Annotated bibliography

Alberto Locca

11th August 2023

1 Introduction

The overall goal of this project – for Unilever's internal scope – is developing a pipeline, using readily available tools, for the analysis of longitudinal RNA-seq data. These data are the results of *in vitro* High Throughput Transcriptomics (HTTr) assays for safety and risk assessment of chemicals, drugs, and products developed by Unilever, in a decade-long effort of developing new approaches that do not involve animal testing.

In order to achieve this goal, we first focused on reviewing the Literature for tools that would satisfy our requirements: we started from 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. We then focused on a second 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.*. Table 1 is a summary of all the tools of interest listed by Oh and Li. The tools listed in table 3 (see *Original table* column in table 1) have been excluded because they involve batch effects handling, which is not taken into account for the scope of our pipeline, but it may be included in a future implementation.

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 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, I added a *DOI* column for easy searching.

Next, I wanted to automate the process of understanding among these tools which are more suitable for our goal, in a more objective and programmatic way. The three crucial criteria that an ideal tool should have are:

- Datasets availability; ideally a tool should have been tested and work on publicly available datasets, which could be easily downloaded and are compatible to Unilever's HTTr assays data format (*i.e.* number of time points, biological and technical replicates, conditions, etc.).

- Code availability; as anticipated, the tool has to be readily available, but in order to understand its model assumptions and if it is suited for our applications, also its code must have been made available, with – ideally – all the code used to produce the authors' results in its original paper.
- Data reproducibility; most crucially, we must be able to obtain the same – if not, as similar as possible – results as the authors' in the original paper of the tool. It also falls under this criterion the ability of a tool to produce a reasonable or expected output when provided with toy or simulated datasets.

I took advantage of the reference list in the paper by Oh and Li, from which I extrapolated the DOIs for each of the relevant tools (as shown in table 1). I could then use this list of DOIs to cross-search 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). By doing this, I was trying to address the first point, *datasets availability*, but I also wanted to develop a metric to assess the goodness of a tool by looking at the number of citations (averaged by year) of each paper. The script `entrez.py` (see 3.1) accomplishes these two tasks and produces table 2, a revised version of table 1.

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: Paper metrics table obtained from table 1 after running the script `entrez.py` (see 3.1). Tools are listed in descending order by *Average yearly citations*. Some columns have been omitted for readability.

The principal shortcomings of my approach are the following:

1. NCBI E-Utilities can perform cross-searches among NCBI databases, but they need to be provided with the relative database ID (*i.e.* PMID to search the Pubmed database). DOI strings are extra fields in a Pubmed entry and can return unexpected results, like for the case of *PairGP* (last row in table 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. Number of citations – and by extent, number of yearly citations – 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 we are 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) correctly listing 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.

In spite of these problems, the script returned a reasonable outcome:

- Interestingly, 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 our HTTr assays involve non-periodical time course data, we can disregard those tools.
- 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.
- Some tools actually have publicly available datasets linked to their papers.

Given these results, I have decided to review for my annotated bibliography the top three non-periodical time course tools, and the other three tools that have a linked GEO Datasets accession, which are:

- Next maSigPro,
- Dream,
- ImpluseDE2,
- DPGP,
- TimeMeter,
- PairGP

2 Tools review

2.1 Next maSigpro

Reference Nueda, M. J. *et al.* (2014). Next maSigPro: Updating maSigPro bioconductor package for RNA-seq time series. *Bioinformatics (Oxford, England)*, **30**(18), 2598–2602

Method **The model** maSigPro was originally developed for the analysis of microarray data. In this paper, the authors present a new adapted version suited for the analysis of RNA-seq data, by changing its model and incorporating generalized linear models. By doing so, the software can model the response variable to more than just Normal distribution, including Poisson, Binomial, Gamma, and Negative Binomial. For RNA-seq data, the recommended default option is Negative binomial.
maSigPro works in two steps: first selects non-flat gene expression profiles, then creates the best regression models for all the genes by calculating the goodness of fit R^2, which can be defined as the percentage of deviance explained by the model.
There is no normalization step, so the data must be normalized first.
Validation The software has been validated on simulated datasets – created with a Negative binomial distribution – in comparison with edgeR. These synthetic data have been modelled to reflect three scenarios: increase expression of all DEGs, half DEGs increase and half decrease, and initial upregulation followed by decrease expression. They also had multiple replicates (up to 5) and up to two time courses.
The authors also analysed a real dataset previously published describing the transcriptional response of thale cress to powdery mildew. After an initial analysis, they ran the tool on data corresponding to four time points with two series.

