# Gene-to-gene Dynamic Methods for Time-Dependent Analysis for Biology Risk Assessments

Alberto Locca

**Birkbeck, University of London**
School of Natural Sciences
MSc Bioinformatics, 2022-2023

Supervised by

**Dr. George Fitton**
position: SEAC Computational Scientist, Unilever
email: george.fitton@unilever.com

**Dr. Alistair Middleton**
position: SEAC Computational Scientist Leader, Unilever
email: alistair.middleton@unilever.com

**Abstract**

Unilever's SEAC division is responsible for the safety and environmental impact assessment of Unilever products. To carry out this task in a sustainable and ethical manner, Unilever promotes the use of non-animal testing methods such as High Throughput Transcriptomics (HTTr) assays, which can be used to test the effect of different concentration of chemicals with different exposure time courses. Time can be factored in using different methodologies, but dynamic methods that model time as a continuous variable are not as well validated as other historically more established approaches.

This project focuses on developing a pipeline for the analysis of RNA-seq time series data by revising some of the most used and popular tools. In particular, the focus of this study is DPGP, a clustering tool that uses a Dirichlet process (DP) to determine the number of clusters, combined with a Gaussian process (GP) for modelling expression levels over time, and Bayesian inference to infer these parameters.

Using DPGP, this study was able (1) to reproduce very similar results to the ones published in its original paper, proving that DPGP can produce very consistent outputs, and (2) to assess its performance on simulated datasets with known characteristics and expected outcomes, which lead to inconclusive interpretations because of the assumptions used by the tool for generating those datasets.

Overall, DPGP has proven to be a potentially good addition to the pipeline. However, it needs to be investigated further in comparison to other similar tools and in the context of a real RNA-seq analysis pipeline.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 SEAC mission and project relevance

Unilever's Safety and Environmental Assurance Centre (SEAC) division was founded in 1990 for assessing the safety and environmental impact of Unilever products. SEAC scientists work together with government scientists, regulators, and academics to carry out pioneering approaches that span different scientific fields – *i.e.* toxicology, microbiology, computational chemistry, bioinformatics, and mathematical modelling – in order to ensure that Unilever products are safe for use and consumption as well as for the environment (Unilever, 2023).

Unilever has a long history of promoting the use of non-animal testing methods for risk assessment, while trying to develop sustainable and viable alternatives, such as *in vitro* assays and computational modelling (Unilever, 2023). Besides the ethical discussion of which I will not enter into the merits, there is also scientific evidence to support the need for non-animal testing: many review studies reported that results from tests conducted in mouse, rat, and rabbit models are highly inconsistent in predicting toxic responses in humans (Van Norman, 2020). Historically, there have been cases of drugs that were previously deemed safe during the clinical trial phases, and later had to be recalled for causing adverse effects, such as Vioxx from Merck which increases the risk of cardiovascular morbidity and mortality, and caused 38000 deaths and 88000 induced heart attacks (Jüni *et al.*, 2004). Other notable cases include Thalidomide, which was withdrawn because it causes phocomelia and fetal deformations in pregnant women, despite there being no teratogenic effect reported in animal models, and Isuprel, which was originally developed for the treatment of asthma and caused an increase in deaths due to the high dosage approved in countries like the United Kingdom, Ireland, Australia and New Zealand (Jalba, 2008). Conversely, the opposite is also true: there are examples of safe drugs which are readily available that would have never passed toxicity testing if conducted in animal models, such as aspirin (that causes reproductive abnormalities and liver toxicity in rats), and paracetamol (that is toxic for cats and dogs) (Van Norman, 2020; Villar *et al.*, 1998).

Another reason in support of non-animal testing is the throughput: Unilever has a portfolio of more than 400 brands – comprising food products, supplements, beauty and personal care products, beverages, and home care products – therefore using animal models for testing all the compounds in all the formulations of all their products is simply not a viable option, both in terms of economical sustainability and in time and resource spending. Animal testing is a very low throughput assay, and relatively expensive when compared to other *in vitro* assays, which, by their nature, can more easily be automated, standardized, and scaled to ideally match the required characteristics for toxicity and risk assessment. An example is the *Tox21* program, a consortium of public agencies, among whose main

objectives are to develop better models for predicting the toxicity of an array of substances (the current goal is set to 10 000), and developing new *in vitro* assays to reduce, refine, and replace the use of animal models (Lynch *et al.*, 2023). Among these technologies, in the current phase of the Tox21 program, the focus is on mid- to high-throughput gene expression screenings using human tissues and cells.

High Throughput Transcriptomics (HTTr) is a particular RNA sequencing (RNA-seq) technique designed to evaluate gene expression changes as effects of multiple conditions, such as treatment with different chemicals, increasing concentrations, or at different time points. In fact, time is an important factor that has often been overlooked in the past, but recent developments in transcriptome studies have allowed the design of more complicated experiments – such as longitudinal studies – in order to capture the biological processes in whole. For example, repeated exposure to small doses of a chemical compound (chronic dosage), or a spike dosage (acute phase) followed by a "recovering" phase. This is where this project begins: our main goal is to leverage the power of HTTr assays for toxicity and safety assessment by developing a new pipeline for the analysis of longitudinal data.

## 1.2 Modelling longitudinal data

RNA-seq can be used to evaluate transcriptional changes in genes already known (present in previous databases), but also to reconstruct the full RNA transcripts, even if they were not previously annotated. The sequencing process generates millions or billions of short sequences called *reads* which are smaller portions of the original transcripts. The reads must then be matched to their original position inside the genome by aligning them to a reference. From this alignment, a final count is produced for each specific region of the genome, and a *count matrix* is produced containing the number of reads per gene (or "genome portion") per different condition included in the experiment, which correspond to how much each gene is expressed (Frazee *et al.*, 2014; Nagalakshmi *et al.*, 2008). This is the starting point of many analytical pipelines.

For the scope of this project, the central principle is how time can be modelled and simultaneously account for expression variation over time. The "common" approach consists of the determination of differentially expressed genes (DEG) among different conditions. One of the most used tools is DESeq (Anders and Huber, 2010) and its later implementation DESeq2 (Love *et al.*, 2014), whose assumption is that each feature is independent, including time points, therefore time is modelled as a categorical variable. The software then performs linear regression for each feature (covariate) across all genes, and a fold change is calculated, indicating how much the expression level of a gene differs compared to the null model, which is when the expression level stays the same.

One obvious issue of this approach is that time cannot always be represented as a categorical variable, and by considering it as such one may oversimplify the biological system. This is an easy implementation of the already consolidated means for analysing genomic data that do not require existing models to be readapted or changed. This interpretation may work when we have less time points or when they are very distant, for example if we are studying a well-known biological mechanism for which we expect a certain outcome only after a certain amount of time has passed, but it poorly applies to an experiment where there are several repeated measurements of the same feature, and therefore each individual measure is no longer truly independent.

To solve this issue, a possible different approach consists in treating time as a continuous variable, and modelling the gene expression using parametric statistics. One of the tools that uses this assumption is ImpulseDE2 (Fischer *et al.*, 2018), which uses "pulse functions" to model gene expression levels over time by representing the transition between an initial state to a peak state, to a steady state. The goodness of fit is then evaluated by calculating a log-likelihood ratio test. One of the limitations of this

approach is the limited number of the possible functions taken into account, which could recapitulate time-dependant gene expression. A more generalized method would be finding the function with the highest likelihood given the expression data, without specifying any function. This can be achieved by using Gaussian Processes (GPs), which are a type of non-parametric non-linear regression. Example tools that apply this concept are PairGP (Vantini *et al.*, 2022) and GPrank (Topa and Honkela, 2018).

## 1.3 Choice of tool

The choice of the tool examined in this study was based on review papers in the Literature – two in particular – detailing methods for RNA-seq data analysis. The first is a comparative study by Spies *et al.* (2019), in which the authors compare tools developed specifically for differential expression analysis of time course data, both on simulated and biological data. The second is a review paper by Oh and Li (2021), in which the authors describe a more comprehensive list of tools and dynamic strategies for studying non-periodical and periodical time course data, *de facto* enriching the list of tools from Spies *et al.*.

From these two papers, a list of tools and their relative original papers was compiled, which was then used in a Python script to cross-search NCBI databases for resources linked to each article. Datasets linked to each publication entry were considered for two main reasons: the importance of reproducibility and the ability to test the consistency of those tools, therefore it is crucial that the authors made their data available, and having as many datasets available as possible for benchmarking the full RNA-seq analysis pipeline of this project. Lastly, the number of citation for each tool was captured in order to develop a ranking system (for a detailed description see section 2.1). The result is an updated table of which table 2.2 shows the most relevant columns.

This approach has three principal shortcomings:

1. To perform cross-searches among NCBI databases, the NCBI E-Utilities software needs to be provided with the relative database ID (*i.e.* PMID to search the Pubmed database). The list compiled from the review papers has DOI strings, which are extra fields in a Pubmed entry and can return unexpected results, like for the case of *PairGP* (last row in table 2.2): the Pubmed search returned 0 papers cited the PairGP paper, which could not be the case since it got cited at least once by Oh and Li. This issue is due to a double DOI corresponding to the same article: one also present in the NCBI records, and one relative to the biorXiv pre-print, which is not accessible by simply using NCBI E-utilities.

2. The average number of citations per year was used to rank the tools, which is not an objective metric for assessing the goodness of a paper, but it gives a rough estimate of the popularity it has among the scientific community. Generally speaking, a more popular tool would get included more often in analytical pipelines than a less popular one, which in turn would result in a more solid consensus. In fact, one of the challenges the computational community is facing is that there is still no consensus for the analysis of longitudinal RNA-seq data, which is also the reason why there is such a large number of tools available.

3. Cross-searches among NCBI databases rely on the authors (or the publishers) to correctly list the resources linked to their publication. In this case, very few Pubmed entries also had the GEO Datasets accession linked, despite having it explicitly written in the text or in the supplementary material. Another possible reason for a GEO Datasets entry to not have an article linked to it could be if the data had been submitted well in advance of the publication, and it had not been updated.

In spite of these problems, the script returned a reasonable outcome. It is worth noting that the tools for the analysis of periodical time courses are among the most popular. It may be because circadian rhythmic and cell-cycling changes have historically been more studied, or because some of these tools have been repurposed from older applications to also adapt to this type of analysis. Either way, since the HTTr assays used in this project involve non-periodical time course data, those tools are not taken into consideration. Within the non-periodical time course tools, the tools previously suggested by Unilever's external collaborators – maSigpro and ImpulseDE2 – are ranked among the highest positions. Lastly, some tools actually have publicly available datasets linked to their papers.

## 1.4 DPGP

This study focuses on the *DPGP* (*Dirichlet Process Gaussian Process*) tool based on the evaluation of the results of the NCBI cross-search from the previous section and shown in table 2.2. Among the tools for the analysis of non-periodical time course, DPGP is one of the highest ranking, but unlike the others that were design to perform differential expression analysis, it has been developed for clustering time series of transcriptional data.

Clustering can be particularly useful in analysing RNA-seq data because it can be used to visualize the results, identify potential bias in the data (such as batch effects), but also to reduce the complexity by grouping genes into sub-categories of "expression behaviours". Furthermore, genes that belong to a cluster can be annotated "by association" assuming that they must share a common feature in order to be put in the same category, *e.g.* regulatory pathway, molecular function, cellular location, sequence motif, etc. (Walker *et al.*, 1999).

Unlike other clustering algorithms, DPGP does not need to pre-determine the number of clusters – like in k-means clustering – but it uses a *Dirichlet process* (DP) to infer it from the data itself, combined with a Gaussian process (GP) for modelling expression levels over time.

### 1.4.1 Bayesian inference

DPGP uses both Dirichlet and Gaussian processes as priors, or in other words, the initial information about the data. Priors are used in Bayesian inference to infer the posterior distribution, which can be described as the updated initial information after factoring new observation (new knowledge) from the data.

In Bayesian inference, the posterior probability is calculated with Bayes' theorem, which uses the estimate of the probability of the hypothesis before factoring the data – prior –, and the probability of observing the data given the parameters (or the model) – likelihood – to get the probability of observing the parameters given the data – posterior.

Bayes' theorem general formula is:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)}$$

where

- $H$ is the "hypothesis" event
- $D$ is the "data" or "information" event
- $P(H|D)$ is the posterior conditional probability (probability of observing the hypothesis given the data)

- $P(D|H)$ is the *likelihood*, or prior conditional probability (probability of observing the data given the hypothesis)
- $P(H)$ is the prior, and
- $P(D)$ is the *marginal likelihood*, or the probability of the observed data.

Lastly, to estimate the posterior distribution of the parameters of the mixed DPGP model, the software uses a Markov chain Monte Carlo method – in particular, a Gibbs sampling algorithm – to simultaneously infer the number of clusters *and* the parameters that describe the gene expression trajectories for each cluster.

### 1.4.2 Dirichlet process

Dirichlet processes are often used in Bayesian models as a non-parametric prior because they allow not having to determine a certain parameter – like in this case, the number of clusters – but lets the data itself inform that decision.

The DP can be considered as a distribution over distributions, where the latter can be parametrized as the "mean" of the DP, with a "concentration" parameter that determines how closely the sampled distributions are to the base "mean" distribution.

DP can be used for clustering because distributions sampled from a DP can be considered an isolated group (or cluster) from which the data comes from.

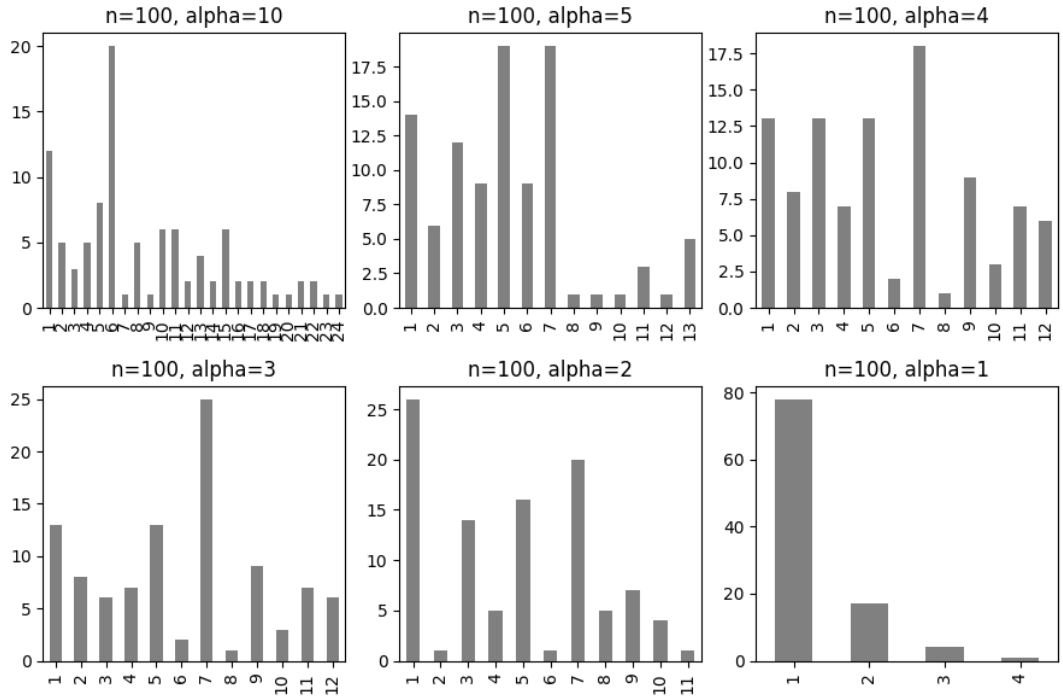


Figure 1.1: Chinese restaurant process example. The picture shows the effect of the concentration parameter $\alpha$, which affects the probability of creating a new group – or table, or cluster – with a fix number to be organized. Each bar represents the number of assigned values to that group. As the $\alpha$ value decreases, also the number of groups gets lower, with a maximum of 24 groups for $\alpha = 10$, and a minimum of 4 for $\alpha = 1$. The default value used in the DPGP software is $\alpha = 1$.

