

Building a data pipeline to extract signatures of evolutionary dynamics from tumours

Razak Nart, Mingfeng Liu, Lukas Rubikas,   
Yuhong Lin, Jesus Vera

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfillment of the requirements of the Degree of Master of Science at the University of Glasgow

December, 2018

**Abstract**

Analysing tumour samples for estimation and clustering of cancer cell fractions is of vital importance when studying intratumour heterogeneity. We introduce a pipeline containing three of the best tools to extract cancer cell fractions (CCF) and clustering from tumour samples and two other tools that will do topic extraction using the CCF values. Separating these tools in two phases, the results from the first phase will be compared on accuracy and speed. As for the results of the second phase will be compared for similarity. The user will be able to run the pipeline through a Docker environment.

Contents

Chapter 1 Introduction 2

Chapter 2 Background Survey 3

2.1 PyClone 3

2.2 Ccube 4

2.3 PhyloWGS 4

2.4 Non-negative Matrix Factorization (NMF) 4

2.4 Latent Dirichlet Allocation (LDA) 5

Chapter 3 Requirements 6

3.1 PyClone 6

3.2 Ccube 7

3.3 PhyloWGS 7

3.4 NMF and LDA 7

3.5 Pipeline 8

Chapter 4 Design and Implementation 9

4.1 PyClone 9

4.2 Ccube 10

4.3 PhyloWGS 12

4.4 Non-negative Matrix Factorization 12

4.5 Latent Dirichlet Allocation 16

4.6 Pipeline 19

Chapter 5 Evaluation 20

5.1 First Phase 20

5.2 Second Phase 20

Chapter 6 Conclusion 22

Chapter 7 Contributions 23

7.1 Final product 23

7.2 Report 23

References 25

Appendix A User Manual 1

# Introduction

With the advance of technology and the continuous increment of cancer cases, it has become of vital importance to keep doing research on this matter and to create computer tools that can help scientist to find cures or to improve the accuracy of analysis for this disease and to get an early prognosis on patients.

Machine Learning plays an important part on this kind of researches because, with the use of different algorithms, computers can be taught to do repetitive processes where models can be improved with the data obtained on each model iteration. Thus, making a model better for predicting and analysing results and giving great importance to the creation of new tools that implement best algorithms for machine learning.

The main objective of this project is to build a pipeline that will use 500 tumour samples from different patients and studying their intratumour heterogeneity to estimate and extract the cancer cell fraction (CCF) values using clustering on each of the samples during the first phase. Then, these results will be analysed for topic extraction during the second phase.

Now, this project will be divided into two phases. Each phase will require different data to be able to complete their task. During the first phase, there are three important tools that are used to analyse the tumour samples: PyClone, Ccube and PhyloWGS. These tools work in different ways, yet they are employed to do the same job. For this project, these tools will be analysed, implemented and integrated into a pipeline that will make possible to run them together. Furthermore, it will be shown differences between the tools which will help in the future to decide which tool works better, although on this project it will be limited to measure only speed and accuracy.

After obtaining the results from the first phase, another two tools will be run using those results: Non-negative Matrix Factorization and Latent Dirichlet Allocation. These tools use two different models for topic extraction, but these models need the results from the first phase to run. Once the models are given with the necessary data, they will analyse the CCF values and will obtain a certain number of topics as a result. The values contained in the topics will represent the probability (LDA) or weights (NMF) of a document to be matched with a certain topic. These two tools will also be integrated into the pipeline that will run the five different tools one after the other.

Finally, once the pipeline is ready with the five different tools, a Docker environment will be built and configured to run it. This environment will contain all the necessary tools already installed and ready to have the pipeline code run.

# Background Survey

In this chapter, it will be explained five different tools used for the analysis of cancer. They are divided into two phases. The first phase includes three tools that will analyse 500 different cancer samples to extract important values from them and with that help the next phase to continue with the analysis. The second phase will include two different tools for topics extraction from the results during the first phase. These topics will be the relations between the different cancer cell fractions (CCF) across all tumour samples.

The first phase will use the tools: PyClone, Ccube and PhyloWGS; and the second phase will consist of the tools for using the models of Non-negative Matrix Factorization and Latent Dirichlet Allocation.

## PyClone

Although the development of measuring allele prevalence by using deep sequencing has grown successfully, there wasn’t a statistical method for clustering deep digital sequencing of mutations into biological groupings.

Aiming at classifying sets of deeply sequenced somatic mutations into a presumptive clonal structure, a hierarchical Bayes statistical model for the assumption of clonal population clusters was developed by Andrew Roth and his workmates [1]. Apart from the classification, the model calculates clusters’ cellular prevalence (which refers to cancer cell fraction (CCF) in this dissertation) as well as makes an explanation of allelic imbalances.

The paper highlights that, during clonal phylogeny, the clusters of mutations which locate at the same point are sharing cellular prevalence. Taking advantage of this property, the model considers it as markers of clonal populations. However, the allelic prevalence of a mutation contributes to several features, which means it is not relevant directly to cellular prevalence. It will be aggravated especially if a single sample estimation is taken.

Based on the property and Bayes theory, a Bayes statistical software tool, PyClone, was developed to overcome the above difficulty. Pyclone consists of four innovation modelling processes, including beta-binomial emission densities, estimation of flexible prior probability for possible mutational genotypes, Bayesian nonparametric clustering and joint analysis of multiple samples.

The software tool was tested in idealized datasets which extracted from four 1000 Genomes project samples and mutational profiles of multiple samples provided from a high-grade serous ovarian cancer (HGSOC) [2][3]. By comparing two results, Pyclone obtains similar conclusions between two datasets, which proves its robustness. The experiments illustrate that PyClone provides a reliable statistic inference method to support researches in the progression of cancer. Additionally, according to cluster accuracy, the paper reveals that PyClone with beta-binomial emission densities with parental and total copy-number priors surpass all other methods in the process of emission densities [1].

## Ccube

For producing the cancer cell fraction (CCF) from somatic point mutation calls and the clusters of every mutation, a good way is to use Ccube to analyse the data and get the output files for the second phase of this project.