Code The software is available as an R package within Bioconductor (`https://bioconductor.org/packages/release/bioc/html/maSigPro.html`). It is actively maintained and updated. The current version is 1.72.0 (last updated 25 April 2023).

Data The script returned no GEO Dataset ID. In the paper, the authors list a previously published dataset that can be accessed with GEO accession GSE43163.

Summary maSigPro is routinely used in the analysis of time-dependant RNA-seq data, which promotes it to a potential candidate for the differential expression analysis step in our pipeline.
It can be easily implemented since it is in Bioconductor, and it also has a descriptive user guide to refer to for example usage.
It can visualize its results and perform clustering.
A potential negative aspect is that the data need to be normalized first.

2.2 Dream

Reference Hoffman, G. E. and Roussos, P. (2021). Dream: Powerful differential expression analysis for repeated measures designs. *Bioinformatics (Oxford, England)*, **37**(2), 192–201

Method **The model** Dream (differential expression for repeated measures) has been developed to address differential expression analysis of data with multiple measurements per condition – *e.g.* time points in our case, biological or technical replicates, etc. – by accounting for the correlation between the repeated observations.
Dream is based on an "existing" linear mixed model using the `limma` R package and its function `duplicateCorrelation`, which estimates a single genome-wide within-group variance, at the cost of reducing power and increasing the false-positive rate. The model then allows the variance to change among genes via an iterative optimization algorithm, which approximates degrees of freedom for hypothesis testing to reduce false positives.
Validation The software has been run on simulated datasets (4-50 individuals each with 2-4 biological replicates) and compared against similar workflows (`duplicateCorrelation` from the `limma`/`voom` workflow, and `macau2`) and differential expression methods that do not account for repeated measurements (`DESeq2` and `limma`/`voom`). In all cases, dream outperformed the other methods, while accurately controlling the type I error.
The authors then tested dream on different real datasets:

- an RNA-seq study on 4 regions of post-mortem brains from 26 individuals with Alzheimer's disease, correctly identifying known dysregulated patterns;
- an RNA-seq study on iPSC-derived neurons and neural progenitor cells from 11 patients with childhood onset schizophrenia and 11 controls with up to 3 lines per donor and cell type;
- a microarray study on iPSC and derived cell types from 2 individuals affected by Timothy syndrome and 4 unaffected, with up to 6 lines per donor per cell type;
- large scale RNA-seq datasets from the post-mortem human brains from the CommonMind Consortium and whole blood from Depression Genes and Networks, to assess if the expression variation across individuals is driven by genetic regulation.

Code The software is available as part of the R package `variancePartition` within Bioconductor (`http://bioconductor.org/packages/release/bioc/html/variancePartition.html`). It is actively maintained and updated. The current version is 1.30.2 (last updated 7 June 2023). All the code used to produce the results as shown in the paper is available at `https://github.com/GabrielHoffman/dream_analysis`.

Data The script returned no GEO Dataset ID. In the main text, the authors reported using real biological data. In the supplementary data, authors listed several datasets from previously published works that can be accessed with

- Synapse datasharing ID syn9907463
- Synapse datasharing ID syn3159438
- GEO accession GSE25542
- GEO accession GSE79636
- SRA accession SRP047194, also available at GEO accession GSE90749

Note: Synapse data sharing platform requires an account to access the data.

Summary Despite the very questionable acronym, dream is presented as a very solid tool, build on top of routinely used packages and applying state-of-the-art practices in RNA-seq data analysis. Its strength lies in the "correction" step which accounts for the otherwise higher false positive rate.
Unlike other methods for analysing RNA-seq data, dream models counts on a weighted linear model instead of a generalized linear mixed model, and giving better hypothesis testing as a result. Because of this, it could be very easily implemented inside pre-existing pipelines since it uses `limma`, which is the go-to standard for analysing microarray and RNA-seq data, and all these packages are part of Bioconductor and very well documented.
Another positive aspect is the presence of the code used in the paper, so reproducing the original results should be quite straightforward.
Negative aspects could be that the data needs to be normalized first, and that it is not clear to me if this model is suited for modelling time series data, since throughout the paper the authors only used measurement repetition of conditions (replicates, number of individuals, cell types, tissue type), but never time points. I assume that the variation between time points could be modelled like any other condition, but it is only mentioned and never shown.
Overall, this is a very interesting tool for differential expression analysis worth looking into.