A classical example used to visualize DP is the Chinese restaurant process (CRP). In this metaphor, people enter an imaginary Chinese restaurant, and they have two choices: sit to a table where there are other people sitting there, or sit to a new empty table. The probability for the first choice is proportional

to the number of people already at the table, meaning it's more likely to sit at a table with many people than few. The probability for the second choice is proportional to the concentration parameter. The script `dp_chinese_restaurant.py` recreates this simple example, and produces figure 1.1, which shows the effect of the concentration parameter: the lower the value, the more "concentrated" people are, meaning it is harder for people to sit to new tables.

At the end of the CRP, the result is a particular probability distribution of assignments, which is one of the possible distributions that could be sampled from the DP.

### 1.4.3 Gaussian process

DPGP uses Gaussian process (GP) as a prior for each specific cluster over the range of possible gene expression trajectories over time.

In this case, the base distribution from the DP captures the distribution of all the parameters for each cluster-specific GP, which in turn captures all the functions that model the data points (gene expression over time).

GP also depends on other parameters, expressed as the hyperprior distribution Inverse Gamma, which can be parametrized as $\alpha$ and $\beta$ variables that capture respectively the shape and rate of the Inverse Gamma distribution. In the DPGP software these two parameters are set to $\alpha = 12$ and $\beta = 2$ based on the authors experience, but they can be altered to allow for different degree of variability in the data; in particular, lowering the shape parameter will allow for a greater marginal variance within each cluster.

# Chapter 2

# Materials and Methods

## 2.1 Papers metric for tool selection

Starting from the review paper by Oh and Li, in which the authors describe a list of available tools and dynamic strategies for studying non-periodical and periodical time course data, table 2.1 was compiled by collecting all the tools discussed by Oh and Li. The ones for handling batch effect have not been included (which are listed in Table 3 of the original paper; see *Original table* column in table 2.1) because batch effects are not taken into account for the scope of the pipeline of this project, but they may be included in future implementations.

| **Name** | **Original table** | **Type** | **Time course** | **DOI** |
|---|---|---|---|---|
| Next maSigpro | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/bioinformatics/btu333 |
| DyNB | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/bioinformatics/btu274 |
| EBSeq-HMM | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/bioinformatics/btv193 |
| Ngsp | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/bioinformatics/btu699 |
| Lmms | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1371/journal.pone.0134540 |
| timeSeq | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1186/s12859-016-1180-9 |
| splineTimeR | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1371/journal.pone.0160791 |
| ImpluseDE2 | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/nar/gky675 |
| Trendy | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1186/s12859-018-2405-x |
| AR | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1038/s41598-018-37397-7 |
| MAPTest | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1111/biom.13144 |
| TimeMeter | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/nar/gkaa142 |
| PairGP | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1016/j.compbiomed.2022.105268 |
| GPrank | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1186/s12859-018-2370-4 |
| Dream | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/bioinformatics/btaa687 |
| rmRNAseq | Table 1 | Dynamic gene-by-gene | Non-periodical | 10.1093/bioinformatics/btaa525 |
| JTK_CYCLE | Table 2 | Dynamic gene-by-gene | Periodical | 10.1177/0748730410379711 |
| MetaCycle | Table 2 | Dynamic gene-by-gene | Periodical | 10.1093/bioinformatics/btw405 |
| RAIN | Table 2 | Dynamic gene-by-gene | Periodical | 10.1177/0748730414553029 |
| DODR | Table 2 | Dynamic gene-by-gene | Periodical | 10.1093/bioinformatics/btw309 |
| LimoRhyde | Table 2 | Dynamic gene-by-gene | Periodical | 10.1177/0748730418813785 |
| Tcgsaseq | Table 4 | Coherent gene-to-gene | Non-periodical | 10.1093/biostatistics/kxx005 |
| FunPat | Table 4 | Coherent gene-to-gene | Non-periodical | 10.1186/1471-2164-16-S6-S2 |
| DPGP | Table 4 | Coherent gene-to-gene | Non-periodical | 10.1371/journal.pcbi.1005896 |
| LPWC | Table 4 | Coherent gene-to-gene | Non-periodical | 10.1186/s12859-019-3324-1 |

Table 2.1: Collection of tools as listed in the paper by Oh and Li (2021). *Type* and *Time course* are two of the labels used to classify the different tools. For convenience, a *DOI* column was included for easy searching.

A Python program was then used to select the most relevant tools among the listed ones by accomplishing two major tasks:

1. searching for databases linked to the publication,
2. developing a metric on which papers could be scored.

The script is a basic wrapper for the NCBI E-utilities (Bethesda, 2010) Python implementation within the `Biopython` module (Cock *et al.*, 2009). For each tool, the listed publication DOI string was used to search the NCBI databases for resources linked to each paper, in the attempt to assess if the datasets used in the publications were made available in public databases, namely NCBI GEO Datasets – a curated repository of gene expression data – and NCBI BioProject – a collection of biological data per project/effort (meaning each entry contains descriptive information about the experimental data, and could link to multiple resources and datasets). In order to score the tools, the number of citation for each paper was used to get the average number of citations per year by calculating the difference in days between the current date and the date of publication, and dividing it by the number of days in a year (365.25, accounting for leap years). Note that because of this, every future execution will produce different values. The result is an updated version of the previous table 2.1 with six extra columns: PMID (Pubmed ID), publication date (formatted as year-month-day), number of citations, average number of citations per year, GEO Datasets ID, and BioProject ID. Table 2.2 shows a shorter version (for readability purpose) of the script output.

| **Name** | **Time course** | **Cited by** | **Average yearly citations** | **GEO Dataset ID** | **Bioproject ID** |
|---|---|---|---|---|---|
| JTK_CYCLE | Periodical | 482 | 37.50 | | |
| MetaCycle | Periodical | 212 | 28.47 | | |
| RAIN | Periodical | 148 | 16.81 | | |
| Next maSigpro | Non-periodical | 143 | 15.59 | | |
| Dream | Non-periodical | 52 | 14.60 | | |
| ImpluseDE2 | Non-periodical | 53 | 8.79 | | |
| DPGP | Non-periodical | 54 | 8.62 | 200104714 | 413586 |
| DODR | Periodical | 46 | 6.17 | | |
| LimoRhyde | Periodical | 26 | 5.54 | | |
| EBSeq-HMM | Non-periodical | 44 | 4.99 | | |
| DyNB | Non-periodical | 33 | 3.61 | | |
| Trendy | Non-periodical | 16 | 3.24 | | |
| splineTimeR | Non-periodical | 20 | 2.75 | | |
| Lmms | Non-periodical | 20 | 2.37 | | |
| Ngsp | Non-periodical | 18 | 2.05 | | |
| FunPat | Non-periodical | 16 | 1.96 | | |
| rmRNAseq | Non-periodical | 8 | 1.95 | | |
| timeSeq | Non-periodical | 11 | 1.48 | | |
| LPWC | Non-periodical | 6 | 1.36 | | |
| Tcgsaseq | Non-periodical | 7 | 0.99 | | |
| GPrank | Non-periodical | 4 | 0.78 | | |
| AR | Non-periodical | 4 | 0.74 | | |
| TimeMeter | Non-periodical | 1 | 0.29 | 200130438 | 540258 |
| MAPTest | Non-periodical | 1 | 0.21 | | |
| PairGP | Non-periodical | 0 | 0.00 | 200154467 | 646394 |

Table 2.2: Paper metrics table obtained from table 2.1 after running the script `entrez.py` (see 2.5). Tools are listed in descending order by *Average yearly citations*. Some columns have been omitted for readability.

## 2.2 DPGP setup

The DPGP software is available as a Python program at the GitHub repository `https://github.com/PrincetonUniversity/DP_GP_cluster` (last updated 22 September 2017). Since its publication (McDowell *et al.*, 2018), it has not been updated, and it looks like it is not actively maintained.

In the repository, it states that it has been tested on GNU/Linux systems with Python 2.7 and Anaconda distributions, so to mirror the installation steps suggested by the authors, a conda environment was created with that specified Python version together with the version of the required packages on which DPGP depends – *i.e.* `scikit-learn`, `pandas`, `GPy`, `numpy`, `scipy`, `matplotlib`, and `cython`. The environment file was exported with all packages dependencies and made it available in the code repository (see 2.5), both in text and YAML format for convenience.

Next, after activating the environment, DPGP is installed from its official GitHub repository using `pip`. To recreate the installation process, run the following commands:

```
conda create --name dpgp --file DPGP_conda_env.txt
conda activate dpgp
pip install git+https://github.com/PrincetonUniversity/DP_GP_cluster.git
conda deactivate
```

## 2.3 Response to $H_2O_2$ in archaebacterium *Halobacterium salinarum*

As test case, McDowell *et al.* used a real biological dataset previously published by their own group (Sharma *et al.*, 2012), which can be downloaded at the GEO accession GSE33980. In that original work, the authors studied the transcriptional response of the archaebacterium *Halobacterium salinarum* to the treatment with hydrogen peroxide.

The choice of using this dataset was backed by the fact that the original study was conducted in a relatively simple system with a small genome, and the results had a clear biological meaning in response to a single stimulus over time.

### 2.3.1 Original experimental design

*Halobacterium salinarum* strains were cultured under standard conditions (37 °C for 48 hours under continuous shaking) until the growth rate reached a mid-logarithmic phase. For RNA extraction, 4 mL culture aliquots were removed prior to the addition of 25 mmol $H_2O_2$, and at five time points following $H_2O_2$ addition – at 10, 20, 40, 60, and 80 minutes. Each measurement had two technical replicates, accounting for 12 total measurements per strain – a control strain, termed *ura3*, and a mutant strain, termed either *d258* or Δ*rosR*. The mutant strain has a deletion that results in the absence of the transcription factor rosR, which is found to have a protective effect to oxidative stress, such as the exposure to $H_2O_2$.

Gene expression was evaluated using Agilent microarray systems – GEO platform accession GPL14876 – which contains 2410 non-redundant open reading frames of the *H. salinarum* NRC-1 genome. Raw data were analysed using R Bioconductor `m-array` (Paquet *et al.*, 2023) and `limma` packages (Ritchie *et al.*, 2015; Smyth *et al.*, 2023): background was subtracted using `normexp`, Loess normalization was performed within each array, and quantile normalization between all arrays. Any probe for each gene lying outside the $99^{\text{th}}$% confidence interval was removed using Dixon's test. The remaining probe intensities for each gene were averaged and log2 ratios were calculated, yielding one expression ratio per gene. This processed count table was made available in the GEO Dataset entry supplementary files as `GSE33980_average_data.txt.gz`.

Sharma *et al.* then used the Gaggle data analysis web environment (Shannon *et al.*, 2006) to perform statistical analysis and differential expression analysis of the data. They identified 626 DEGs which they clustered using a k-means algorithm.

### 2.3.2 Data processing and clustering

In order to mirror the analysis conducted by McDowell *et al.*, a series of Python scripts were used to perform the following steps.

**Data collection** A python script – `geo_DPGP.py` – that uses the GEOparse package (Gumienny, 2021) to access each GEO Dataset entry provided – in this case GEO accession GSE33980 – and download all the supplementary data listed – in this case the processed count table `GSE33980_average_data.txt.gz`. A second Python script – `gse33980_suppl.py` – that extrapolates the list of the names of all 626 DEGs, which was included in an Excel table in the supplementary materials of the paper by Sharma *et al.*.

**Data pre-processing** In order to run DPGP, the data have to be formatted in a specific way: each condition – control and mutant strains – has to be run separately, and the count table has to be a tab formatted table where the first column contains the gene identifiers (or any other type of label), and the column names must represent each time point. Each replicate has to be separated in its own count table, but they have to be input together to DPGP.

The Python script `gse33980_data_preprocess.py` was written for this purpose. It filters the dataset obtained in the previous step for only the 626 DEGs (which were actually 616), and generates a "metadata" table from the measurement labels to ulteriorly subset the data into the desired format: retains only the measurements for the group treated with $H_2O_2$, splits the dataset into the two strains, which it ulteriorly splits into single replicates.

DPGP takes as inputs the list of all the expression count tables replicates, but they must have matching number of time points. In this case, the second replicate of the mutant strain is missing the 20 min time point, which is present in all other conditions. DPGP was run using different inputs combination – excluding that time point in the mutant group only, or in both groups (data not shown) – but the results that more closely match the ones from McDowell *et al.* are obtained by filling the missing measurements using the data from the first replicate. When both strains were treated in this way – using the same 20 min time point of the first replicate in both tables – the results were again discordant from the ones obtained in the DPGP paper (data not shown).

**Clustering** Finally, a bash script – `dpgp_gse33980_paper.sh` – executes DPGP and performs the clustering, using the same flags that corresponds to the parameters used in the paper: `--fast` will run DPGP in "fast" mode, which McDowell *et al.* used throughout their paper. `--sigma_n2_shape` and `--sigma_n2_rate` are the hyperparameters $\alpha$ and $\beta$ respectively, that can be varied to allow for greater variability by reducing the shape parameter. In the paper, they stated that they used $\alpha = 6$ (default 12) and $\beta = 2$ (default 2), because these are microarray data. Finally, `--plot` will output cluster images like the ones shown in the original work (see figure 3.1), in the format specified by `-p`.

### 2.3.3 Results comparison

McDowell *et al.* clustered gene trajectories for each of the control and mutant strains in independent DPGP modelling runs, so to compare them and determine the gene cluster assignment changes in response to the mutation, they computed the Pearson correlation coefficient on all the mean trajectories combinations of clusters. The clusters with the highest coefficients were considered equivalent across the two strains. To test the significance of gene switching between equivalent clusters, the authors used the Fisher's exact test.