Ccube is a tool written with R code for clustering the cancer cell fraction. Moreover, it is a probabilistic framework that includes a variational inference method for model fitting, which allows the user to process samples with a large number of variants while quantifying the uncertainty in a Bayesian fashion [2]. From the whole genome sequencing data, the cancer cell fraction and sub-clonal composition of somatic cell point mutations could be inferred. It requires sequencing read analysis of single nucleotide variants (SNV), correcting their copy number changes and purity, and generating CCF estimates for all mutations in the sample.

Furthermore, the Ccube choose the suitable number of clusters by variational inference which can get the best evidence lower bound and assign the mutations to the clusters according to the probability. Also, inside the Ccube pipeline there is an optional parameter named “runQC” that when is set to TRUE, the Ccube can remove small clusters and re-assign variants to make the result of clustering more accurate.

Thus, using Ccube it is possible to find the CCF and VAF relationship. Hence, the map between them is as follows:

*Formula 1*: *Linear mapping between the probability of observing a variant read at a mutated locus, f, and the CCF of the mutation . Where is a uniform sequencing error, and m is the number of mutated chromosomal copy [4].*

## PhyloWGS

## 2.4 Non-negative Matrix Factorization (NMF)

Non-negative Matrix Factorization (NMF) introduced by Lee and Seung [5][6], is a model that is used to achieve a dimension reduction on a large complex data matrix to obtain valuable features. It works much like Principal Component Analysis (PCA) but in NMF, each feature in the data matrix, must be greater or equal to zero (non-negative) [7].

Since the introduction, other researchers had successfully implemented NMF in different areas, such as document clustering [8]; information retrieval [9]; facial expression recognition [10]; gene expression analysis [11][12]. The aim is to factorise a non-negative data matrix *A* with dimension *m x n* to produce approximation matrices *WH* with dimensions *m x k* and *k x n* respectively, that is,

,

where the k serves as the number of component factors in the model and normally selected so that *mn > k(m+n)* [5][11].

Regarding this project, Cancer Cell Fraction (CCF) expression values from a set of tumour samples are presented in a matrix *A* with dimension *m x n*, where the rows *m* relates to the band of the CCF, the columns *n* relates to the tumour samples and entries are the total values of the CCF that falls within the bands. After applying NMF on the CCF expression value matrix *A* to *WH*, where *W* has dimension *m x k*, such that *k* columns determine the number of topics in the expression and where H has dimension *k x n,* such that *n* columns reveal the weight of the expression of the tumour sample. Furthermore, the *W* can be described as a feature matrix and *H* as the coefficient matrix.

To factorise the data matrix *A* to get the product of *WH*, an objective function must be defined to measure the approximation and the reconstruction error. This function can be designed by adjusting the distance between *A* and the multiplication result *WH* [11]. The commonly used method for measuring the distance is the squared *Frobenius norm* a branch of the *Euclidean norm* [8].

*Formula 2*: The Frobenius norm for measuring the distance separating *A* and its product *WH*, [5][18].

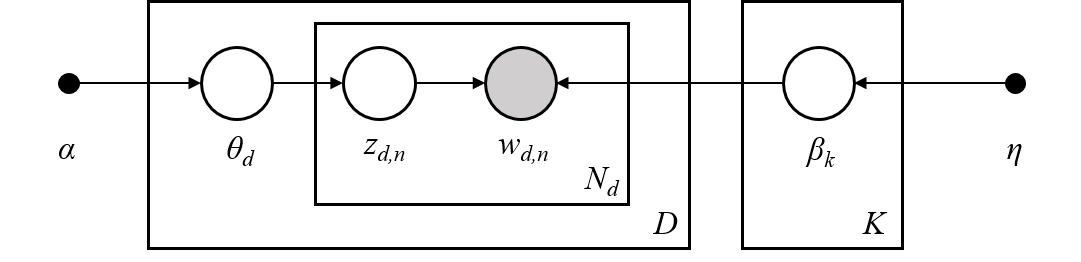
A different method for measuring the distance is *Kullback-Liebler* [5][11]. The commonly used process for adjusting the *W* and *H* in *Formula 2* to reduce the objection function is to use the multiplicative update rules to iterate between the *W* and *H* until convergence [5]. The rules are as follow,

,

*Formula 3*: The Multiplicative Update Rules for *W* and *H* [5].

## Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model that can abstract topics from collections of discrete data. It can also be seen as a model with three levels, and every item of a collection is matched with a topic (or topics) and their probabilities, therefore showing a clear representation of text corpora [13].



*Figure 1: Graphical model representation of LDA, from scikit-learn “Decomposing signals in components (matrix factorization problems)” [14]. The corpus is represented by D, the document is a sequence of N words, K is the topics in the corpus and the boxes are repeated sampling.*

Explained differently, LDA assumes that each word is generated by an assortment of topics, then it models the relationships between documents and topics and the relationships among such topics and words.

LDA also assumes, in this project, for each document *w*in a corpus*D [13][15]:*

1. Choose a multinomial *ξk* (*k*∈{1, …, *K*}) for each topic from a Dirichlet distribution (*β*);
2. Choose a multinomial *θs* (*k*∈{1, …, *S*}) for each CCF band from a Dirichlet distribution (*α*);
   1. Choose a topic *z* from a multinomial (*θs*).
   2. Choose a word *wn* (*n*∈{1, …, *N*}*,* where *N* is the number of words in the current document) from a multinomial (ξz).

Therefore, the output of LDA model are two matrices, a document-topic and a topic-words matrix, which can be represented with dimensions (N, K) and (K, M) respectively, where N is the number of documents, K is for the number of topics and M is the vocabulary size. These matrices, represent the probability distributions of documents-topics and words-topics.

# Requirements

Since this project was a mixture of different tools, the requirements varying depending on the tool, hence the requirements would be better explained if separated. Furthermore, the requirements for the pipeline with the five tools will be explained and it will include how it should be implemented and ran.

## PyClone

For the PyClone tool, the only operative system employed was Unix (like Linux or MacOs) and the version of Python must be 2.7.