2.3 ImpluseDE2

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

Method **The model** ImpulseDE2 is the successor of ImpulseDE, and it models gene expression trajectories over time using a pulse function. This function represents the transition between an initial state to a "peak" state, to a "steady" state.
The differential expression analysis is based on the model fits with a log-likelihood ratio test. The software assumes the number of reads are negative binomially distributed. When supplying normalized data, the user needs to check that the assumptions for the negative binomial distribution are not violated.
ImpulseDE2 can perform two types of differential expression analysis:

- case-only, where the impulse fit is compared against a constant fit, so the software looks for changes over time;
- case-control, where two time courses are provided, and the software uses a null model with a function that fits both against an alternative model with two fits for each time course.

Validation The authors tested ImpulseDE2 against ImpulseDE, `DESeq2`, and `limma` using 7 different biological datasets.

Code The software 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). ImpulseDE2 package has been removed starting from Bioconductor version 3.13.
It looks like it is not actively maintained.

Data The script returned no GEO Dataset ID. In the main text, the authors report a table with the 7 datasets they used. In the supplementary material, they listed in detail how to access them:

- GEO accession GSE84874
- GEO accession GSE75748
- GEO accession GSE59636
- GEO accession GSE59784
- GEO accession GSE78167
- GEO accession GSE57439
- full list of SRR run IDs used to download the FASTQ files (omitted in this document)

Summary The authors showed that ImpulseDE2 is suitable for the analysis of time series RNA-seq, as well as ChIP-seq and ATAC-seq. Although they claim the software outperforms the other tools, their results seem to show that it has comparable performances or sometime slightly worse, but ImpulseDE2 is implemented to perform well out-of-the-box, whether all other tools had to be used with non-standard settings.
The model seems to recapitulate expression variations over time as a single maximum or minimum per gene, which I think could be a little reductive. Another negative aspect is that it is poorly maintained, and it is no longer in Bioconductor, but the package seems to be very well documented.

2.4 DPGP

Reference 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

Method **The model** DPGP is a clustering tool for time series analysis of transcriptional data. It uses a Dirichlet process to estimate the number of clusters, and a Gaussian process for modelling time series dependency. Given this mixed model, the software then uses a Markov chain Monte Carlo method to estimate the posterior probability distribution of a certain gene to belong to a certain cluster according to the DP prior and with the likelihood according to the cluster GP distribution.
Validation The authors have tested DPGP on 620 simulated datasets, and compared the results with other clustering methods.
They also showed that they were able to recapitulate similar previously published results on a biological dataset of a bacterial response to oxygen peroxide, and they showed how DPGP can be used on a novel dataset from a study on the glucocorticoid response in a human cell line. In the latter, the software clustering correlated with transcription factor occupancy and histone modification, with a clear biological interpretation.

Code The 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, it has not been updated.
It looks like it is not actively maintained.

Data The script returned a GEO Dataset ID, which links to the GEO accession GSE104714. In the main text, there is a second dataset listed from a previously published paper that can be accessed with GEO accession GSE33980.

Summary DPGP is a clustering tool designed to overcome two problems of classical clustering methods:

1. the need to specify a number of clusters,
2. time points are assumed to be independent of each other.

I believe it could be a useful addition to our pipeline, in particular for visualizing the results of a differential expression analysis step. It could also be used to subset the list of DEGs for later stages, *i.e.* pathway analysis.
The fact that the software is not maintained and developed is the main negative aspect, but the authors provided the software and enough information so that it should be possible to easily reproduce their results and eventually include DPGP in our pipeline.

2.5 TimeMeter

Reference Jiang, P. *et al.* (2020). TimeMeter assesses temporal gene expression similarity and identifies differentially progressing genes. *Nucleic Acids Research*, **48**(9), e51

Method **The model** TimeMeter uses the dynamic time warping algorithm – originally developed for speech recognition, and available as the R package `dtw` – to assess temporal similarity among genes.
The software assumes the sequences are comparable, so all indices must be matched. Alternatively, it could be used a smaller time window so that the mismatched time points are not taken into consideration. TimeMeter then calculate 4 metrics on the matched alignments produced by `dtw`:

1. percentage of alignment for the query,
2. percentage of alignment for the reference,
3. aligned gene expression correlation, and
4. likelihood of alignment arising by chance.

Temporal patterns for each gene pair are then scored by the slopes in segmented piecewise regression, indicating the velocity of dynamic changes (with 1 meaning no changes).

Validation The authors have tested TimeMeter on simulated datasets, and on previously published biological ones:

- a comparative study between axolotl and Xenopus early embryonic development
- a comparative study between human and mouse embryonic gene expression during neural differentiation
- a comparative study between mouse digit regeneration and axolotl blastema differentiation at amputation site

Code The software is available as an R package at the website `http://www.morgridge.net/TimeMeter.html` (version 1.0.4).