In order to reproduce this analysis, the Python script `gse33980_dpgp_results.py` first calculates the mean expression trajectories of each cluster using the posterior cluster model. This script has been adapted from function present in the DPGP code, `plot.py` in particular. The script also produces single plots for each cluster (used in the section 3.1.1), instead of the standard graphical output of DPGP, which has six cluster plots per image, and alternative similarity matrix images for both strains (without the default dendrogram, see figure 3.10), and with the gene order rearranged for the mutant strain (see

figure 3.11) to mirror the image shown in figure 3.1 panel N. Since this script rely on the DPGP Python module, it must be run in the conda environment with the following command:

```
conda run -n dpgp python DPGP/src/gse33980_dpgp_results.py
```

A second Python script was written – `gse33980_dpgp_analysis.py` – to calculate the Pearson correlation coefficient – using the `pearsonr` function of the Python `SciPy` module (Virtanen *et al.*, 2020) – of all the possible pairs of posterior cluster mean expression trajectories, and determine which are the equivalent clusters by selecting the pair with the highest value of $r$. The matrix of $r$ values of the full pairwise comparisons is shown in table 3.1. The full 3D matrix of $r$ and $p$ values is saved as a `numpy` binary file – `geo_correlation_matrix.npy`.

The same script also extrapolates from the DPGP results (specifically from the optimal clustering output, which is a table containing the list of all genes and their corresponding cluster label) the number of co-clustered genes in each possible pairwise combination of control *vs.* mutant clusters, and produces table 3.2, where the highlighted cells correspond to the number of genes that are clustered in the equivalent cluster, meaning that they maintain the same expression behaviour over time. Table 3.2 is used to calculate the contingency tables of the equivalent cluster pairs, which in turn are used to calculate the Fisher's exact test. Each contingency table has the following format:

| | **Belong to mutant cluster $n$** | **Do NOT belong to mutant cluster $n$** |
|---|---|---|
| **Belong to control cluster $m$** | $x$ | $y = row - x$ |
| **Do NOT belong to control cluster $m$** | $z = col - x$ | $616 - (x + y + z)$ |

Where

- $x$ is the number of genes belonging to both clusters
- $y$ is the number of remaining genes in the control cluster
- $z$ is the number of remaining genes in the mutant cluster
- the last cell is the number of genes not belonging to any of those two clusters, which correspond to the total number of genes – 616 in this case – minus the genes in either one of the two clusters $m$ and $n$

Finally, a summary of the statistical analysis of the equivalent cluster pairs is produced (see table 3.3), showing for each control cluster which is the equivalent mutant cluster, the Pearson correlation $r$ and $p$ values, and Odds ratio and Fisher's exact test (FET) $p$ value, which are calculated using the `fisher_exact` function within the same `SciPy` module (Virtanen *et al.*, 2020). The last column contains the number of genes that show a different gene expression dynamic (also expressed as a percentage of the total number of genes of control cluster), which was calculated from the relative contingency table.

## 2.4 Data simulation

Since the pipeline this project is building will perform differential expression analysis using, among other tools, `ImpulseDE2` (Fischer *et al.*, 2018), DPGP was also tested in this study on toy datasets that have gone through the pipeline.

### 2.4.1 Datasets generation

**ImpulseDE2 installation** ImpulseDE2 is available as an R package at the GitHub repository (`https://github.com/YosefLab/ImpulseDE2`, last updated 14 September 2022), and in Bioconductor version 3.10 (`https://bioconductor.org/packages/3.10/bioc/html/ImpulseDE2.html`, version 1.10.0). The package has been removed starting from Bioconductor version 3.13. For this reason, ImpulseDE2 has been downloaded from the GitHub repository and included in an R project (available in the repository of this study, see section 2.5) using `renv`, which creates a `lockfile` containing the information of all the packages used. To recreate the installation process, open the R project `impulseDE2.Rproj`, and run the following command in an R console:

```
renv::restore()
```

**Dataset creation and DE analysis** The following steps were recreated from the tutorial present in the `ImpulseDE2` vignette. In order to produce the simulated datasets to use in this study, the `simulateDataSetImpulseDE2` function was used, which can generate a specified amount of gene expression vectors modelled according to four possible functions:

1. *constant*, which produces constant gene count with no effect over time,
2. *impulse*, where the effect over time is represented from an initial state that shifts to a "peak" state, and to a "steady" state,
3. *linear*, where the gene counts follow a linear trajectory over time, and
4. *sigmoid*, which produces genes counts with a sigmoidal time progression.

Two datasets were created: one with 400 genes using the linear function only (*linear dataset*), and a second one using the constant, impulse, and linear functions with 200 genes each, for a total of 600 genes (*mixed dataset*). Both datasets have 8 time points and three replicates. These numbers were chosen to have a starting size comparable to the one from the real biological dataset (which has 626 genes, see section 2.3.1), and still be relative fast to cluster with DPGP.

DE analysis was performed with the `runImpulseDE2` function, which in this case – where no alternative condition has been provided – compares gene expression levels against a constant expression model with no variations over time. The resulting list of DEG was used in the next step to filter the whole dataset before running DPGP.

### 2.4.2 Clustering and analysis

Next, a processing step was performed, similar to the one described in section 2.3.2, which is necessary for running DPGP.

**Data pre-processing** The Python script `simulated_data_preprocess.py` reads each of the two datasets – linear and mixed – and produces the individual expression matrices for each replicate, which are then provided together as the input of DPGP. For each dataset, the script also creates two alternative sub-datasets: one with the full list of genes, and a second one with only DEGs, for a total of four datasets.

**Clustering** DPGP was run on each one of these sub-datasets – with the full list of genes, or only DEGs, from each simulated dataset, linear or mixed – using the bash script `dpgp_simulated.sh`. The parameters used are again similar to the ones used on the biological data: `--fast` is used to run DPGP in "fast" mode,

and `--plot` to produce the standard graphical output in the format specified by `-p`. The script also iteratively pass a value for the $\alpha$ hyperparameter (`--sigma_n2_shape`), which is used to allow for variability within the data. Not knowing how variable these datasets are (since the `simulateDataSetImpulseDE2` function also includes dispersion factors), an array of values was chosen: $\alpha = 12$ (the default), 9, 6 (the value used by McDowell *et al.*), and 3.

**Result analysis** The results of each DPGP run are then reformatted using the Python script `simulated_dpgp_results.py`, which produces the images shown in section 3.2, with similar quality to the one described in section 2.3.3. Since this script rely on the DPGP Python module, it must be run in the conda environment with the following command:

```
conda run -n dpgp python DPGP/src/simulated_dpgp_results.py
```

Lastly, the Python script `simulated_dpgp_analysis.py` evaluates DPGP ability to capture groups of genes generated by different `ImpulseDE2` model functions. The script takes advantage of the fact that the genes are in sequential order in the generated dataset, where the first are the number of constant genes, followed by the one generated by the impulse function, then the linear, and finally the sigmoid. The script takes all the DPGP results from the mixed sub-datasets and produces summary tables showing the count of genes per cluster that were generated by the three different functions – constant, impulse, and linear. The background gradient is the result of a row-wise comparison: the darker the colour, the higher number of genes generated from that function belong to a specific cluster.

## 2.5 Code availability

All the code used in this dissertation is available at the GitHub repository `https://github.com/bio-tilion/AL_MScProject.git`.

# Chapter 3

# Results

This study was conducted to test the viability of the DPGP tool on longitudinal RNA-seq data, with the end goal of including it in the project pipeline whether it produces reasonably sound results. For this purpose, DPGP was first used to recreate the findings published in its original paper (McDowell *et al.*, 2018), and then tested it on simulated datasets, as a way to assess its behaviour in a controlled setting environment with known expectations.

## 3.1 Reproducing DPGP results

The objective of this study was to obtain a result that is as close as possible to the analysis done by McDowell *et al.* in the DPGP paper, by focusing on the dataset obtained from the treatment of *Halobacterium salinarum* with $H_2O_2$. Figure 3.1 – which is labelled *Fig 2* in the original paper – shows the plots that were recreated in this study.

First, the GEO GSE33980 dataset was downloaded, which also contains data from a second experiment where *H. salinarum* has been treated with a different compound. The pre-processing step filters that data out and prepares the expression matrices to input into DPGP (for the full description, see 2.3.2). In the original work that produced the GEO dataset (Sharma *et al.*, 2012), the authors identified 616 differentially expressed genes (DEGs) which they clustered using a k-means algorithm. McDowell *et al.* ran DPGP only on the DEGs and not the whole dataset, but it was not shared among the supplementary material of the DPGP paper, so the list of DEGs was retrieved from the supplementary material of the paper by Sharma *et al.*. That table includes another version of the gene expression dataset already divided by experiment, so throughout the pre-processing step the same actions were performed on both datasets – the one obtained from the supplementary material, and the one downloaded from the GEO database. After the individual expression matrices were obtained to be used as DPGP input, a comparison was performed using `diff`, which showed that the matching expression table files were identical (data not shown) therefore, throughout the analysis, only the dataset downloaded from the GEO database was used, because it is easier to access and publicly available, despite needing an extra filtering step.

Then, by running the script `dpgp_gse33980_paper.sh`, DPGP is iteratively executed for both strains. The same flags and parameters used by McDowell *et al.* were used, even though some of their choices could be disputed. For instance, they claimed that the fast (fDPGP) and default implementation of DPGP gave very similar results on simulated datasets, so they chose to use fDPGP for all biological data. However, when the default DPGP was run on the same GEO dataset, the results were very different: 17 clusters for the control strain, with median 34 genes per cluster, and 14 clusters for the mutant strain, with median 27 genes per cluster (data not shown). They also used a lower value for the

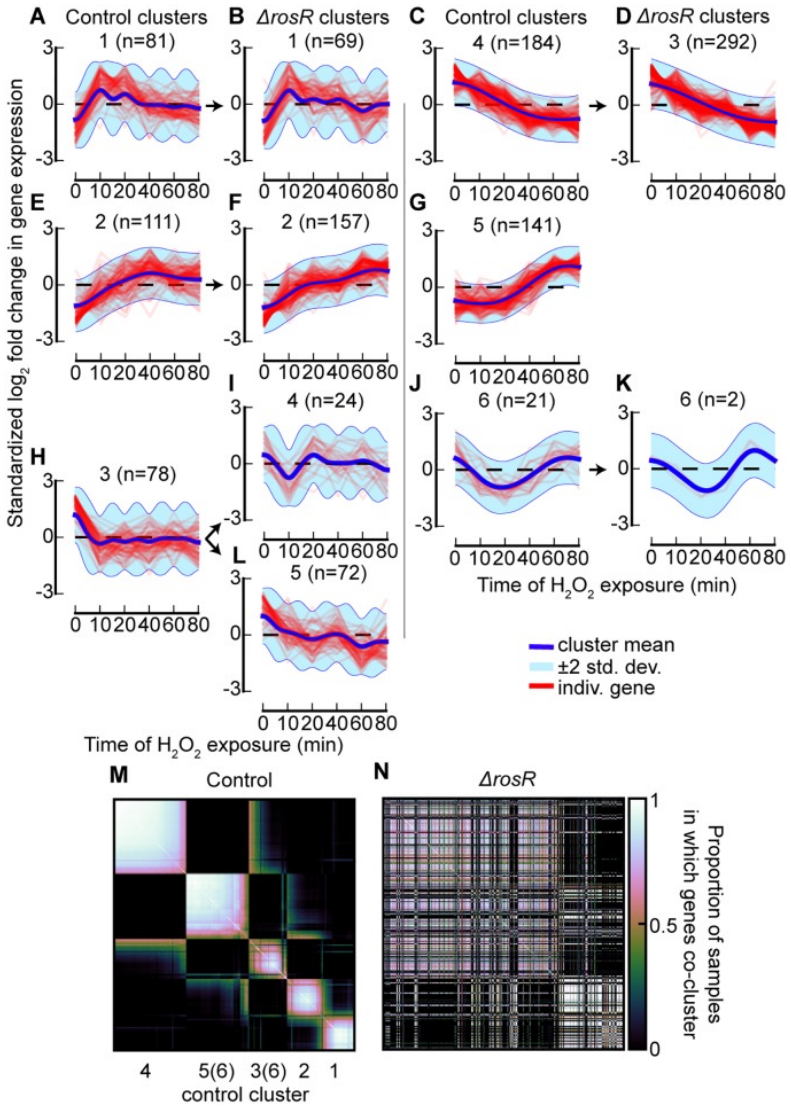


Figure 3.1: Original figure of the oxidative stress response clusters from McDowell *et al.*, 2018. (A-L) Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels. The first and third columns (panels A, E, H, C, G, and J) show control clusters, and the second and fourth columns (panels B, F, I, L, D, and K) the mutant clusters. The arrows point at the equivalent mutant cluster based on the Pearson correlation coefficient of the mean trajectories. (M, N) Heatmaps showing the proportion of sampled genes that cluster with every other gene for the control strain (M), and for the mutant ΔrosR strain (N). The gene order has been determined by Ward's linkage for the control, and for the mutant it has been assigned the same order as the control.

*shape* hyperparameter, which is done to allow for a greater marginal variance within clusters. This is the case of microarray data, but they did not specify how they chose a particular shape value, making it a quite arbitrary choice. Lastly, the tool has an argument that allows for the true time difference between time point to be modelled, but the authors did not pass it, so the model assumed that all the time points are equally spaced, which means that the rate of change in expression is roughly equivalent between all neighbouring time points. Conceptually speaking, this may be true depending on the time course of the experiment but, in this case, it could have affected the analysis, because the first three time points are 10 min apart, and the next three are 20 min apart. In an organism like *H. salinarum* – or in a model bacterium – this can be equivalent to twice the number of replication cycles, and significant changes in gene expression.

