# Comparison of several forms of dimension reduction on cuantitative morphological features for normal, abnormal and reactive lymphocyte diferentiation.

*Giménez Gredilla, Daniel*

*November 2017*

## Contents

# 1 - Project's ongoing development description

Up to this date, a complete background for the project has been researched. The goal of the project is clear, and a good wealth of literature on the topic and multiple sub-topics at hand has been sought, found and studied. The ups and downs of both dimension reduction techniques and programming languages and packages have been assessed, evaluated and discriminated. The data quality and structure has been assessed. The redaction of a complete report is on its way, and the general state of development is in accordance with both the official timetables and the personally scheduled tasks.

## 1.1 - Level of goal accomplishment and planned results

By the end of the first phase of this project, all goals and tasks have been achieved. Referring to the original milestone table:

Table 1: Phase 1 milestones

| Deadline | Milestone |
|---|---|
| 01-NOV-2017 | Array of candidate bioinformatics languages, tools and protocols assessed |
| 20-NOV-2017 | Definitive subset of bioinformatics languages, tools and protocols selected |
| 20-NOV-2017 | Monitoring report for Phase 1 |

All milestones and tasks have been accomplished for the first phase of the project, so the original milestone table remains unchanged. The final bioinformatics languages, tools and protocols chosen for the second phase of the project are as follows:

### 1.1.2 - Dimension reduction techniques

**PCA:** PCA is the most used unsupervised, linear dimension reduction technique currently available. It is also the best, in the mean-square error sense (Fodor 2002). Its central idea is the construction of a set of features from a number of initial variables (Jolliffe 2002). The number of new features will be less than the initial variables, while retaining as much as possible of the initial variation. This is achieved by linear transformations of the original data, and then establishing a descending order of the new features attending to the amount of variation retained or explained by each of them.

**ICA:** Independent Component Analysis (*ICA*) is a statistical method for transforming an observed multidimensional random vector into components that are statisticaly as independent from each other as possible, this is, a tendency to **redundancy reduction** (Tobergte and Curtis 2013). In its linear approach, as with other dimension reduction algorithms, its goal is to take a zero-mean, $m$-dimensional variable, and by means of a linear transformation, find its $n$-dimensional transform, such that $n \leq m$, this transformation having some suitable properties. The vectors obtained from this transformation are neither orthogonal nor ranked in order.

**Factor Analysis:** The basic idea underlying Factor Analysis is that $p$ observed random variables, $\mathbf{x}$, can be expressed, except for an error term, as linear functions of $m(< p)$ hypothetical (random) variables or *common factors* (Jolliffe 2002). The aim of Factor Analysis is to group variables that share a "common theme" under the same grouping, such that the dimensionality of the dataset is decreased.

**Autoencoders:** An autoencoder is an unsupervised machine learning algorithm, with an emphasis on feature extraction, that applies backpropagation, setting the targets to be equal to the inputs. The aim of the autoencoder is to learn a function $h_{W_b}(x) \approx x$ (University, n.d.). Briefly explained, an autoencoder, through at least an input layer, an output layer and a hidden layer, tries to encode and decode data such that the output layer's result is as similar as possible to the original data, and, in the process, attempts to learn the identity function, this is, the central layer is the real goal. Even though autoencoders have enough freedom

to easily be able to overfit the model, when handicapped with different types of constraints they can find interesting traits of the data structure.

**T-distributed Stochastic Neighbor Embedding:** This is a nonlinear algorithm for dimension reduction. It was developed in 2008 by Laurens Van der Maaten and Geoffrey Hinton (Maaten and Hinton 2008). It's a variation of **Stochastic Neighbor Embedding** and improves it by allowing a better visualization of high-dimensional data lying in several, lower-dimension, related manifolds. This technique is allegedly able to capture much of the local structure of the original, high-dimensional data , while also revealing global structure such as the presence of clusters at several scales.

### 1.1.3 - Programming languages and implementations

**Programming language - Python: Python** is a powerful, system accessible, interpreted scripting language (B. W. J. Chun and Chun 2006). Data types like lists (resizable arrays) and dictionaries (hash tables) are built-in, providing a dynamic typing instead of having to declare types of variables, as is the case in **C++**. This reduces the framework development time. **Python** is an *OOJ* (*Object-Oriented Programming*), high-level, general-purpose language. It is initially developed with a focus on being easy to read and write (Granger and Hunter 2011), while also granting access to low-level proccesses, offering simple portability and well-defined exception catching and handling. Even so, it doesn't force this work model on the user, and can be actuated upon in a procedural way if needed.