Additionally, it should be mentioned that Pyclone is the main library used in this project and it’s also an open-source code [4]. According to its documents, the main dependencies are PyDp, PyYAML. Those two libraries are used for analysing. Besides, Matplotlib, Numpy, Pandas, Scipy and Seaborn (higher or equal than 0.6.0) are required for plotting result and clustering. PyClone and all the required libraries can be installed via pip install (note: refer to README.txt for installation instructions).

Additionally, R scripts are used to pre-process input files and generate output files. The required libraries for R scripts are “dplyr” and “mcclust”.

The final requirement is the data. According to the documents of PyClone, the input is a set of deeply sequenced mutations from one or more samples extracted from a single patient as well as allele-specific copy number at each mutation trajectory in each sample [1]. In our case, there are 500 samples from different patients.

In summary:

* OS: Unix System (Linux and MacOs)
* Coding language: Python (2.7) and R (3.4.0 or above)
* Main library used: PyClone (latest version)
* Other Python libraries:
  + PyDp (0.2.3 or above)
  + PyYAML (3.10 or above)
  + Matplotlib (1.2.0 or above)
  + Numpy (1.6.2 or above)
  + Pandas (0.11 or above)
  + Scipy (0.11 or above)
  + Seaborn
* Other R libraries: dplyr, mcclust
* 500 samples data (previously provided by the supervisor)

## Ccube

Similarly, in the case of the tool Ccube, the required operating system is Linux. It works using the programming language R. Additionally, the core package is “Ccube” with version 0.0.0.9000; which includes all functions of Ccube to analyze the mutations and get the CCF values. Ccube also needs the package “dplyr” to add new variables that work as functions of existing variables and the package “doParallel” to run it using multi-cores.

Another important requirement is the data used, which should be converted to a suitable format. The code of this process is based on the Python. It needs the package “os” to read the path from the command shell, and the package “numpy” to stack the arrays.

In summary:

* OS: Unix System (Linux and MacOs)
* Coding language: Python (2.7), R (3.4.0 or above)
* Main library used: Ccube (latest)
* Other Python libraries:
  + Numpy (1.6.2 or above)
  + Os (latest for Python 2.7)
* Other R libraries: DoParallel, Dplyr

## PhyloWGS

## NMF and LDA

For the NMF and LDA, and because of the nature of this tools, the requirements can be explained together.

To begin with, it should be mentioned that the operating system (OS) doesn’t have to be any in specific, if the Python language can be used for coding. Nonetheless, when working with both tools Windows was selected as the main OS.

For both models, the main library needed is Scikit-learn [16], which is a tool for doing Machine Learning using the Python language. This tool will require the installation of Numpy and SciPy, which are another two libraries used for development in Python.

The final requirement for these two models to work would be the results from the previous phase, but more specifically, the Subclonal Structure files. From these files, a large dataset will be created containing the 500 tumour samples and the cancer cell fractions (CFF) values. This dataset will be 10 rows and 500 columns long, representing the 10 CCF bands {0.1, …, 1.0} and the 500 samples.

There were no problems during the installation of any of these libraries or using these tools.

In summary:

* OS: Any
* Coding language: Python (2.7)
* Main library used: Scikit-learn (0.19.2)
* Other libraries:
  + Matplotlib (1.2.0 or above)
  + Seaborn (0.9.0)
  + Numpy (1.6.2 or above)
  + Pandas (0.11 or above)
  + Scipy (0.11 or above)
* CCF dataset of 500 tumour samples with dimension (10, 500).

## Pipeline

# Design and Implementation

Since this project is only for analysis of cancer cells, it uses five different tools and each tool has its own implementation. There are no designs related whatsoever. Nonetheless, the construction of the pipeline does have a design and a different implementation.

This report contains the description of the different tools used for the analysis of cancer tumours samples, for which every tool required a different implementation. In the description of every implementation, it will be mention what the tool needed for running, the parameters that the tool would use and the output at the end.

Further, the pipeline needed to be designed and build in a way that every tool from phase one was running and producing a result. Then, in phase two and with the produced result from phase one, the other two tools could run the analysis of the data to extract the information needed, which in this case will be the relations between topics and documents.

## PyClone

The main framework for PyClone contains pre-processing, analysing and post-processing. In pre-processing, 500 simulated tumour samples are inputted as source data. They are constructed as follows:

*Figure 2: How the input data for PyClone is built.*

However, this data cannot be analysed directly. The required fields for PyClone are[4]:

* Unique ID (mutation\_id)
* Number of reads covering the mutation which contain the reference allele (ref\_count)
* Number of reads covering the mutation which contain the variant allele (var\_count)
* Copy number of the cells in the normal population (normal\_cn)
* Minor copy number of the cancer cell (minor\_cn)
* Major copy number of the cancer cell (major\_cn)

Thus, a translator program is needed to extract useful information from given files and generate the target file containing the required fields. Additionally, a configuration file is also needed for analysing. In such file, the working directory, density method, number of iterations and sample information are needed to be specified.

The next step is analysis, it can be done by calling the function “run\_analysis” in PyClone. This stage will generate a folder named “trace” to store the clustering process.

In the final stage, all the result produced by PyClone are post-processed by R script. The output files from this process are the sub-clonal structure, which contains the cancer cell fraction (CCF) for each putative cluster, mutation assignment for each mutation and multiplicity.