Nevertheless, with the parameters used by McDowell *et al.*, the results were almost identical to the

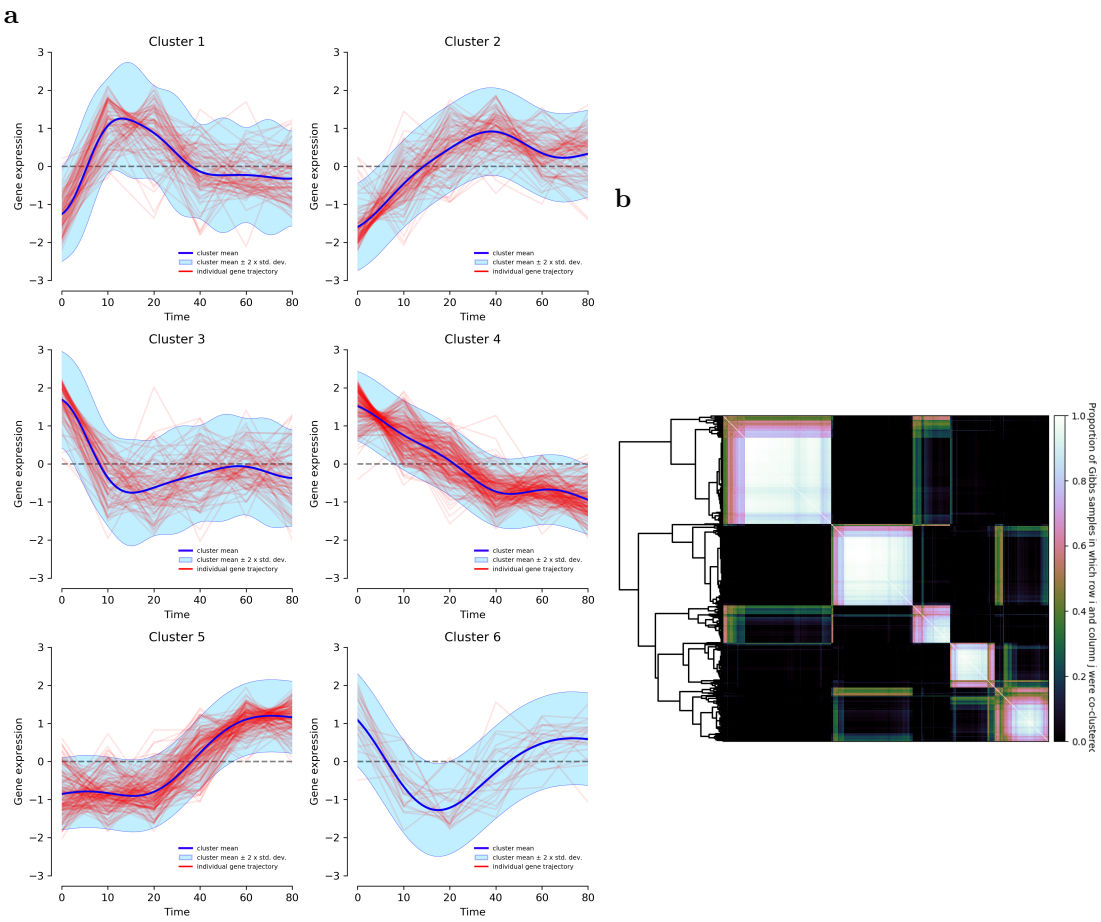


Figure 3.2: DPGP default graphical output for the control strain. (a) Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels. (b) Heatmap showing the proportion of sampled genes that cluster with every other gene. The gene order has been determined by complete linkage (using the `linkage` function within `SciPy`), and displayed as a dendrogram on the left.

ones shown in the paper. Figures 3.2 and 3.3 show the default DPGP graphical output, corresponding to the gene expression trajectories of all the clusters (panel a), and to the heatmap of the proportion of sampled genes that cluster together (panel b). The latter may be used to visualize at a glance the main groups within the data, identified by the genes that more consistently are clustered together are closer to the diagonal and forms whiter squares, or by genes that are represented by other hues and are farther from the diagonal have a chance to be clustered in other cluster. The main white "blocks" are indicative of the main clusters.

Figure 3.3 shows the default graphical output for the mutant strain. When compared to the corresponding graphs in figure 3.1, both the gene expression trajectories and the number of genes in each cluster are identical. The heatmaps (figure 3.3 panel b and figure 3.1 panel N) are different because the one in the paper was obtained by rearranging the order of the genes. The output for the control strain (figure 3.2) is instead slightly different from the corresponding published one, in terms of gene trajectories, number of gene assignment, and heatmap visualization. This can be explained by two possible scenarios: either the authors used a different seed when clustering the mutant strain, or they processed those data in a way not accounted in this study without documenting it.

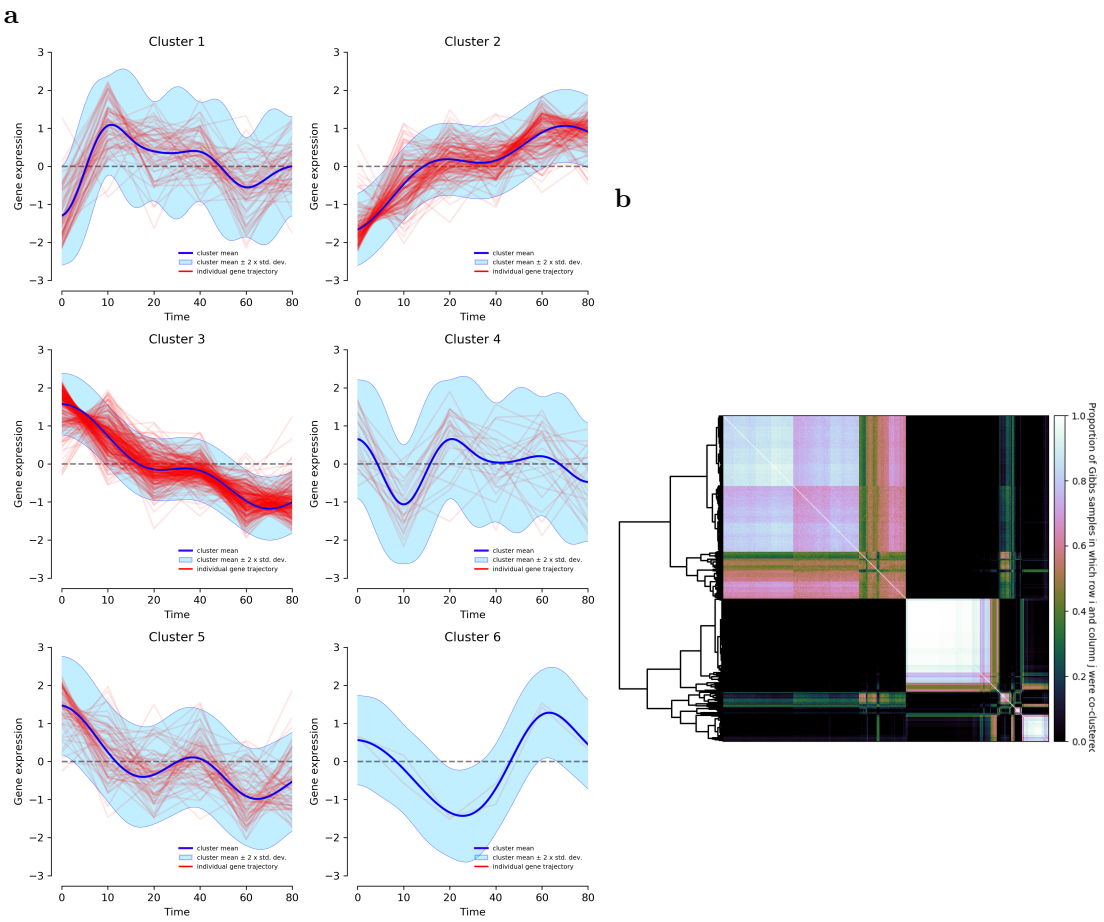


Figure 3.3: DPGP default graphical output for the mutant strain. (a) Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels. (b) Heatmap showing the proportion of sampled genes that cluster with every other gene. The gene order has been determined by complete linkage (using the `linkage` function within `SciPy`), and displayed as a dendrogram on the left.

### 3.1.1 Equivalent clusters

In order to identify corresponding equivalent cluster pairs, functions present in the code of DPGP software was adapted to export the posterior cluster mean expression trajectories (represented by the blue lines), since the standard output does not include them. Then, the Pearson correlation coefficient between all possible mean trajectory pairs was calculated. The resulting $r$ values are collected in table 3.1.

| | mut cluster 1 | mut cluster 2 | mut cluster 3 | mut cluster 4 | mut cluster 5 | mut cluster 6 |
|---|---|---|---|---|---|---|
| ctrl cluster 1 | **0.8867** | -0.0080 | 0.1128 | -0.3315 | -0.1528 | -0.6432 |
| ctrl cluster 2 | 0.2599 | **0.7708** | -0.7663 | 0.2601 | -0.7118 | -0.3161 |
| ctrl cluster 3 | -0.7122 | -0.6457 | 0.5704 | 0.1651 | **0.7097** | 0.4704 |
| ctrl cluster 4 | 0.0866 | -0.9131 | **0.9338** | -0.1773 | 0.8104 | -0.1514 |
| ctrl cluster 5 | -0.4718 | **0.7939** | -0.8350 | 0.0442 | -0.6852 | 0.6682 |
| ctrl cluster 6 | -0.8280 | 0.0777 | -0.1538 | 0.0047 | 0.0398 | **0.8915** |

Table 3.1: Pearson correlation coefficient between all possible pairs of posterior cluster mean expression level. Clusters obtained from the control strain are represented as rows, and clusters obtained from the mutant strain are represented as columns. Highlighted values are the highest values from a column-wise comparison, which correspond to the cluster pair with the most similar mean expression trajectory.

By performing a column-wise comparison, the cluster pairs with the highest Pearson correlation coefficient (highlighted values) were selected, and each pair was plotted with the control and mutant clusters side by side, mirroring the pairing done by McDowell *et al.* in figure 3.1 with the pointing

arrows.

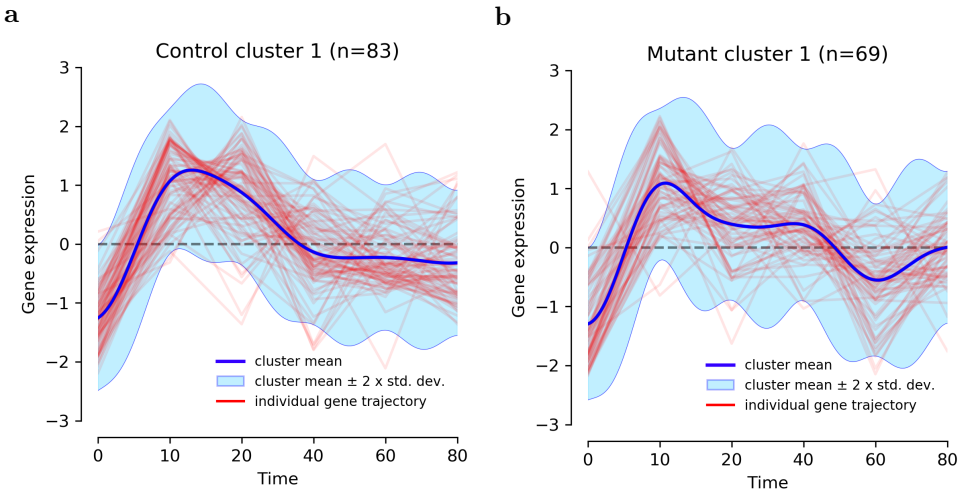


Figure 3.4: Equivalent cluster pair: control cluster 1 (a) and mutant cluster 1 (b). Red lines are standardized $\log_2$ fold change of single gene expression trajectories. Blue lines are the posterior cluster mean expression levels.

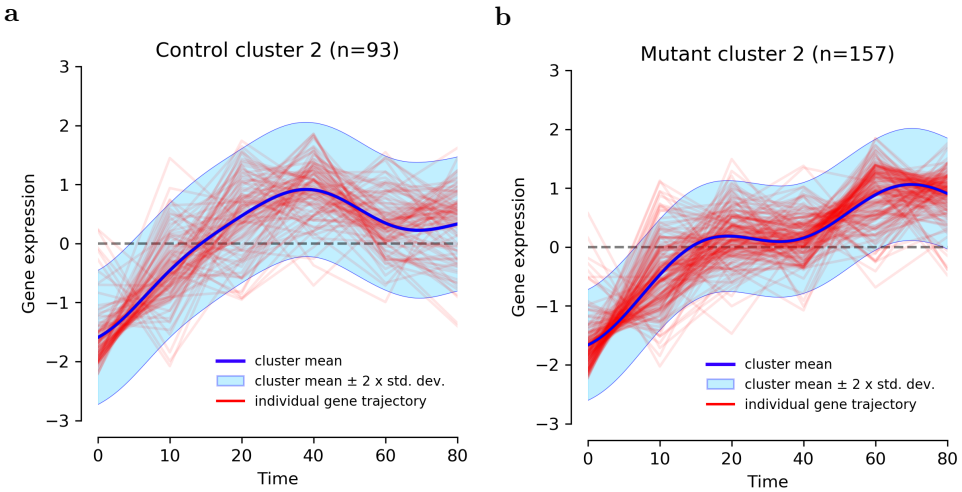


Figure 3.5: Equivalent cluster pair: control cluster 2 (a) and mutant cluster 2 (b). Red lines are standardized $\log_2$ fold change of single gene expression trajectories. Blue lines are the posterior cluster mean expression levels.

Figures 3.4 and 3.5 show a very consistent pairing to the one in the paper ($r = 0.8867$, $p = 5.35 \times 10^{-169}$, and $r = 0.7708$, $p = 1.38 \times 10^{-99}$ respectively), with a fast spike in expression levels followed by downregulation to base level for the cluster 1 pair, and a somewhat steady upregulation for the cluster 2 pair.

Figure 3.6 starts to diverge from the analysis in the paper, because mutant cluster 5 was also identified as the equivalent one ($r = 0.7097$, $p = 8.57 \times 10^{-78}$), but mutant cluster 4 could not since it does not have even the second-highest Pearson correlation coefficient ($r = 0.1651$), unlike what McDowell *et al.* reported (see figure 3.1 panels H, I, and L).

Figure 3.7 has the highest correlation of all pairings ($r = 0.9338$, $p = 1.91 \times 10^{-224}$), showing a steady downregulation trajectory over time.

Figure 3.8 shows the most inconsistent pairing compared to the published one. McDowell *et al.* reported that control cluster 5 does not have an equivalent cluster among the mutant ones, but there are clusters with a clear similar trajectory. Moreover, the way they evaluated this equivalence is by

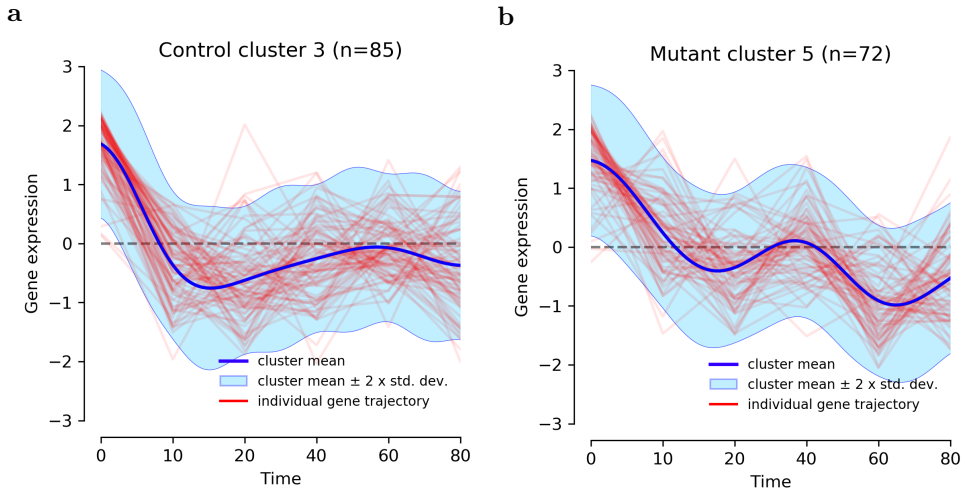


Figure 3.6: Equivalent cluster pair: control cluster 3 (a) and mutant cluster 5 (b). Red lines are standardized $\log_2$ fold change of single gene expression trajectories. Blue lines are the posterior cluster mean expression levels.

