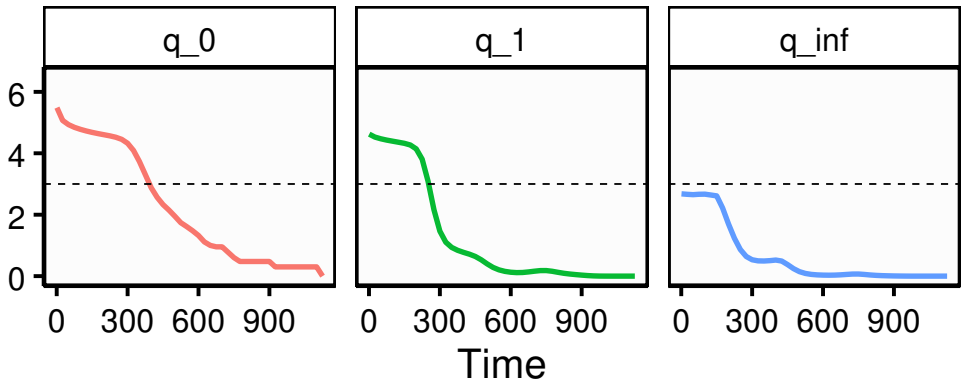


Figure 2: Diversity of the dataset accross 3 different orders (q = 0, 1, ∞)

2.4 Extracting the clonal dynamics

To infer clonal clusters, we employ a clustering approach based on the trajectory of frequency patterns exhibited by barcoded lineages. This entails grouping together lineages that demonstrate similar frequency patterns. To enhance the precision of our clustering approach, we omit barcodes failing to attain a **minimum mean frequency** as well as those lacking an **adequate number of time points** (controlled by the -c argument in the command line). This stringent criterion ensures that all barcode lineages included in the clustering analysis possess a sufficient number of data points for meaningful pairwise comparisons. Consequently, the lineage clustering process prioritizes dominant and persistent barcodes while excluding those that rapidly become extinct.

```
Step 3: Clustering...
Specify a minimum mean frequency below which lineages are not taken into account during
clustering (ex: 0.00005): 0.00005
3.1 Filtering the input data...
```

To conduct hierarchical clustering of dominant and persistent barcodes, users need to specify a linkage/agglomeration method from those available in the `stats::hclust()` function in R. The hierarchical clustering relies on a pairwise distance matrix that encapsulates the similarity between pairs of barcoded lineages. This distance matrix is computed based on a user-defined similarity metric, which can be either Pearson correlation or Dynamic Time Warping (DTW). If the user opts for Pearson correlation, they must also specify a method for computing covariances in the presence of missing values, as per the options provided in the `stats::cor()` documentation. Figure 3 presents a pairwise heatmap, where blue indicates similar lineages and red highlights significant differences among the lineages.

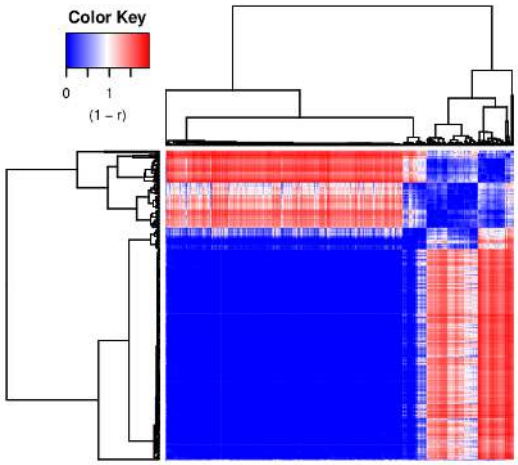


Figure 3: Pairwise heatmap

Therefore, the procedure for hierarchical clustering involves the following steps:

1. Select a linkage/agglomeration method from those available in `stats::hclust()`.
2. Choose a similarity metric (Pearson correlation or DTW) for computing the pairwise distance matrix.
3. If Pearson correlation is selected, specify a method for computing covariances in the presence of missing values, as outlined in the `stats::cor()` R documentation.

```
3.2 Clustering the filtered data...
Enter an agglomeration method (refer to stats::hclust() R documentation): average
Enter the metric to be used to measure similarity between two time-series (pearson/dtw) :
  pearson
Enter a method for computing covariances in the presence of missing values.
Please refer to stats::cor() R documentation (ex: pairwise.complete.obs) :
  pairwise
```