PyClone adheres to the purpose of this project by exporting the interfaces for the final code integration, including:

* Prefix name for the inputted samples (prefix)
* The burnIn number N will drop out first N row of output data for analyzing (burnIn)
* Number of iterations (num\_iter)
* Purity of input sample (purity)

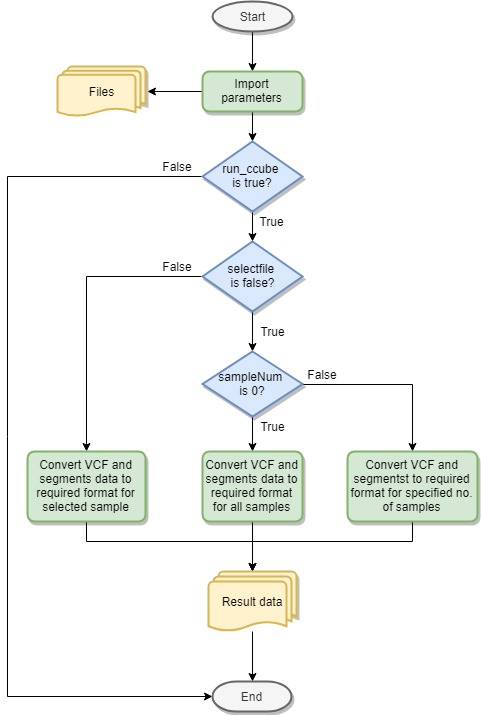
## Ccube

Ccube can be divided into three parts. The first part consists on passing the parameters to the main functions, which will be used during the process of formatting the input and the analysis, this includes seed values, number of repetitions, clusters number, the maximum number of iterations, number of cores to run, the path and the sample to analyse. However, Ccube will only be run if the parameter “run\_ccube” is set to TRUE. Also, the number of samples which will be converted and analysed is based on the parameter “selectedfile” and “sampleNum”.

The second part is to do a pre-process where the chromosome, position, vaf\_number, ref\_number, the copy number of major and minor must be obtained from the VCF, segment and pp\_table files of the samples selected. Then, this data will be stored into a TSV file on the target folders. Moreover, this data will be formatted in a suitable way for Ccube.

The third part is the analysis of the mutations and outputting the target files, where the accuracy and speed of the Ccube analysis will depend on the parameters passed. Consequently, according to the default value of the parameters, the average working time for the first ten samples is 54.33 seconds.

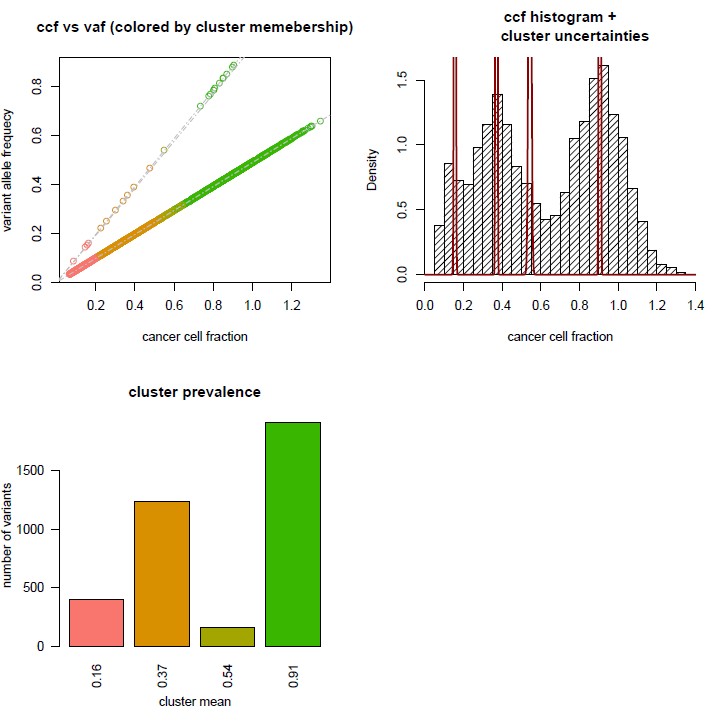
The following flowchart shows the different processes inside Ccube when it's running in the pipeline from start to end, and the user decisions when pre-processing the data:



*Figure 3*: *Ccube flowchart inside the pipeline*

The output files include three text files which are saved on different folders. Moreover, the subclone structure file has the number of simple semantic mutations (SSMS) and the CCF of all clusters. The mutation assignments file has the assignment of mutations in the different clusters. The multiplicity file shows that the chromosome, position, tumour copy numbers and multiplicity.

Finally, when testing the pipeline, the command shell will show the name of samples selected and the working time of analysing the sample.



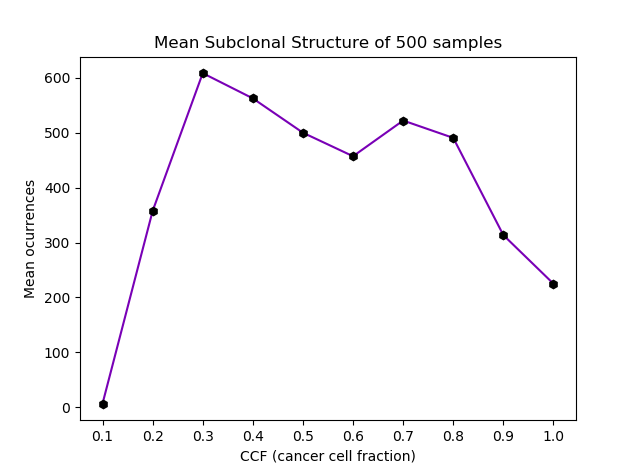
*Figure 3: Summary of the Ccube results, the relationship between CCF and VAF, and the situation of clusters*

## PhyloWGS

## Non-negative Matrix Factorization

To implement the NMF model, a data matrix was created using the 500 tumour sample files from the Subclonal Structure after phase one. Each sample file has a corresponding CCF values. Thus, all the CCF values from each file were counted and arranged in bands that start from 0.1 to 1 with step 0.1. The shape of the data matrix is10 x 500 (row x column).