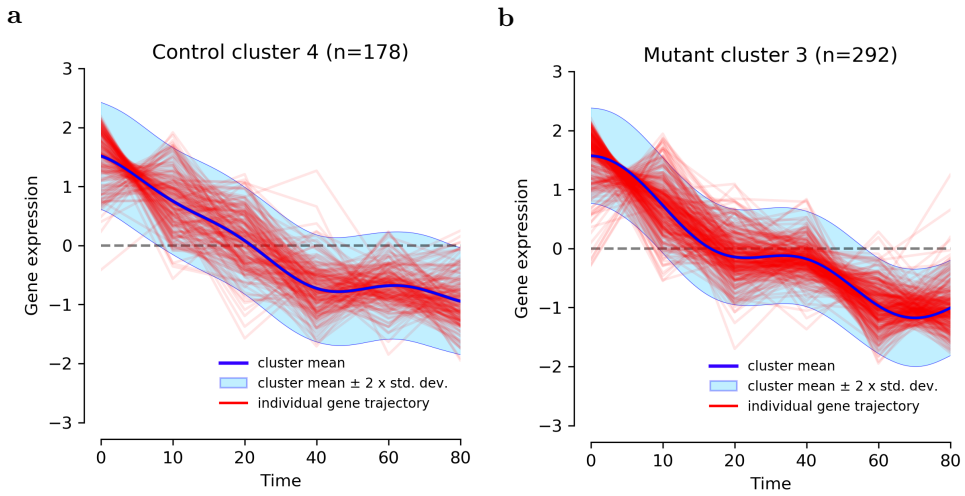


Figure 3.7: Equivalent cluster pair: control cluster 4 (a) and mutant cluster 3 (b). Red lines are standardized $\log_2$ fold change of single gene expression trajectories. Blue lines are the posterior cluster mean expression levels.

computing the Pearson correlation coefficient and selecting the highest one among a series of possible pairs; even in the case all $r$ values were very low, or even close to -1 (denoting a complete opposite behaviour), by selecting the highest value they could still get the closest possible expression trajectory. For these reasons, the pairing shown in figure 3.8 is reasonable, as also proven by the statistic ($r = 0.7939$, $p = 1.17 \times 10^{-109}$), which is not even the lowest among all equivalent pairs.

Lastly, figure 3.9 shows another quite consistent pairing ($r = 0.8915$, $p = 2.33 \times 10^{-173}$), characterized by a dip in expression levels towards the middle time points, returning to baseline levels at later stages. Despite this observation, the very low number of genes assigned to these clusters makes them poorly represented clusters that may not correspond to true groups in the dataset. In fact, the two heatmaps – figure 3.2.b for the control, and figure 3.3.b for the mutant – show 5 and 3 major groups respectively.

### 3.1.2 Cluster switching

McDowell *et al.* then proceeded with the testing of the statistical significance of cluster switching among genes, by calculating the Fisher's exact test. To do so, the number of genes that belong to all possible

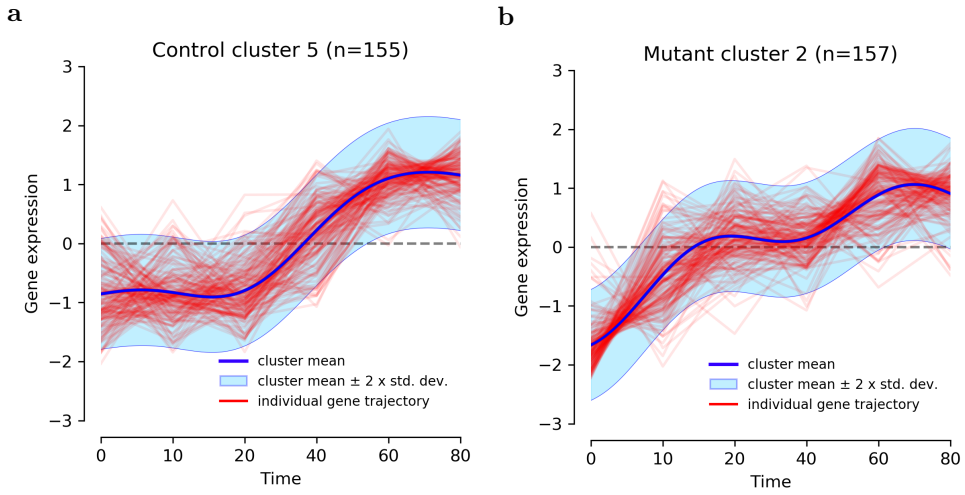


Figure 3.8: Equivalent cluster pair: control cluster 5 (a) and mutant cluster 2 (b). Red lines are standardized $\log_2$ fold change of single gene expression trajectories. Blue lines are the posterior cluster mean expression levels.

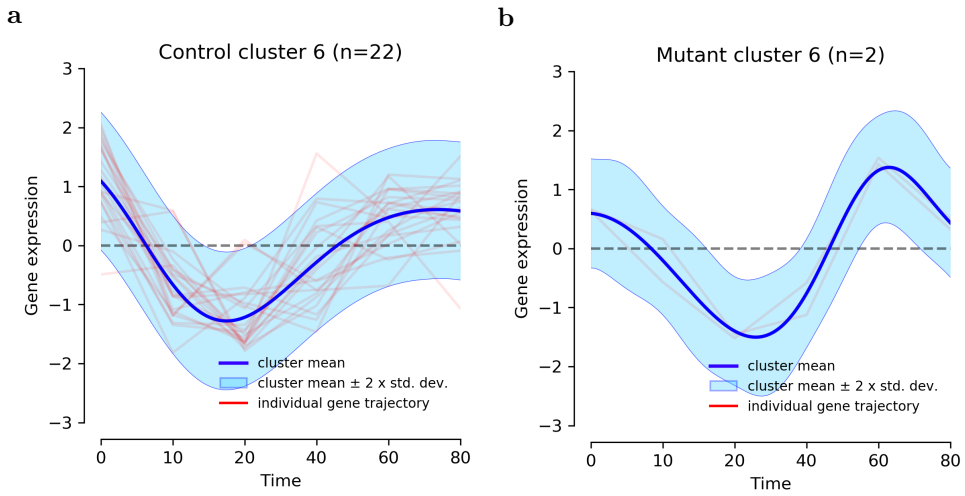


Figure 3.9: Equivalent cluster pair: control cluster 6 (a) and mutant cluster 6 (b). Red lines are standardized $\log_2$ fold change of single gene expression trajectories. Blue lines are the posterior cluster mean expression levels.

cluster pairing was computed by looking at the intersection of the list of gene labels in each control cluster *vs.* each mutant one. Table 3.2 is the resulting count table, where the highlighted cells correspond to the equivalent gene trajectory behaviour as assessed in section 3.1.1.

| | mut cluster 1 | mut cluster 2 | mut cluster 3 | mut cluster 4 | mut cluster 5 | mut cluster 6 |
|---|---|---|---|---|---|---|
| ctrl cluster 1 | **30** | 44 | 7 | 1 | 1 | 0 |
| ctrl cluster 2 | 11 | **64** | 12 | 2 | 3 | 1 |
| ctrl cluster 3 | 0 | 3 | 56 | 8 | **18** | 0 |
| ctrl cluster 4 | 11 | 10 | **125** | 3 | 29 | 0 |
| ctrl cluster 5 | 17 | **35** | 78 | 8 | 16 | 1 |
| ctrl cluster 6 | 0 | 1 | 14 | 2 | 5 | **0** |

Table 3.2: Number of genes belonging to both control and mutant clusters. Each value represents the same list of genes that are assigned to that particular cluster pair. This basically is a contingency table of all control clusters *vs.* all mutant clusters. Highlighted values correspond to the number of genes that are clustered in the equivalent mutant strain cluster (as identified by Pearson correlation, see 3.1), meaning that they maintain the same expression behaviour over time.

It is worth mentioning that the number of genes that belong to a clusters pair is not taken into account for the "equivalence" assessment – only the mean expression levels are. In fact, clusters can be considered

as "behaviours" of the genes, which are represented as trajectories of expression levels over time (*e.g.* initial upregulation followed by stationary expression, or upregulation followed by downregulation, or downregulation at later time, and so on). For instance, the equivalent pair for cluster 6 does not contain any gene, meaning that all those genes displayed a different behaviour if they came from the control or the mutant strain. In this particular case, McDowell *et al.* would not have considered this pair as equivalent clusters (joined by an arrow) because they had no co-clustered genes – analogously to what they claimed for control cluster 5 – but, by considering clusters as model behaviours, it is logical to assign to each cluster its equivalent. Moreover, a 2 by 2 contingency table has to be generated in order to compute the Fisher's exact test, so there have to be two categorical variables to count, in this case belonging to the control cluster *vs.* to the mutant cluster in each pair of clusters.

For each of the equivalent cluster pair, table 3.2 was used to calculate a contingency table, which I then used to compute the Fisher's exact test. Because the `fisher_exact` function within the same `SciPy` module was used, the statistic produced is not the same as the standard function implemented in R: the `SciPy` function calculates the odds ratio, and the R function the conditional maximum likelihood estimate.

A final summary table 3.3 recapitulates the statistical analysis on the equivalent gene trajectory pairs. By taking into account the number of genes that get assigned to each cluster, the first four pairings have a statistically significant relationship, but cluster switching for the control cluster 5 pair is not significant, as opposed to what reported in the paper ($p \leq 2.2 \times 10^{-16}$). The number of genes that display a different dynamic was also reported – meaning they belong to any other cluster except the one deemed equivalent – because McDowell *et al.* used the number of total genes with different dynamic to compare their results with the ones previously published by Sharma *et al.*. In this regard, the results in this study are reasonably similar to the one published, despite the discrepancies discussed so far: the total number of genes that show a different expression trajectory in the mutant strain group is 344, *vs.* 372 reported in the paper.

| **Control strain** | **Mutant strain** | **Pearson corr r** | **Pearson p value** | **Odds ratio** | **FET p value** | **Different dynamic genes** |
|---|---|---|---|---|---|---|
| ctrl cluster 1 | mut cluster 1 | 0.8867 | $5.35 \times 10^{-169}$ | 7.17 | $3.08 \times 10^{-11}$ | 53 (63.9%) |
| ctrl cluster 2 | mut cluster 2 | 0.7708 | $1.38 \times 10^{-99}$ | 10.20 | $4.09 \times 10^{-22}$ | 29 (31.2%) |
| ctrl cluster 3 | mut cluster 5 | 0.7097 | $8.57 \times 10^{-78}$ | 2.37 | $5.92 \times 10^{-3}$ | 67 (78.8%) |
| ctrl cluster 4 | mut cluster 3 | 0.9338 | $1.91 \times 10^{-224}$ | 3.83 | $5.31 \times 10^{-13}$ | 53 (29.8%) |
| ctrl cluster 5 | mut cluster 2 | 0.7939 | $1.17 \times 10^{-109}$ | 0.81 | $3.94 \times 10^{-1}$ | 120 (77.4%) |
| ctrl cluster 6 | mut cluster 6 | 0.8915 | $2.33 \times 10^{-173}$ | 0.00 | 1.00 | 22 (100.0%) |

Table 3.3: Summary of the statistical analysis of the equivalent cluster pairs. Pearson correlation $r$ and $p$ values are calculated with the `pearsonr` function of the Python `SciPy` module (Virtanen *et al.*, 2020). Odds ratio and Fisher's exact test (FET) $p$ value are calculated using the `fisher_exact` function within the same `SciPy` module. The number of genes with different dynamic is calculated using the relative contingency table, and expressed as a percentage of the total number of genes in that cluster.

Although these results would suggest otherwise, control cluster 5 was examined more closely because the authors focused on it. Of the 155 genes belonging to it, 120 (77.4%) exhibit a different dynamic in the mutant strain. Since this cluster has an up-regulated trajectory over time, the trajectory of genes that have an inverted dynamic – downregulated trajectory – was evaluated, restricting the previous 120 genes to 94 (60.7%), of which 78 belonged to mutant cluster 3, and 16 to mutant cluster 5. In the paper, the authors got very similar numbers: out of 141 genes in control cluster 5 (which all of them had different dynamic in the mutant strain), 89 (63.1%) had an inverted dynamic in gene expression trajectory, of which 72 belonged to mutant cluster 3, and 17 to mutant cluster 5.

All together, these findings suggest that DPGP produces very consistent results, in line with previously published observations.

### 3.1.3 Heatmaps

To conclude the analysis, a Python script recreated the heatmaps as shown in the original paper, by adapting functions present in the DPGP code to produce the images without the dendrograms, and adding the possibility to change the order of the genes.

Figure 3.10 shows the resulting heatmaps for the control (panel a) and mutant (panel b) strains but, since the latter is not present in the original paper, only the former is taken into consideration. The control strain heatmap shows a slightly different proportion matrix, in particular in the "perpendicular stripes" regions that seem to be displaced because of a different order. Unfortunately, without the dendrogram or the list of genes produced by the linkage, it is not possible to assert this for certain. Another possibility is that DPGP may have been run using a different seed, hence the different results, which in turn would be reflected in the heatmap. Alternatively, it may also be due to the linkage algorithm: in the caption of the figure in the original paper, the authors state that they used Ward's linkage, but in the DPGP code, the function that produces the gene order performs a complete linkage, which is the one used in this study.

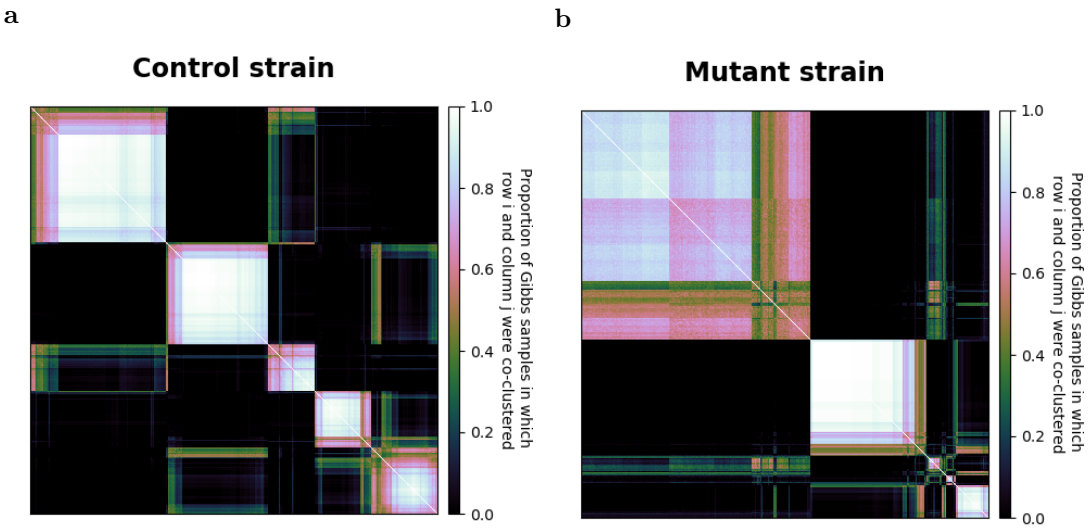


Figure 3.10: Heatmaps for the control (a) and mutant (b) strains, showing the proportion of sampled genes that cluster with every other gene. The gene order has been determined by complete linkage (using the linkage function within SciPy).