4. If DTW is selected, specify a method for computing the local distance, as outlined in the dtwclust::dtw_basic() R documentation (i.e. L1 or L2).

```
3.2 Clustering the filtered data...
Enter an agglomeration method (refer to stats::hclust() R documentation): average
Enter the metric to be used to measure similarity between two time-series (pearson/dtw):
  dtw
Enter the norm for the local distance calculation
('L1' for Manhattan or 'L2' for (squared) Euclidean): L2
```

To determine the optimal clustering threshold in our hierarchical clustering process, we consider three overarching trends:

Sensitivity to Sequencing Error: Clusters with very few lineages may be overly sensitive to sequencing errors. Hence, their corresponding LOESS trajectories may not accurately represent underlying dynamics.

Similarity Among Clusters: When the threshold is set too low, numerous clusters emerge, but many of them exhibit similar dynamics. This abundance of similar clusters may not provide meaningful insights.

Distinctiveness of Clusters: Conversely, setting the threshold too high results in only a few clusters, potentially grouping together barcodes with diverse dynamics. This can obscure important variations within the dataset.

Our approach involves identifying the cross-over point between the smallest distance between cluster centroids (represented by their respective LOESS average) and the number of clusters (Figure 4). To accomplish this, we compute relative clusters for thresholds ranging from 0.1 to the maximum height of the hierarchical clustering tree. This step empowers users to visualize potential clusters across varying thresholds, enabling informed decision-making regarding the selection of an appropriate threshold. By balancing between sensitivity, cluster distinctiveness, and similarity, users can effectively identify the optimal threshold for their analysis. Figure 4 illustrates the relationship between the number of clusters (blue curve), and the smallest distance between cluster centroids (black curve). As the distance between clusters diminishes, the number of generated clusters tends to increase. Optimal clustering typically occurs around the crossover point (in this case: 0.3).

As previously mentioned, clusters with a small number of lineages may be disproportionately affected by sequencing errors. To mitigate this issue, users are prompted to specify a minimum number of members per cluster (e.g., X=8). However, disregarding clusters with fewer than X members could lead to the omission of dominant clusters. In such instances, users are required to provide a minimum average frequency that must be attained by at least one of the lines within potentially disregarded clusters for them to be considered.

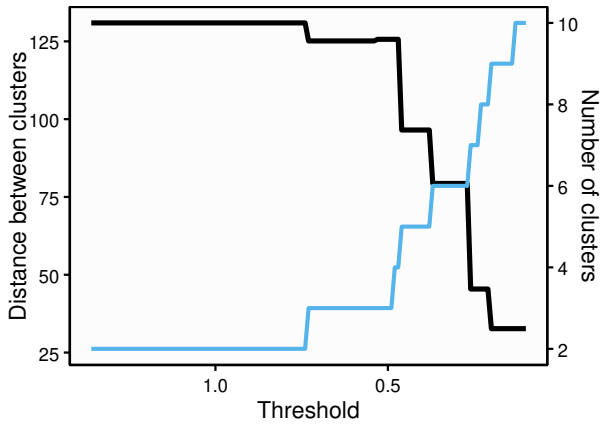


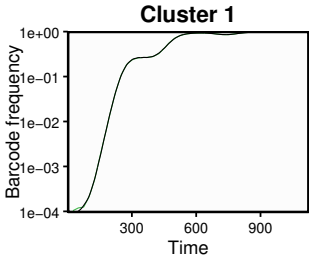
Figure 4: Threshold Selection

```
3.2.1 Computing the relative clusters for ALL thresholds between 0.1 and maximum
height of hierarchical clustering...
3.2.2 Filtering the hierarchical clustering results...
Enter the minimum number of members per cluster for test : 8
Enter the minimum number of members per cluster for test : 8
Enter the minimum average frequency to rescue small clusters: 0.001
Warning message:
  By ignoring clusters with fewer than 8 members, you are potentially ignoring
  dominant clusters.
```

```
3.2.3 Quantifying the hierarchical clustering...
3.2.4 Enter the chosen threshold for the clustering of test : 0.3
```