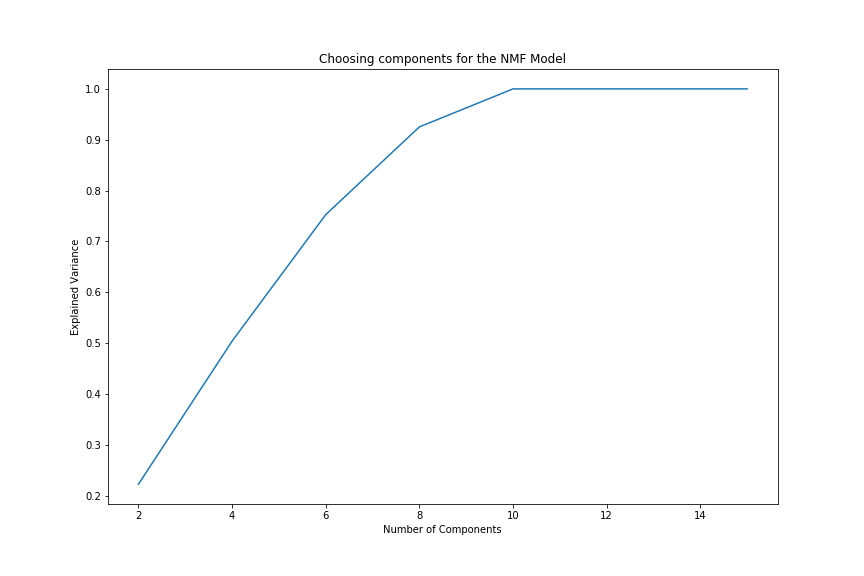
The figure below shows the average (mean) count of the total CCF value from each of the tumour samples.



*Figure 5: The CCF against the mean number of occurrences from the subclonal structure results during the first phase.*

Regarding NMF, the data matrix can be defined as *A* with dimension *m x n*, where the row *m* defines the 10 different bands *{0.1, 0.2, ...., 1}* and where column *n* defines the value of tumour mutation from the 500 sample.

Also, for this project, data pre-processing such as using Count Vectorise to return the count of the data is not needed as the data matrix is already in numbers. But to get a better factorisation of the data matrix *A* into a feature matrix *W* and coefficient matrix *H*, we had to perform a few analyses such as tuning the parameters to find the less reconstruction error and estimating the number of components (*k*) for the model to determine the optimal parameters for the NMF model. The first phase of the analysis was to estimate the number of components to use for the NMF model. This was achieved by measuring the explained variance from the data matrix *A* to reveal how many variances (information) can be associated with each of the components. We measure this more directly by creating a method to utilise a metric library from Scikit-learn to get the explain variance score from the NMF model and the data matrix *A*.



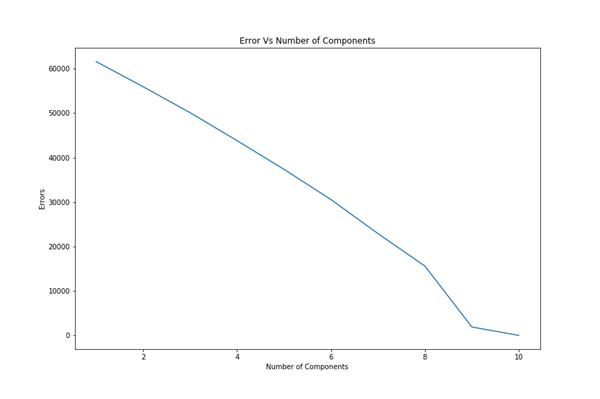
*Figure 6, estimating the number of components (k) for NMF model through explain variance.*

The judging from the figure above, it evidently shows that 6 to 9 components are needed to approximately explain 92% of the variance. However, 99.95% to be specific of the variance can be explained using components from 10. The next phase of the analysis was to tune the model to determine the parameters with least reconstruction error. To achieve this, we created a method that takes in the data matrix *A* and some NMF model parameters and returned the reconstruction error. Considering this project the NMF parameters are, n\_components, this states the number of components to use; init, this defines the initialisation method utilised for the model; max\_iter, this determines the maximum iterations before timeout; random\_state, this is stated to check reproducibility; l1\_ratio, this defines the L1 and L2 penalties to regularise the model; alpha, this is used to control the strength of the regularization [19]. Subsequently, a form a grid search was performed to find the optimal parameters for the model. The table below shows the three smallest and largest reconstruction errors for different parameter combinations.

*Table 1, shows three combinations with the smallest and largest error*.

|  |  |
| --- | --- |
| Three Smallest Error | |
| Parameters | Reconstruction Error |
| (10, ‘nndsvd’, 50, 0, 0, 0) | 1.024160 |
| (10, ‘nndsvd’, 50, 1, 0.5, 0.75) | 1.235355 |
| (10, ‘nndsvd’, 50, 1, 1, 0.75) | 1.653756 |
|  | |
| Three Largest Error | |
| (4, ‘nndsvda’, 2, 1, 1, 0) | 61975.362915 |
| (4, ‘nndsvda’, 2, 0, 1, 0.25) | 61967.598853 |
| (2, ‘nndsvda’, 2, 1, 1, 0) | 61965.296342 |

Prior searching for the optimal parameters for the model, it was explicit that most of the variance can be explained using 10 components. As a result, the maximum number of components used for tuning the model was 10. Furthermore, when tuning the model, it clearly shows that the higher the component the lower the reconstruction error of the model will be, (*Figure 7).*

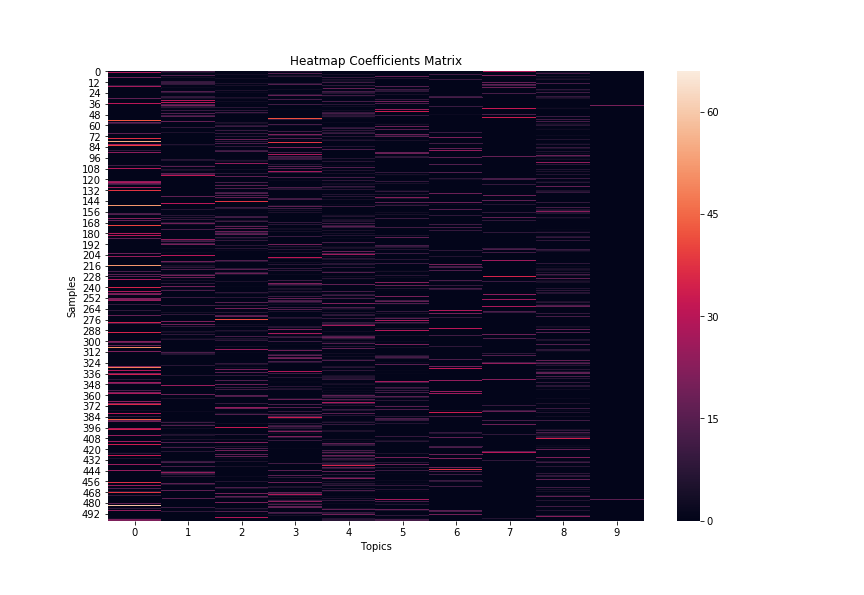


*Figure 7, the number*

*of components against the reconstruction error*.