Lastly, figure 3.11 shows the heatmaps for the two strains, where the order of the genes is the same for both the control (panel a) and the mutant (panel b). Although the images match those from the original text, no conclusive interpretation could be made, other than the pattern disruption: had the genes kept the same gene expression trajectory in the mutant strain, they would have had similar sampled proportions, resulting in "whiter blocks" in the same positions. The fact that the patterns are completely disrupted but in very few and small sections, means that the two clustered strains have distinct gene groupings.

## 3.2 Using DPGP on simulated data

The purpose of using simulated datasets was to assess DPGP performance in a controlled setting environment with known expectations. Because the end goal of this project is building a pipeline for the analysis of longitudinal RNA-seq data, prior to the clustering and data visualization steps, a differential

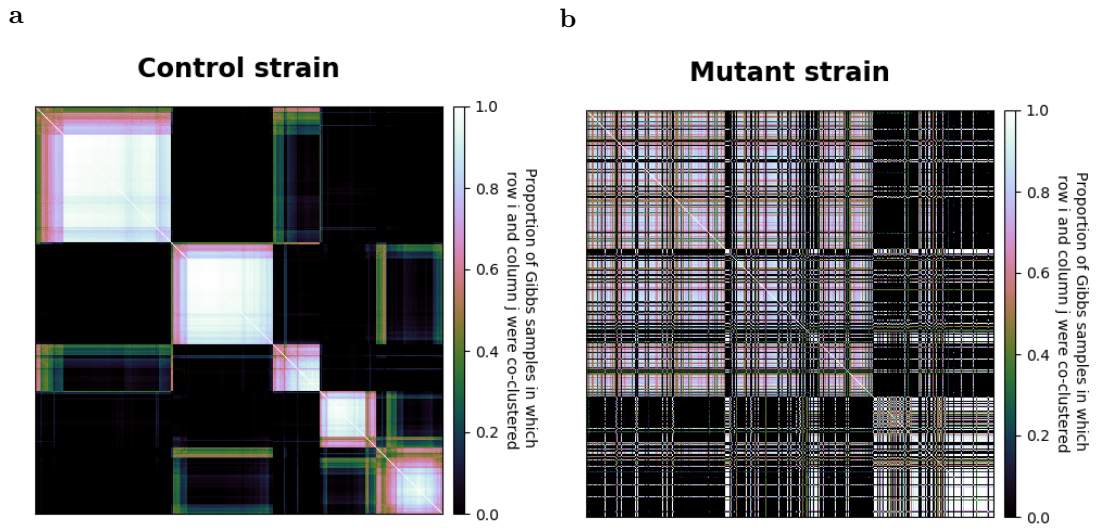


Figure 3.11: Heatmaps showing the proportion of sampled genes that cluster with every other gene. For the control strain (a), the gene order has been determined by complete linkage (using the linkage function within SciPy). For the mutant strain (b), it has been used the same gene order as the control strain.

expression analysis is conducted using `ImpulseDE2` (Fischer *et al.*, 2018). For this reason, DPGP was tested on simulated datasets that were first created and analysed with it.

Two synthetic datasets were created: a *linear dataset*, with 400 genes whose trajectories follow a linear progression, and a *mixed dataset*, with 200 genes that follow a constant, impulse, and linear trajectory each, for a total of 600 genes (for a detailed description see section 2.4.1).

### 3.2.1 Linear dataset

Since the number of simulated genes in the linear dataset is lower than the biological dataset described in section 2.3.1, DPGP was first run on the whole dataset, to test its ability to single out noisy expression data.

Figures 3.12, 3.13, 3.14, and 3.14 show the graphical output of the clusters produced by DPGP when run with shape hyperparameter $\alpha = 12$, 9, 6, and 3 respectively. In all four cases, the clusters appear to describe two main groups – linear down-regulation, and linear up-regulation – in a quite consistent manner. All other clusters seem to be representative of very noisy data, or of many linear gene expression trajectory centred around a "stationary" expression level – an almost horizontal mean trajectory.

The DE analysis performed with `ImpulseDE2` is meant to compare gene expression against a Null model represented by a constant expression level, which should produce a more significant subset of genes with meaningful gene expression variations over time.

DPGP was then run again on the DEGs identified from the linear dataset, using the same shape hyperparameters. The output is displayed as figures 3.16, 3.17, 3.18, and 3.19. As expected, the number of clusters is lower because the DE analysis excluded all genes with a fold change over time that approaches 0.

In regard to the variability allowed with the shape hyperparameter, DPGP identified in each case the two major groups – up-regulation and down-regulation clusters. Below $\alpha = 9$, the confidence intervals (light blue area) got increasingly wider at the expense of the number of clusters. Based on this result, the default ($\alpha = 12$) value is likely to be the optimal in order to capture enough variability of the data without loosing too much information.

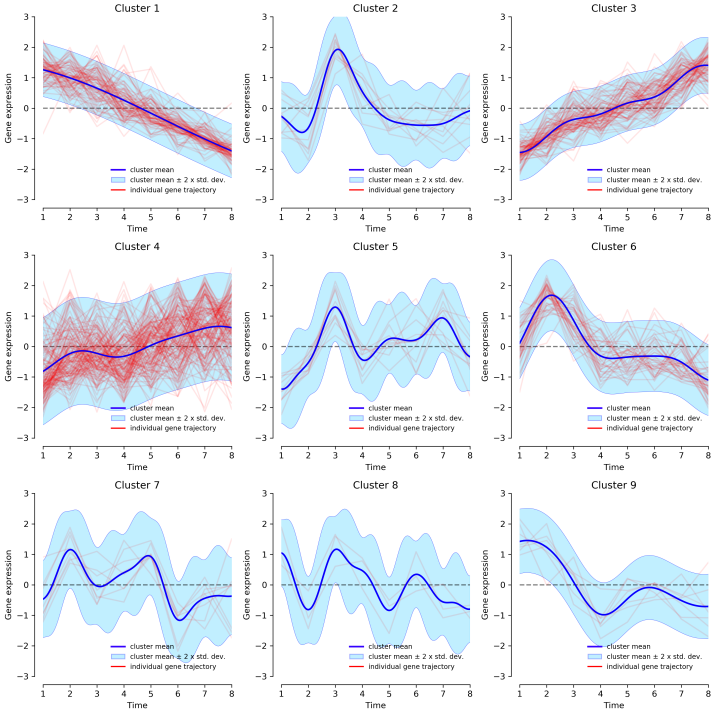


Figure 3.12: Graphical output for the complete linear dataset, and DPGP run with shape $\alpha = 12$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

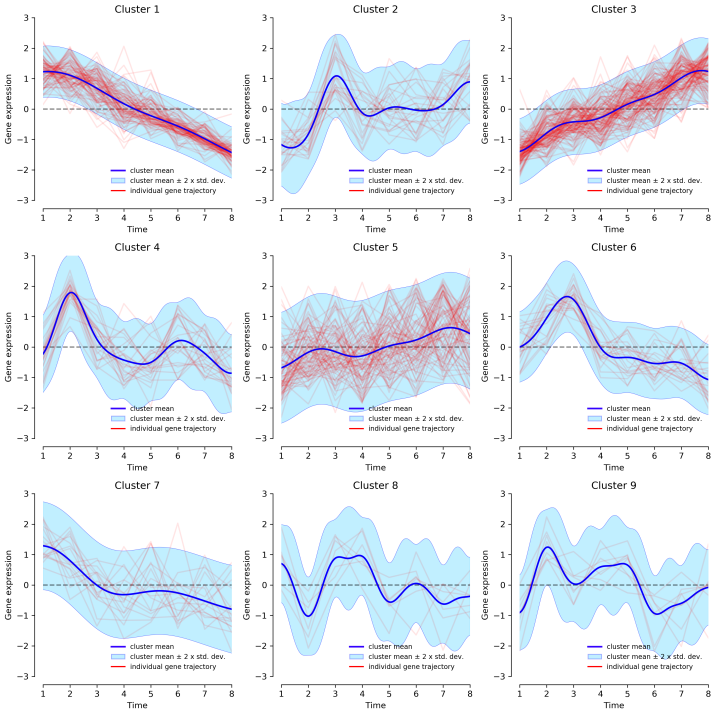


Figure 3.13: Graphical output for the complete linear dataset, and DPGP run with shape $\alpha = 9$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

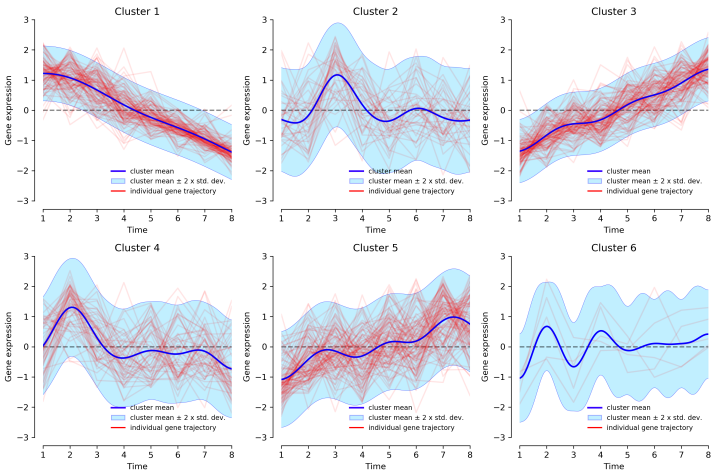


Figure 3.14: Graphical output for the complete linear dataset, and DPGP run with shape $\alpha = 6$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

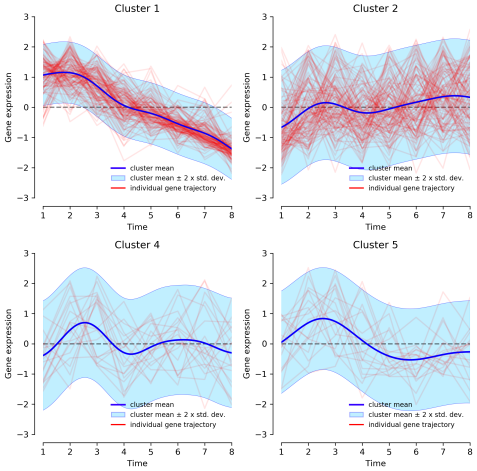


Figure 3.15: Graphical output for the complete linear dataset, and DPGP run with shape $\alpha = 3$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

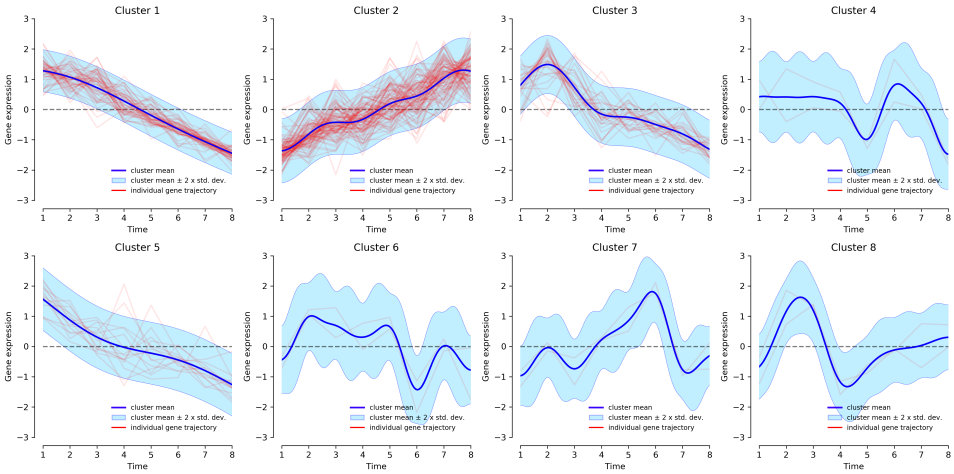


Figure 3.16: Graphical output for the DEGs from linear dataset, and DPGP run with shape $\alpha = 12$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

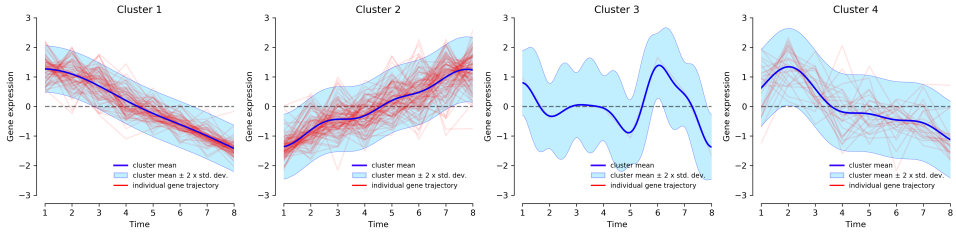


Figure 3.17: Graphical output for the DEGs from linear dataset, and DPGP run with shape $\alpha = 9$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

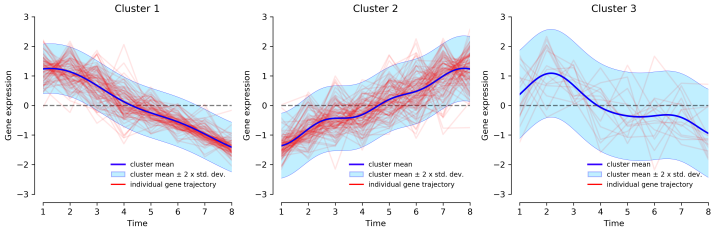


Figure 3.18: Graphical output for the DEGs from linear dataset, and DPGP run with shape $\alpha = 6$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

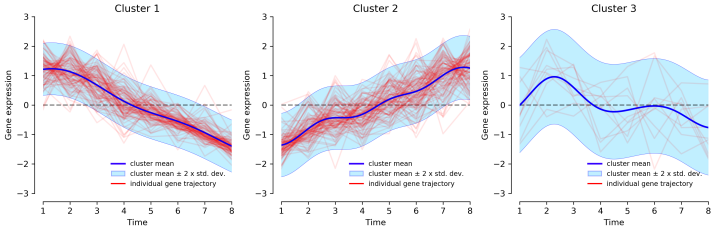


Figure 3.19: Graphical output for the DEGs from linear dataset, and DPGP run with shape $\alpha = 3$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

### 3.2.2 Mixed dataset

Next, DPGP was tested on a more complex dataset, to test its ability to recognize different temporal patterns in gene expression levels. The same analysis conducted on the linear dataset was also performed on this mixed dataset: first on the whole dataset, and then on the DEGs only, and in both cases with increasingly low values for the shape hyperparameter.