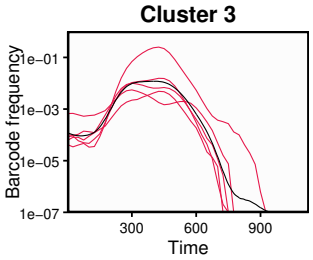
Once a specific threshold has been provided, *Doblin* plots every resulting cluster as well as the final clonal dynamics. It's important to note that the clonal dynamics consist of the LOESS curves of each resulting cluster. Figures 5-a) through 5-f) illustrate the six resulting clusters and their respective lineages. Each cluster's black curve corresponds to its LOESS curve. Figure 5-g) illustrates the clonal dynamics of the dataset.

```
3.2.5 Plotting the resulting clusters...
DONE
```

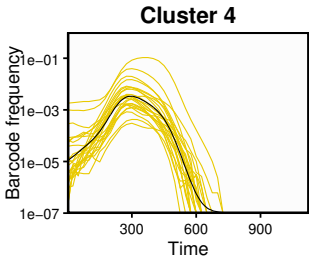


(a) Composition of cluster 1

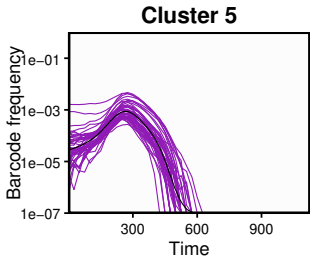
(b) Composition of cluster 2



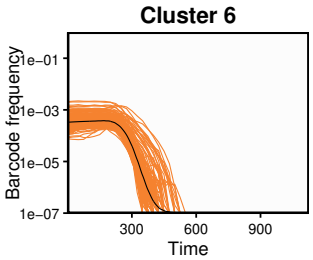
(c) Composition of cluster 3



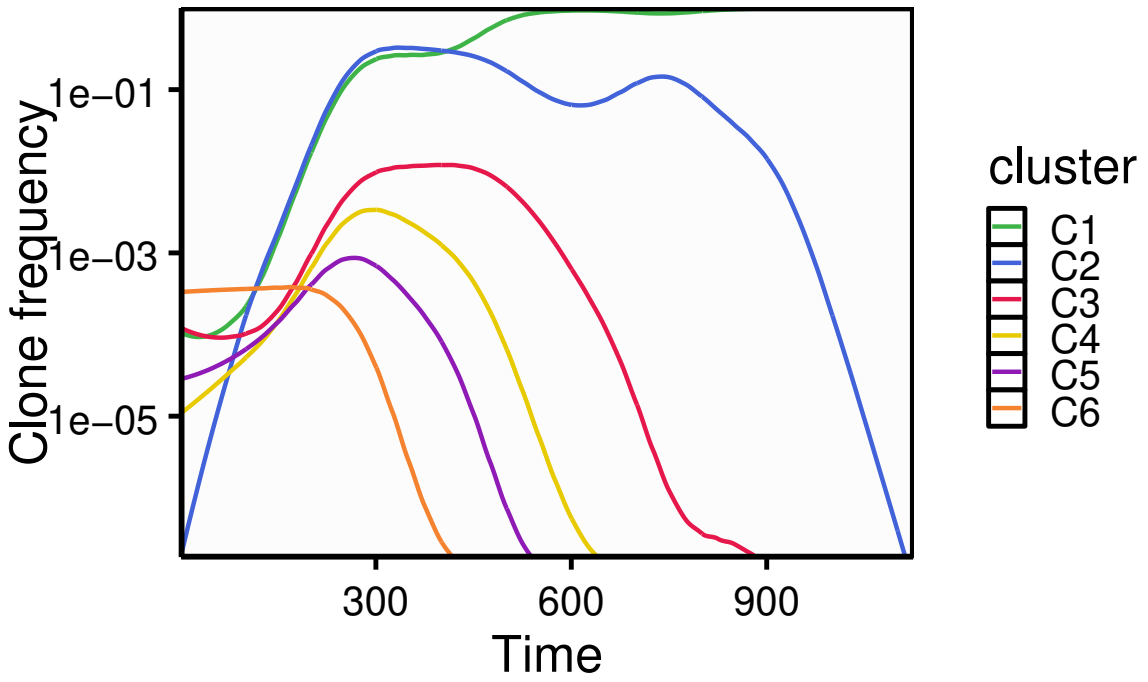
(d) Composition of cluster 4



(e) Composition of cluster 5



(f) Composition of cluster 6



(g) Clonal dynamics

Figure 5: Hierarchical clustering results