The NMF model was applied to the data matrix A utilising the optimal parameters identified during the analysis phase. The number of components was set to 10; init = ‘nndsvd’, which means the initialization method utilised for the model is "Nonnegative Double Singular Value Decomposition"; max\_iter = 50; random\_state = 0; l1\_ratio = 0, this means the L2 (Frobenius Norm) penalty for regularisation; alpha = 0. After factorising the data matrix *A* with dimension 10 x 500 (*m x n*), the model produced approximation matrices *WH*, where *W* is the feature (topic) matrix with dimension 10 x 10 (*m x k*) and *H* the coefficient matrix with dimension 10 x 500 (*k x n*). For the feature matrix, the rows are the bands of the CCF and the columns are the number of components (topics) used for the model. While for the coefficient matrix, the rows are the number of components and columns are the tumour samples. Furthermore, the feature matrix was normalised using a library part of the Scikit-learn preprocessing libraries to get a unit value.

TheFigure below is plot of the coefficient matrix of the NMF model.



*Figure 8, plot for the coefficient matrix*

## Latent Dirichlet Allocation

For this case in specific, there’s a given dataset containing 500 tumour samples and their CCF values. As explained previously, LDA is utilized to model the relationships between these samples and extract the document-topic distribution matrix.

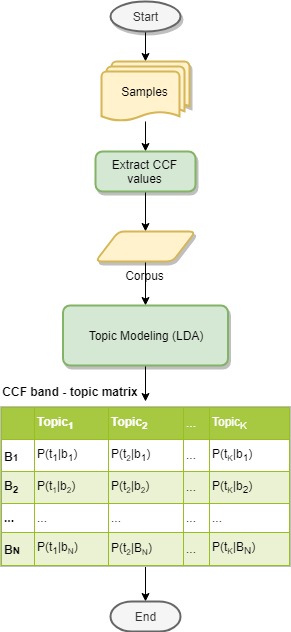
Based on a generative process describe in a previous chapter, the probability of a given dataset D = {D1, …, Ds} is formalized as

*Formula 4*: *Probability of a given dataset D*, *from Topic modelling for cluster analysis of large biological and medical datasets [7].*

The LDA is implemented in this project taking as a document the number of cell mutations found with the same CFF (cancer cell fraction) value from all 500 samples. Thus, creating 10 documents (or rows) with different bands {0.1, 0.2, …, 1} and the number of mutations from each sample working as our text corpora. The final corpus of the cancer samples dataset contained 10 documents and each document contained at most 500 words.

Moreover, since the words from the corpus are only numbers, there was no pre-processing needed for this. Normally, the preprocessing would include lemmatizing the words, removing all stop words, removing numbers, etc. Thus, helping the model to easily do the topic extraction.

Further, the corpus for this project was created reading the Subclonal Structure files resulted from phase one and extracting the CFF values from every file (represented as “proportion”), then these CCF values were inserted into an array and sent to the LDA tool for topic extraction.



*Figure 10: The workflow of the topic modelling using Latent Dirichlet Allocation.*

In order to get the best results from the LDA tool utilized, the hyperparams used were decided after doing an analysis with a different tool. This tool is *GridSearchCV* [17], which it’s also from Scikit-learn, and it helps to look for the best hyperparams for certain model (in this case, LDA model) using cross-validation. This tool takes the log-likelihood (score) and the perplexity obtained from the results to analyse the best set of hyperparams.

In this project what we tried to achieve was high likelihood and low perplexity. In the case of LDA, a lower perplexity score indicates better performance. Hence, with a test set of M documents, the perplexity would be:

.

*Formula 5: Perplexity for a test set of M documents [5].*

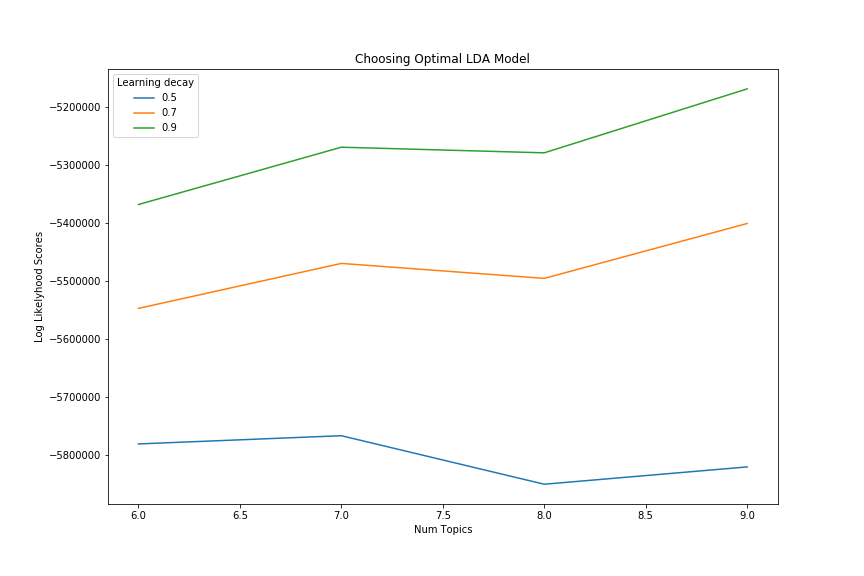
As for the log likelihood, it can also be expressed with:

.