Figures 3.20, 3.21, 3.22, and 3.22 show the graphical output of the clusters produced by DPGP when run with shape hyperparameter $\alpha = 12$, 9, 6, and 3 respectively. Unlike the previous dataset, it is difficult to make interpretations based on this output due to the high number of clusters and the variability of the modelled expression patterns. The two clusters that stand out the most and are among the most represented are once again the ones that show an up-regulated and down-regulated gene trajectory.

DPGP was then run on the DEGs identified from the same mixed dataset, using the same shape hyperparameters. The output is still of harder interpretation, but it allows for a much clearer picture: when comparing this output (figures 3.24, 3.25, 3.26, and 3.27) with the previous one, most of the clusters seem to be missing, meaning that they mostly represented noisy data. However, the two most predominant clusters are still showing steady up-regulation and down-regulation trajectories.

Lastly, taking advantage of the known function used to generate each gene expression array, two tables were compiled using a Python script, one for the clusters of full mixed dataset (table 3.5), and another for the clusters of the DEGs (table 3.4), showing how many genes belong to each cluster for each function used to produce the data. From an initial comparison, it is clear that the DE analysis excluded almost all constant genes, and halved the other two "sub-categories": from the starting 200 per function, only 3 constant DEGs remain (1.5%), 104 impulse (52%), and 94 linear (47%).

DEG dataset, shape 12

| cluster / function | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **constant** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **impulse** | 0 | 7 | 18 | 8 | 20 | 12 | 5 | 9 | 14 | 10 | 1 |
| **linear** | 1 | 3 | 10 | 11 | 44 | 12 | 4 | 0 | 7 | 2 | 0 |

DEG dataset, shape 9

| cluster / function | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| **constant** | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| **impulse** | 0 | 26 | 26 | 12 | 9 | 10 | 19 | 2 |
| **linear** | 1 | 17 | 51 | 9 | 0 | 1 | 15 | 0 |

DEG dataset, shape 6

| cluster / function | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **constant** | 2 | 1 | 0 | 0 | 0 |
| **impulse** | 0 | 34 | 32 | 14 | 24 |
| **linear** | 1 | 30 | 58 | 2 | 3 |

DEG dataset, shape 3

| cluster / function | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **constant** | 1 | 2 | 0 | 0 | 0 |
| **impulse** | 0 | 48 | 31 | 16 | 9 |
| **linear** | 1 | 32 | 58 | 2 | 1 |

Table 3.4: Partition of ImpulseDE2-simulated DEGs among DPGP clusters. The background gradient highlights the results of a row-wise comparison: the darker the colour, the higher number of genes generated from that function belong to a specific cluster.

By looking in more detail at table 3.4, it is worth noting that no cluster predominantly represents the impulse or the linear generated genes: at best, only one has a proportion of impulse:linear equal to about 1:2 (clusters 5, 3, 3, and 3 in figures 3.24, 3.25, 3.26, and 3.27 respectively).

Overall, these observations suggest that the assumptions made by the `ImpulseDE2` tool to generate the dataset may be incompatible with the ones made by DPGP.

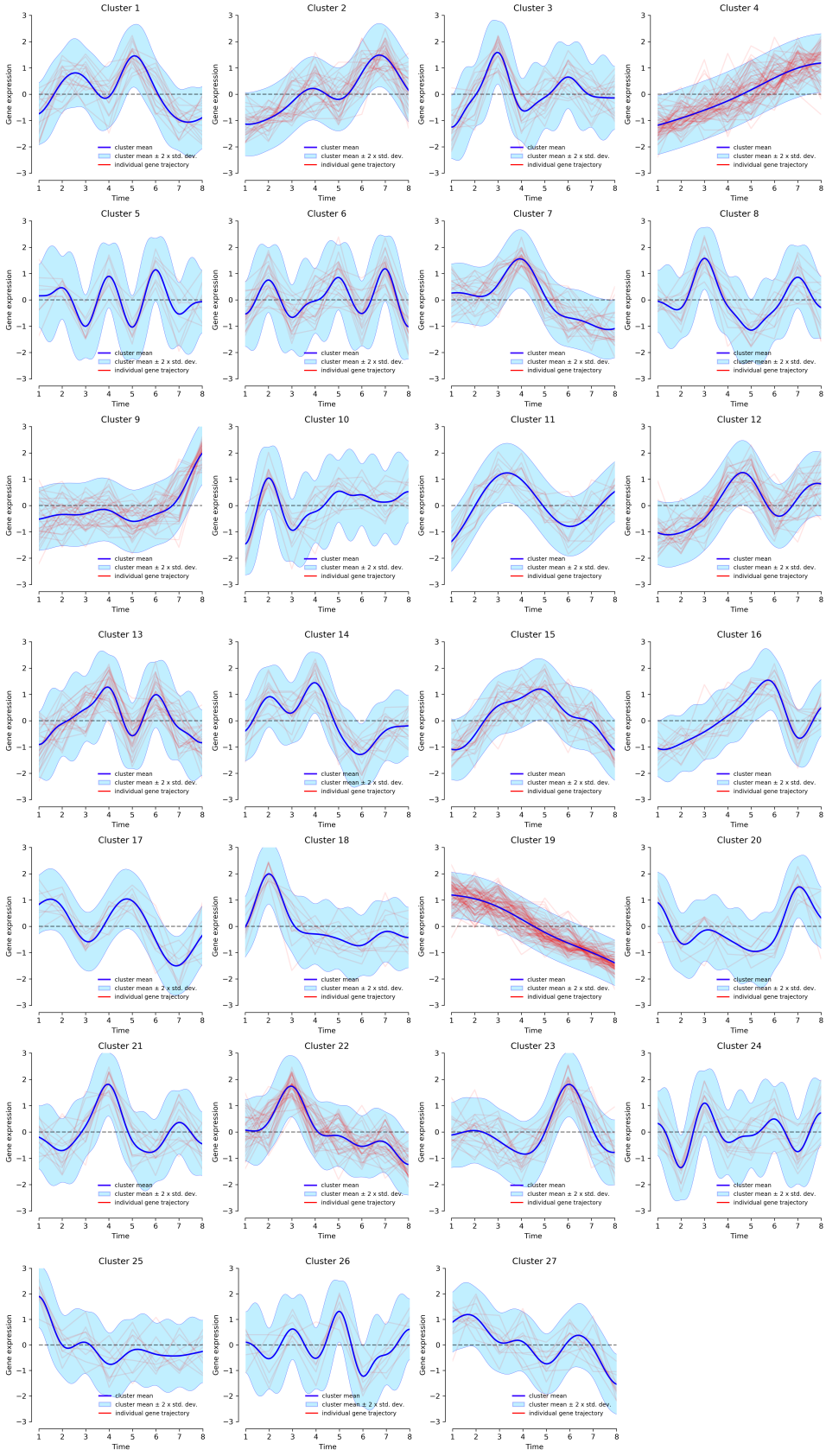


Figure 3.20: Graphical output for the complete mixed dataset, and DPGP run with shape $\alpha = 12$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

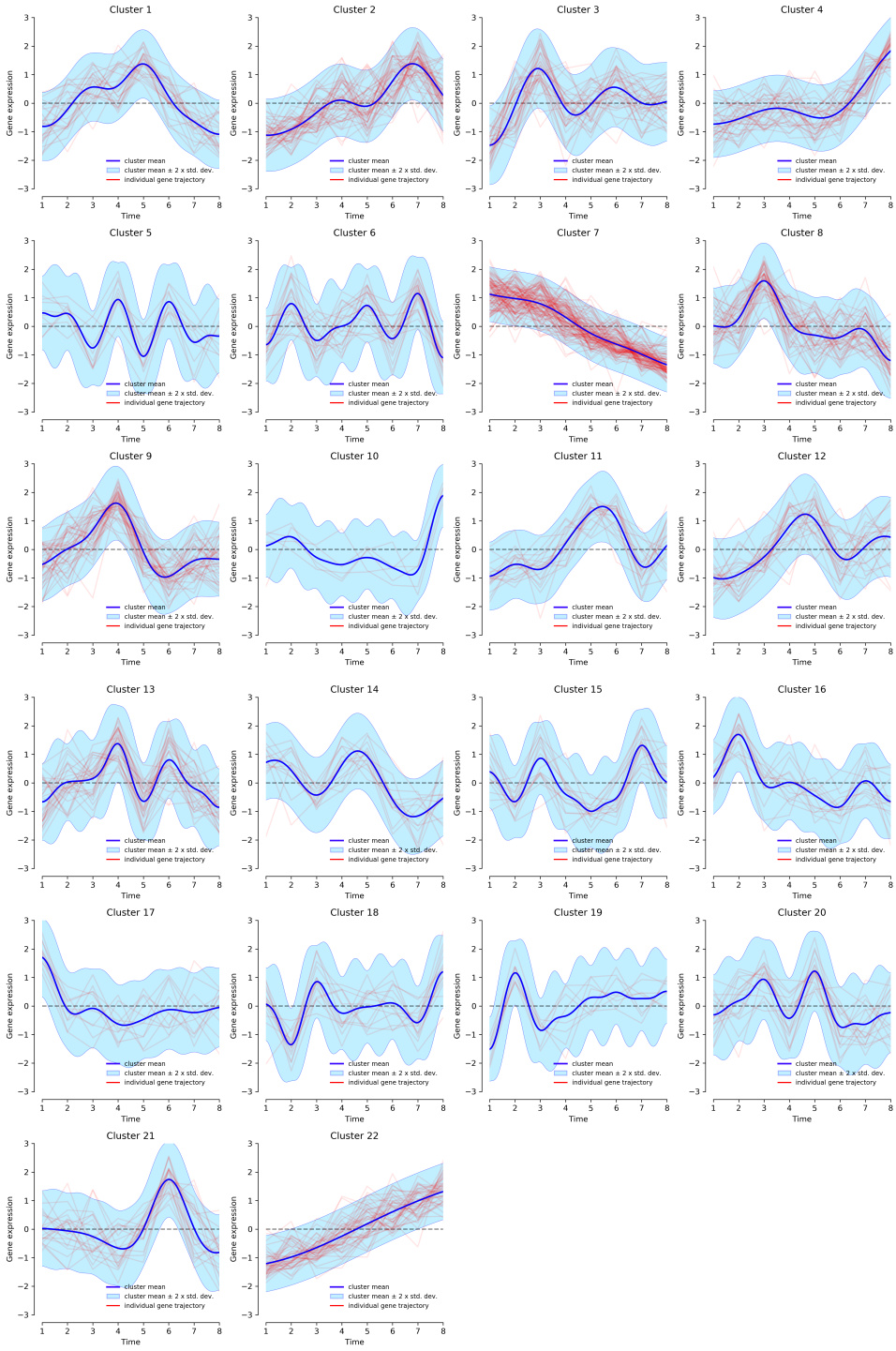


Figure 3.21: Graphical output for the complete mixed dataset, and DPGP run with shape $\alpha = 9$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

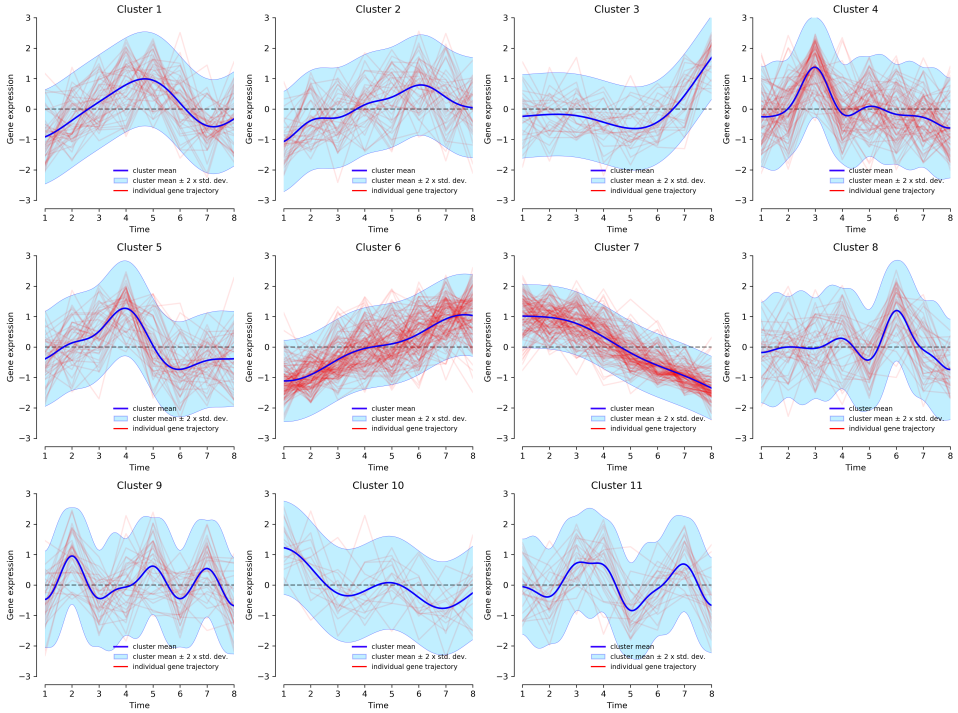


Figure 3.22: Graphical output for the complete mixed dataset, and DPGP run with shape $\alpha = 6$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

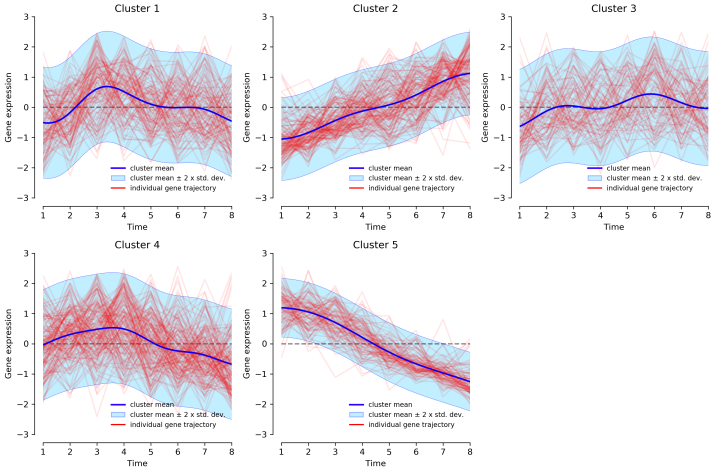


Figure 3.23: Graphical output for the complete mixed dataset, and DPGP run with shape $\alpha = 3$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