Data The script returned a GEO Dataset ID, which links to the GEO accession GSE130438. In the main text, the authors mention other datasets they had previously published, and they analysed again with TimeMeter. They do not provide any accession number, but by looking up the original papers it is possible to access those datasets with

- GEO accession GSE78034
- GEO accession GSE65785
- GEO accession GSE90053
- GEO accession GSE34394
- GEO accession GSE92429

Summary It is my understanding that this software basically performs pairwise comparison of temporal trajectories. It is unclear to me what other assumption the `dtw` package has, and how valuable TimeMeter output is.
Besides this, it seems to be a somewhat obscure tool that has not been used from others but its developers.

2.6 PairGP

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

Method **The model** PairGP has been developed for modelling longitudinal time series using a non-linear, non-stationary and non-parametric method, by implementing Gaussian processes that can account for paired experimental designs.
Gene expression is considered separately, log transformed and modelled as a combination of

- a response model, chosen among all the possible partitionings by the largest marginal likelihood;
- a pairing model, which models the deviation from the response model, it is shared by all measurements of batches or replicates, and its cumulative sum is constrained to 0;
- random noise.

In the "base model", all possible partitionings are scored and combined to obtain the response function. In this implementation – which is similar to previous implementations of Gaussian processes – the "pairing effect" is not modelled.

Validation The authors have tested PairGP on simulated datasets with 9 time points, 3 or 4 conditions, 3 replicates, and a total of 1000 genes, comparing its performance between the regular model and the same model but without "pairing effect". They showed PairGP implementation is better when the variance is larger.
They also used PairGP on real longitudinal datasets:

- human T-helper cell differentiation microarray data, with 2 treatments, 9 time points, and 3 technical replicates (cell cultures)
- Mouse T-helper cell differentiation RNA-seq data, with 5 treatments, 9 time points, and 6 cell cultures, each with 3 technical replicates

Code The software is available as a Python program at the GitHub repository `https://github.com/michelevantini/PairGP` (last updated 17 October 2021). Since its publication, it has not been updated.
It looks like it is not actively maintained.

Data The script returned a GEO Dataset ID, which links to the GEO accession GSE154467. In the main text, there is a second dataset listed from a previously published paper that can be accessed with GEO accession GSE18017.

Summary This study is basically a proof-of-concept. The authors did not compare the performance of PairGP to any other software, despite listing other publications in which Gaussian process methods had been used, but only against their own implementation of a "standard" Gaussian process method.
PairGP output a partition frequency, which could be used to interpret the differential expression between all possible conditions. It is unclear to me if it can also output another statistics to assess the degree of differential expression. In the paper, the authors also did not compare their findings to any other biological interpretation, so it is not clear if their results are in line with prior knowledge about T cell differentiation.
In the repository, it is also provided a tutorial notebook, but it is very minimalistic, and there is no other documentation provided.
Overall I would not consider this tool for our pipeline, but it is worth looking into to understand possible implementations of Gaussian processes.

3 Appendices

3.1 Paper metric

The following script – `entrez.py` – produces the full table 2 as output. It assumes the presence of a file containing the user's NCBI credentials (please refer to this guide `https://support.nlm.nih.gov/knowledgebase/article/KA-05317/en-us`) to pass to E-utilities, called `entrez_credential.py` with the following format:

```
email = "USER_EMAIL"
api_key = "USER_API_KEY"
```