*Formula 6*: *Likelihood of a document [5].*

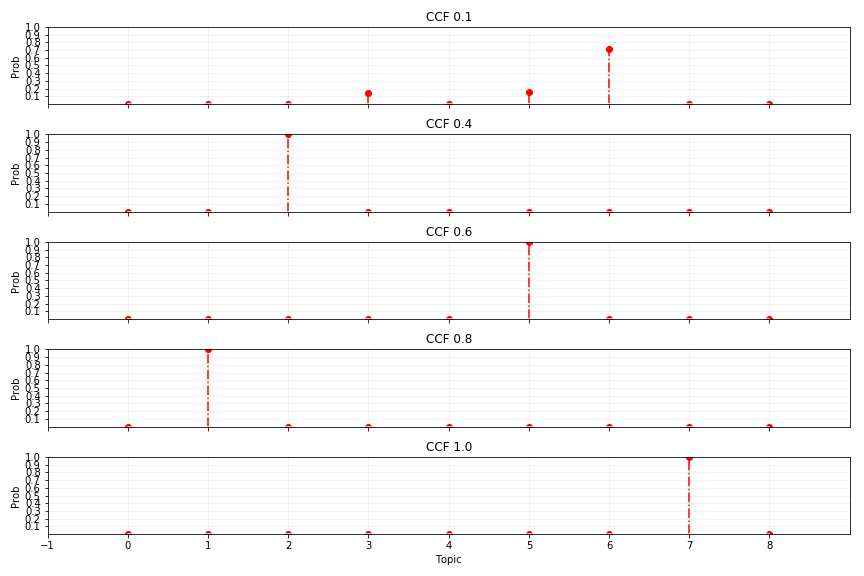
Nevertheless, when searching for the best perplexity and log likelihood, it was better to rely on a tool that could help us to achieve these values without going deeper into working with many calculations. Thus, making *GridSearchCV* a powerful tool for us.

In the following graph can be seen that picking an LDA model with nine topics, a learning decay of 0.9, a number of six maximum iterations and with a random state of 2018 is better for this dataset.



*Figure 11*: *Choosing the Optimal LDA model using log-likelihood score*.

The following graph shows the results of using the optimal set of hyperparams in five of the CCF bands:



*Figure 12: Topics distributions for five CCF (cancer cell fraction) bans.*

## Pipeline

# Evaluation

During the first phase of this project, three tools were implemented to carry on the analysis of 500 cancer samples. These cancer samples were a mix of different files: VFC files, for mutation information; segments files, with a consensus style copy number profiles; and the BB files, containing the BAF values (barrier-to-autointegration factor).

These three different tools had different data to be input, which was previously provided by the supervisor. Since each tool required a different dataset with different data, for each there was a pre-process to get the data and create the dataset (or datasets) needed.

Further, after setting the necessary params on each tool, everyone was ran obtaining the results required for the second phase. The necessary files for the second phase were the Subclonal Structures containing the clusters created and the CCF values needed for the topic extraction. These files were examined in order to extract the CCF values and then create a dataset to be used with each of the models for topic extraction. The models selected for doing topic extraction from the 500 cancer samples were the Non-negative Matrix Factorization and Latent Dirichlet Allocation.

## First Phase

## Second Phase

During the second phase, two models were used for topic extraction: Non-negative Matrix Factorization and Latent Dirichlet Allocation. In order to carry out an evaluation of these models, it will be shown a comparison to see how similar or different they were when doing the topic extraction.

Farther, when working with both models, the same dataset was used to ensure that the results from both can be compared and the differences accounted. Such dataset was created using the CCF values {0.1, …, 1.0} from the 500 tumour samples giving, as a result, a dataset with dimension (10, 500).

To make the comparisons between both models, the most important value that must be equally fed to the models is the number of topics to extract. Hence, after meticulously examining the best hyperparameters for both models, it was decided that the number of topics to be extracted would be ten. These results might look slightly different from the previous ones got for LDA, since the model was executed with nine topics to extract.

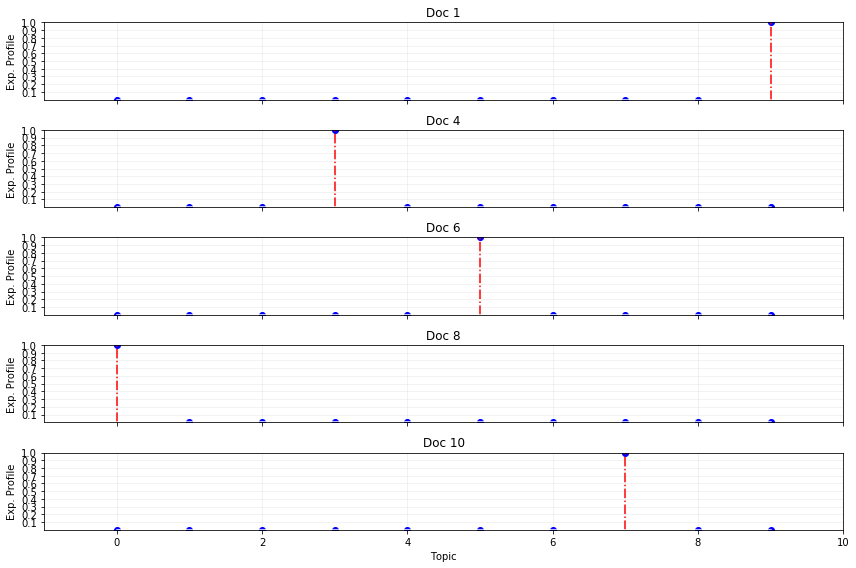
Nonetheless, the analysis was carried out selecting ten topics in both models. The results are as shown below.

*Table 2: Document-topic table from NMF*.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CCF band | Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 | Topic 6 | Topic 7 | Topic 8 | Topic 9 | Topic 10 |
| 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0.3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0.4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0.7 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

*Table 3: Document-topic table from LDA.*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CCF band | Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 | Topic 6 | Topic 7 | Topic 8 | Topic 9 | Topic 10 |
| 0.1 | 0 | 0 | 0 | 0.14 | 0 | 0.15 | 0.71 | 0 | 0 | 0 |
| 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0.3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0.4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0.6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0.7 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.8 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |



*Figure 8, five Documents for the feature (Topic) matrix*

Furthermore, from the previous tables, it can be seen that the results are very different. Although, it’s very noticeable that a big similarity is that in both models there is mostly one topic for one band (document), meaning that the topic is dominant over the CCF band. Also, the only topic that shows similitude is *Topic 6* when it’s matching with *CCF band 0.6* in both models. However, besides that match, there are no more equal combinations.