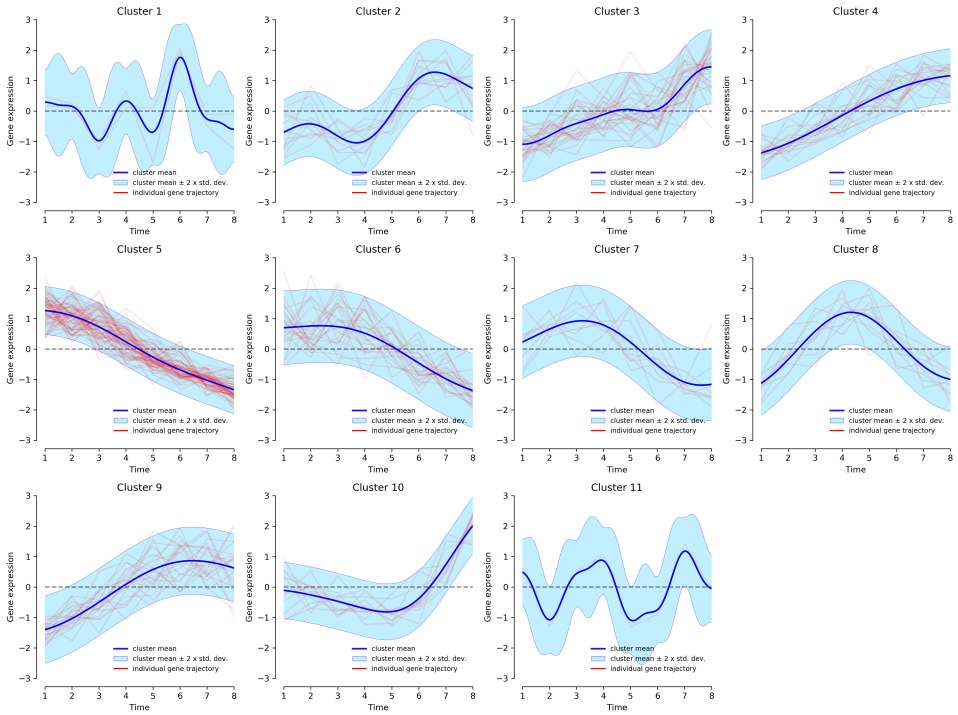


Figure 3.24: Graphical output for the DEGs from mixed dataset, and DPGP run with shape $\alpha = 12$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

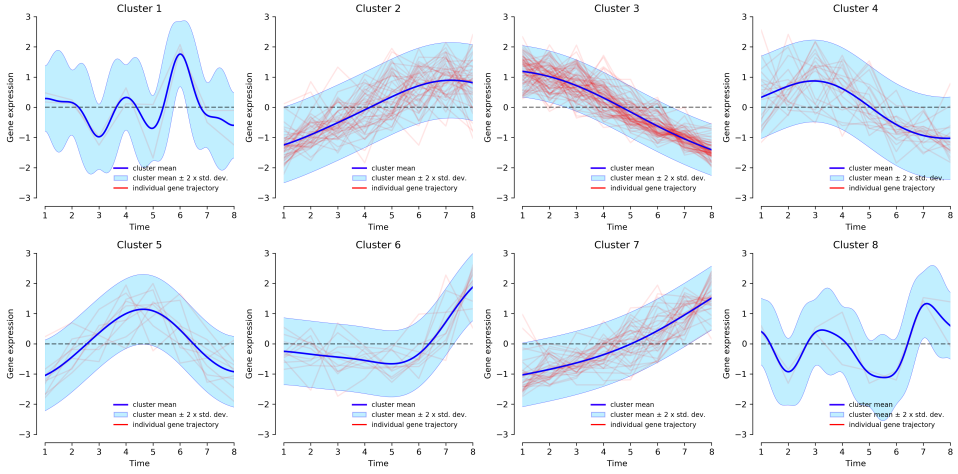


Figure 3.25: Graphical output for the DEGs from mixed dataset, and DPGP run with shape $\alpha = 9$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

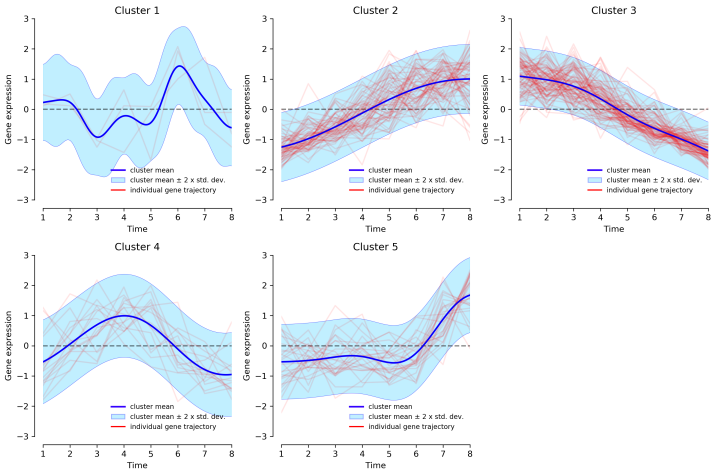


Figure 3.26: Graphical output for the DEGs from mixed dataset, and DPGP run with shape $\alpha = 6$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

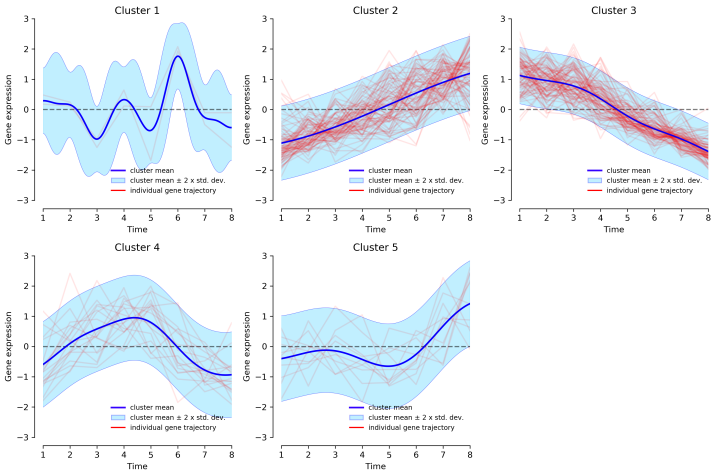


Figure 3.27: Graphical output for the DEGs from mixed dataset, and DPGP run with shape $\alpha = 3$. Gene expression trajectories for each cluster. Red lines are standardized $\log_2$ fold change of single gene expression. Blue lines are the posterior cluster mean expression levels.

Complete dataset, shape 12

| cluster / function | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **constant** | 11 | 8 | 13 | 6 | 7 | 8 | 12 | 10 | 4 | 5 | 8 | 10 | 15 | 11 | 7 | 7 | 3 | 4 | 3 | 4 | 8 | 12 | 9 | 6 | 5 | 4 | 0 |
| **impulse** | 6 | 14 | 2 | 29 | 1 | 3 | 9 | 2 | 19 | 4 | 2 | 8 | 13 | 2 | 11 | 4 | 4 | 6 | 28 | 3 | 3 | 14 | 5 | 3 | 2 | 0 | 3 |
| **linear** | 1 | 11 | 3 | 27 | 1 | 10 | 5 | 1 | 11 | 3 | 3 | 11 | 1 | 3 | 5 | 4 | 2 | 1 | 53 | 2 | 3 | 13 | 6 | 4 | 3 | 4 | 9 |

Complete dataset, shape 9

| cluster / function | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **constant** | 11 | 10 | 19 | 5 | 6 | 10 | 7 | 16 | 26 | 2 | 9 | 9 | 16 | 8 | 8 | 6 | 6 | 6 | 2 | 9 | 9 | 0 |
| **impulse** | 11 | 18 | 9 | 17 | 2 | 4 | 35 | 13 | 8 | 2 | 7 | 8 | 10 | 3 | 5 | 4 | 3 | 3 | 3 | 2 | 6 | 27 |
| **linear** | 2 | 21 | 5 | 13 | 2 | 8 | 60 | 13 | 9 | 1 | 3 | 8 | 4 | 3 | 3 | 5 | 2 | 5 | 3 | 6 | 6 | 18 |

Complete dataset, shape 6

| cluster / function | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **constant** | 25 | 21 | 7 | 41 | 32 | 11 | 10 | 20 | 12 | 9 | 12 |
| **impulse** | 18 | 7 | 12 | 16 | 10 | 62 | 41 | 11 | 9 | 6 | 8 |
| **linear** | 5 | 13 | 9 | 20 | 7 | 51 | 65 | 11 | 13 | 1 | 5 |

Complete dataset, shape 3

| cluster / function | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **constant** | 60 | 20 | 46 | 68 | 6 |
| **impulse** | 33 | 62 | 20 | 53 | 32 |
| **linear** | 27 | 55 | 24 | 38 | 56 |

Table 3.5: Partition of all ImpulseDE2-simulated genes among DPGP clusters. The background gradient highlights the results of a row-wise comparison: the darker the colour, the higher number of genes generated from that function belong to a specific cluster.

# Chapter 4

# Discussion

## 4.1 Conclusions

This study was conducted as part of a bigger project aiming to develop a pipeline for the analysis of longitudinal RNA-seq data, as generated by HTTr assays. In this regard, one of the main challenges faced by the scientific community is that there is still no consensus on how to conduct this type of analysis, and which tools are best suited in this context. Hence the importance of revising some of the most used and popular tools.

Unlike other tools that perform differential expression (DE) analysis, the focus of this dissertation was DPGP, a tool specifically designed for clustering genomic features over time. This choice was based on three main reasons:

1. It had been decided that the DE step in the pipeline will be performed using `ImpulseDE2`;
2. DPGP is among the most popular tools, based on an initial review of the Literature;
3. Finally, DPGP uses Bayesian inference to infer the parameters that fit the data from the data itself by using model agnostic approaches. In this case, DPGP uses a Dirichlet process (DP) to determine the number of clusters, combined with a Gaussian process (GP) for modelling expression levels over time.

This study had two main objectives:

1. Reproduce the original results from the DPGP paper (McDowell *et al.*, 2018) to assert its ability of generating consistent results, and simultaneously evaluate its assumptions;
2. Assess DPGP behaviour on simulated datasets with known characteristics and expected outcomes.

Despite minor discrepancies, the results on the biological dataset originally published by Sharma *et al.* gave very consistent results (see section 3.1), showing that it is sufficient to provide the same parameters and seed to be able to generate the same outcomes.

On the contrary, the analysis conducted on the simulated datasets (see section 3.2), highlighted how the assumptions used to generate the gene expression levels over time may affect the performance of the tool, which can lead to inconclusive interpretations.

Overall, DPGP has proven to be a potentially good addition for an analytical pipeline, and it could be further investigated.

## 4.2 Future directions

So far DPGP has been used in this study as a stand-alone tool, but it still needs to be evaluated in comparison to other tools with similar scope and characteristics, and in the context of a full pipeline for the analysis of RNA-seq time series data. For that purpose, many possible combinations of tools will need to be taken into consideration, *e.g.* employing different pre-processing practices, or DE analysis with different tools, etc.

Also, this study did not focus on the output of DPGP and on how it can be used in the downstream steps of a potential pipeline, such as annotations and pathway analysis, in order to extrapolate a biological interpretation from the data.

Finally, DPGP (and the pipeline) has yet to be tested on data produced from HTTr assays like the ones used by Unilever's R&D department, and how valuable its interpretation is when compared to other "common" pipelines that treat time as any other categorical variable.

# Bibliography


Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biology*, **11**(10), R106.

Bethesda, MD. (2010). *Entrez Programming Utilities Help*. National Center for Biotechnology Information (US), https://www.ncbi.nlm.nih.gov/books/NBK25501/.

Cock, P. J. A. *et al.* (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics (Oxford, England)*, **25**(11), 1422–1423.

Fischer, D. S. *et al.* (2018). Impulse model-based differential expression analysis of time course sequencing data. *Nucleic Acids Research*, **46**(20), e119.

Frazee, A. C. *et al.* (2014). Differential expression analysis of RNA-seq data at single-base resolution. *Biostatistics (Oxford, England)*, **15**(3), 413–426.

Gumienny, R. (2021). GEOparse: Python library to access Gene Expression Omnibus Database (GEO).

Jalba, M.-S. (2008). Three generations of ongoing controversies concerning the use of short acting beta-agonist therapy in asthma: A review. *The Journal of Asthma: Official Journal of the Association for the Care of Asthma*, **45**(1), 9–18.

Jüni, P. *et al.* (2004). Risk of cardiovascular events and rofecoxib: Cumulative meta-analysis. *Lancet (London, England)*, **364**(9450), 2021–2029.

Love, M. I. *et al.* (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, **15**(12), 550.

Lynch, C. *et al.* (2023). High-Throughput Screening to Advance In Vitro Toxicology: Accomplishments, Challenges, and Future Directions. *Annual Review of Pharmacology and Toxicology*.

McDowell, I. C. *et al.* (2018). Clustering gene expression time series data using an infinite Gaussian process mixture model. *PLoS computational biology*, **14**(1), e1005896.

Nagalakshmi, U. *et al.* (2008). The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science (New York, N.Y.)*, **320**(5881), 1344–1349.

Oh, V.-K. S. and Li, R. W. (2021). Temporal Dynamic Methods for Bulk RNA-Seq Time Series Data. *Genes*, **12**(3), 352.

Paquet, Y. H. Y. *et al.* (2023). Marray: Exploratory analysis for two-color spotted microarray data. Bioconductor version: Release (3.17).

Ritchie, M. E. *et al.* (2015). Limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, **43**(7), e47.

Shannon, P. T. *et al.* (2006). The Gaggle: An open-source software system for integrating bioinformatics software and data sources. *BMC bioinformatics*, **7**, 176.

Sharma, K. *et al.* (2012). The RosR transcription factor is required for gene expression dynamics in response to extreme oxidative stress in a hypersaline-adapted archaeon. *BMC genomics*, **13**, 351.

Smyth, G. *et al.* (2023). Limma: Linear Models for Microarray Data. Bioconductor version: Release (3.17).

Spies, D. *et al.* (2019). Comparative analysis of differential gene expression tools for RNA sequencing time course data. *Briefings in Bioinformatics*, **20**(1), 288–298.

Topa, H. and Honkela, A. (2018). GPrank: An R package for detecting dynamic elements from genome-wide time series. *BMC bioinformatics*, **19**(1), 367.

Unilever (accessed 15-08-2023). Alternatives to animal testing. https://www.unilever.com/planet-and-society/responsible-business/alternatives-to-animal-testing/.

Unilever (accessed 15-08-2023). Leading safety and environmental sustainability sciences. https://www.unilever.com/planet-and-society/safety-and-environment/leading-safety-and-environmental-sustainability-sciences/.

Van Norman, G. A. (2020). Limitations of Animal Studies for Predicting Toxicity in Clinical Trials: Part 2: Potential Alternatives to the Use of Animals in Preclinical Trials. *JACC. Basic to translational science*, **5**(4).

Vantini, M. *et al.* (2022). PairGP: Gaussian process modeling of longitudinal data from paired multi-condition studies. *Computers in Biology and Medicine*, **143**, 105268.

Villar, D. *et al.* (1998). Ibuprofen, aspirin and acetaminophen toxicosis and treatment in dogs and cats. *Veterinary and Human Toxicology*, **40**(3), 156–162.

Virtanen, P. *et al.* (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, **17**(3), 261–272.

Walker, M. G. *et al.* (1999). Prediction of gene function by genome-scale expression analysis: Prostate cancer-associated genes. *Genome Research*, **9**(12), 1198–1203.