`entrez.py` script:

```
import pandas as pd
from Bio import Entrez
from datetime import datetime
import entrez_credentials
from time import sleep


# User credentials to pass to E-utilities
Entrez.email = entrez_credentials.email
Entrez.api_key = entrez_credentials.api_key


def my_elink(**kwargs):
    """
    Function to handle Elink function calls
    """
    handle = Entrez.elink(
        **kwargs
    )
    result = Entrez.read(handle)
    handle.close()
    return result


def get_citedby(pmid: str) -> int:
    """
    Function for returning the number of papers that cite the original input

    Argument
        pmid     Pubmed ID
    """
    param = {
        "dbfrom": "pubmed",
        "linkname": "pubmed_pubmed_citedin",
        "id": pmid,
    }

    result = my_elink(**param)
    out_citedby = len(result[0]["LinkSetDb"][0]["Link"] if result[0]["LinkSetDb"] else list())
```

```
    sleep(0.2)
    return out_citedby


def get_pub_date(pmid: str) -> str:
    """
    Function for returning the date an article has been published ("entrez" PubDate)

    Argument
        pmid     Pubmed ID
    """
    param = {
        "db": "pubmed",
        "id": pmid,
    }

    handle = Entrez.efetch(**param)
    result = Entrez.read(handle)
    handle.close()

    year = result["PubmedArticle"][0]["PubmedData"]["History"][0]["Year"]
    month = result["PubmedArticle"][0]["PubmedData"]["History"][0]["Month"]
    day = result["PubmedArticle"][0]["PubmedData"]["History"][0]["Day"]
    date = "-".join([year, month, day])

    sleep(0.2)
    return date


def get_linkDb_id(pmid: str, db: str, **kwargs) -> str | None:
    """
    Function for returning the ID in a different NCBI Db linked to the original paper (given a PMID)

    Arguments:
        pmid        Pubmed ID
        db          NCBI database

    Return a string with the new ID or None object if no IDs are found
    """
    param = {
        "dbfrom": "pubmed",
        "db": db,
        "id": pmid,
    }

    result = my_elink(**param, **kwargs)
    out_id = result[0]["LinkSetDb"][0]["Link"][0]["Id"] if result[0]["LinkSetDb"] else None

    sleep(0.2)
    return out_id


def get_pmid(doi: str) -> str:
    """
    Function for returning the PMID of a paper given its DOI

    Argument
        doi     DOI string
    """
    handle = Entrez.esearch(
        db="pubmed",
        term=doi,
        field="doi",
    )
    result = Entrez.read(handle)
    handle.close()

    sleep(0.2)
    return result["IdList"][0]


if __name__ == "__main__":
```

```
    table = pd.read_csv("report/src/oh2021.csv", sep=";")

    table["PMID"] = table["DOI"].apply(get_pmid)
    table["pub_date"] = table["PMID"].apply(get_pub_date)
    table["pub_date"] = pd.to_datetime(table["pub_date"], format="%Y-%m-%d")
    table["cited_by"] = table["PMID"].apply(get_citedby)
    table["avg_year_cit"] = table["cited_by"] / table["pub_date"].apply(lambda date:
    ↪  (datetime.today() - date).days / 365.25)
    table["geo_dataset_id"] = table["PMID"].apply(get_linkDb_id, db="gds")
    table["bioproject_id"] = table["PMID"].apply(get_linkDb_id, db="bioproject")

    table.to_csv("report/src/oh2021_metric.csv", index=False)
```

3.2 LaTeX table output

The following script – `tables.py` – produces the two LaTeX formatted tables included in this document, starting from the output of the previous script, `entrez.py` (see 3.1).

```
import pandas as pd


# Table adapted from Oh2021 paper
table = pd.read_csv("report/src/oh2021.csv", sep=";")

table.style \
.hide(axis=0) \
.format_index("\\textbf{{{}}}", escape="latex", axis=1) \
.format(escape="latex") \
.to_latex(buf="report/src/oh2021.tex")

# Table obtained with entrez.py script
types = {
    'PMID': 'object',
    'geo_dataset_id': 'object',
    'bioproject_id': 'object'
}
table = pd.read_csv("report/src/oh2021_metric.csv", sep=",", dtype=types)

# Filter table for readability: subset of columns and relative new labels
subset_cols = [
    "Name", "Time course",
    "cited_by", "avg_year_cit",
    "geo_dataset_id", "bioproject_id"
]

relabels = [
    "Name", "Time course",
    "Cited by", "Average yearly citations",
    "GEO Dataset ID", "Bioproject ID"
]

col_format_latex = [
    "l", "l",
    "c", ">{\centering}m{2.3cm}",
    "c", "c"
]

new_labels = dict()
for i, j in zip(subset_cols, relabels):
    new_labels[i] = j

col_latex = "".join(col_format_latex)

# Sort table before export it in LaTeX
table[subset_cols].sort_values("avg_year_cit", ascending=False).rename(new_labels, axis=1).style \
.hide(axis=0) \
.relabel_index(relabels, axis=1) \
.format_index("\\textbf{{{}}}", escape="latex", axis=1) \
.format(escape="latex", na_rep="", precision=2) \
.to_latex(buf="report/src/oh2021_metric.tex", column_format=col_latex)
```

References

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

Hoffman, G. E. and Roussos, P. (2021). Dream: Powerful differential expression analysis for repeated measures designs. *Bioinformatics (Oxford, England)*, **37**(2), 192–201.

Jiang, P. *et al.* (2020). TimeMeter assesses temporal gene expression similarity and identifies differentially progressing genes. *Nucleic Acids Research*, **48**(9), e51.

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.

Nueda, M. J. *et al.* (2014). Next maSigPro: Updating maSigPro bioconductor package for RNA-seq time series. *Bioinformatics (Oxford, England)*, **30**(18), 2598–2602.

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

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.

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