# Conclusion

From the evaluation and comparison of the topic extraction models we can conclude that when working with models for topics extraction and numbers instead of words, there are not clear results if we just stop the analysis after the topic extraction. There should be, maybe, more analysis to obtain more in-depth data from the topic extraction. In future work, it could be possible to obtain if the topic shows any closeness with the cancer types or if there could be more pre-processing of the values to obtain more topics and different probability distributions and/or weights.

# Contributions

In this chapter is presented a description of everyone’s contribution in both the final product and submitted report.

## Final product

|  |  |
| --- | --- |
| Tool/item/doc | Student(s) |
| PyClone | Mingfeng Liu |
| Ccube | Yuhong Lin |
| PhyloWGS | Lukas Rubikas |
| NMF | Razak Nart |
| LDA | Jesus Vera |
| Pipeline | Mingfeng Liu, Lukas Rubikas |
| Code review | Mingfeng Liu, Lukas Rubikas |
| Docker | Yuhong Lin |
| User Manual/README.txt | Lukas Rubikas |

## Report

|  |  |
| --- | --- |
| Chapter/Section | Student(s) |
| **Chapter 1 – Introduction** | Jesus Vera |
| **Chapter 2 – Background Survey** | Jesus Vera |
| Section 2.1 – PyClone | Mingfeng Liu |
| Section 2.2 – Ccube | Yuhong Lin |
| Section 2.3 – PyhloWGS | Lukas Rubikas |
| Section 2.4 – NMF | Razak Nart |
| Section 2.5 – LDA | Jesus Vera |
| Chapter 3 – Requirements | Jesus Vera |
| Section 3.1 – PyClone | Mingfeng Liu |
| Section 3.2 – Ccube | Yuhong Lin |
| Section 3.3 – PyhloWGS | Lukas Rubikas |
| Section 3.4 – NMF and LDA | Jesus Vera |
| Section 3.5 – Pipeline |  |
| **Chapter 4 – Design and Implementation** | Jesus Vera |
| Section 4.1 – PyClone | Mingfeng Liu |
| Section 4.2 – Ccube | Yuhong Lin |
| Section 4.3 – PyhloWGS | Lukas Rubikas |
| Section 4.4 – NMF | Razak Nart |
| Section 4.5 – LDA | Jesus Vera |
| Section 4.6 – Pipeline |  |
| **Chapter 5 – Evaluation** | Jesus Vera |
| Section 5.1 – First Phase | Lukas Rubikas |
| Section 5.2 – Second Phase | Jesus Vera and Razak Nart |
| **Chapter 6 – Conclusion** | Lukas Rubikas and Jesus Vera |
| **Chapter 7 – Contributions** | Jesus Vera |
| Section 7.1 – Final product | Jesus Vera |
| Section 7.2 – Report | Jesus Vera |
| References | Everyone |
| Appendix A – User Manual |  |

# References

[1] Roth, A., et al. (2014). PyClone: statistical inference of clonal population structure in cancer. Nature Methods, 11(4), p.396.

[2] 1000 Genomes Project Consortium (2010). A map of human genome variation from population-scale sequencing. Nature, 467(7319), p.1061.

[3] Harismendy, O., et al. (2011). Detection of low prevalence somatic mutations in solid tumors with ultra-deep targeted sequencing. Genome biology, 12(12), p. R124.

[4] Yuan, K., Macintyre, G., Liu, W. and Markowetz, F. (2018). Ccube: A fast and robust method for estimating cancer cell fractions.

[4] <https://bitbucket.org/aroth85/pyclone>

[5] Lee D, Seung H. Learning the parts of objects by non-negative matrix factorization. Nature. 1999;401(6755):788-791.

[6] Lee D, Seung H. Algorithms for non-negative matrix factorization. In Advances in neural information processing systems. 2001; 556-562.

[7] Müller A, Guido S. Introduction to Machine Learning with Python. 1st ed. Sebastopol: O’Reilly Media, Inc.; 2016.

[8] Shahnaz F, Berry M, Pauca V, Plemmons R. Document clustering using nonnegative matrix factorization. Information Processing & Management. 2006;42(2):373-386.

[9] Tsuge S, Shishibori M, Kuroiwa S, Kita K. Information Retrieval Using Non-Negative Matrix Factorization. IEEJ Transactions on Electronics, Information and Systems. 2004;124(7):1500-1506..

[10] Salim W, Santika D. Human Facial Expression Recognition Using Two-dimensional Non-Negative Matrix Factorization. Procedia Engineering. 2012;50:758-767.

[11] Devarajan K. Nonnegative Matrix Factorization: An Analytical and Interpretive Tool in Computational Biology. PLoS Computational Biology. 2008;4(7):e1000029.

[12] Lee C, Mudaliar M, Haggart D, Wolf C, Miele G, Vass J et al. Simultaneous Non-Negative Matrix Factorization for Multiple Large Scale Gene Expression Datasets in Toxicology. PLoS ONE. 2012;7(12):e48238.

[13] Blei, D., Ng A. and Jordan, M. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research 3.

[14] <https://scikit-learn.org/stable/modules/decomposition.html#latentdirichletallocation>

[15] Zhao, W., et al. (2014). Topic modeling for cluster analysis of large biological and medical datasets. BMC Bioinformatics.

[16] <https://scikit-learn.org/>

[17]<https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV>

[18] Decomposing signals in components (matrix factorization problems) — scikit-learn 0.20.1 documentation [Internet]. Scikit-learn.org. [cited 3 December 2018]. Available from: <https://scikitlearn.org/stable/modules/decomposition.html#nmf>

[19] Sklearn.decomposition.NMF — scikit-learn 0.20.2 documentation [Internet]. Scikit-learn.org. [cited 3 December 2018]. Available from: <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html>

# 

###### User Manual

<Use Heading 6 for the Appendix heading>