**Programming language - R: R** (Team 2008) is a quite modern statistics-focused programming language. It is an implementation of the **S** language with lexical scoping semantics inspired by **Scheme**. **S** was developed at Bell Laboratories by John Chambers and coleagues, while **R** was developed by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. The project started its development in 1992, with a first release in 1995 and a stable beta in 2000. The base package of the program provides with native functions for many common-use mathematical workflows, and it is easily expanded via libraries and packages delivered in an Open Source environment through the CRAN site cluster.

**Python implementation - Scikit Learn: Scikit-learn** is a "toolbox" of implementations of many popular machine learning algorithms. It has been developed with researchers from fields outside of computer sciences to use, thus its simplicity of application for many machine learning problems. It is distributed under a *BSD* license, and it only has **NumPy** and **SciPy** as dependencies. It has even been distributed as part of main OS distributions such as Ubuntu or Debian (Pedregosa et al. 2012).

**Python implementation - NumPy and SciPy:** these two modules act as dependencies for **Scikit-learn**. **NumPy** (Walt et al. 2011) arrays are the standard object for data representation in **Python**. These arrays can have any number of dimensions and can contain other kinds of elements. **ScyPy** (Jones, Oliphant, and Peterson 2001), is a collection of algorithms and functions built on **Numpy**. It adds high-level commands and classes for manipulating and visualizing data.

**R implementation - prcomp():** (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/prcomp.html) *Prcomp()* is R's native implementation of **Principal Component Analysis**. It is parameterized with the options to scale and center data before the analysis, rank (the maximum number of principal components to be used) or the magnitude of the standard deviation below which components should be omitted.

**R implementation - icafast():** (A. N. E. Helwig and Helwig 2015) This package assesses the implementation of several **ICA** algorithms, including **FastICA**, **Infomax** and **JADE**. **FastICA** (Hyvärinen and Oja 2000) will be used for this project, accepting parameters like the number of components to extract, options to center data before ICA decomposition or convergence tolerance.

**R implementation - factanal()** (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/factanal.html) This function performs maximum-likelihood factor analysis on a covariance matrix or a data matrix. As parameters it accepts the number of factors to be fitted, the type of scores to be output (*Thompson's*, *Bartlett's*, etc), or the number of observations if the input given to the function is a covariance matrix.

**R implementation - RsdA():** (Package, Kou, and Sugomori 2015) The **RcppDL** package includes a kit of basic, multilayer machine learning algorithms, **Restricted Boltzmann Machines** and **Deep Belief Networks** amongst them. The *rsda()* function is a wrapper to initialise a *deeplearning* object implementing stacked denoising autoencoding on a set of data. It can be then pretrained, fine tuned and used for prediction or classification.

**R implementation - tsne():** (Package 2016) The **Tsne** package contains only one function, namely *tsne()*, an implementation of the **T-distributed Stochastic Neighbor Embedding** for R. It provides an interface for the application of **t-SNE** on **R** matrices or *dist* objects.

### 1.2 - Change justifiation (if necessary)

There have been no changes to the arranged timetables and planned content, so no appliable justification will be supplied.

## 2 - Relation of undertaken tasks

### 2.1 - Scheduled activities

1.1.1 - Choose a subset from the most used and widely applied dimension reduction techniques appliable to the present topic, including **PCA**, **ICA** and **Factor Analysis**. (1 week, 21 hours equivalent)

**State: Complete - In schedule.** Two additional techniques (**Autoencoders** and **T-distributed Stochastic Neighbor Embedding**) have been added to the pool for the reasons stated in the report.

1.2.1 - Elaborate a list of widely bioinformatics-applied languages (and frameworks, if used within one). (7 days, 21 hours equivalent)

**State: Complete - In schedule.** Languages such as **SPSS**, **Matlab** or **Haskell** have been assessed and evaluated for their usefulness and fit to the goals of this project.

1.2.2 - Choose a subset from those languages and frameworks and elaborate a briefing of characteristics and examples of application. (3 days, 9 hours equivalent)

**State: Complete - In schedule.** The final candidates for comparison, due to the factors described in the project, are Python and R.

1.3.1 - Elaborate a list of dimension reduction packages and functions from chosen languages. (7 days, 21 hours equivalent)

**State: Complete - In schedule.**

1.3.2 - Choose a subset and elaborate a briefing of package traits: optimal application, parameters, example workflows it has actually been used for, etc. (4 days, 12 hours equivalent)

**State: Complete - In schedule.** The chosen packages have been listed with a historical and mathematical background where appliable, and reasons for selection.

1.3.3 - Elaborate monitoring report for **Phase 1**. (7 days, 21 hours equivalent)

**State: Complete - In schedule.**

### 2.2 - Undertaken unscheduled activities

As a preliminary action to the analysis proper, an assessment of data traits and quality has been undertaken. The analysis includes structure for the principal factors, as well as a summary of descriptive and qualitative traits for numeric variables. This brief analysis is intended to help understand the context of the project.

# 3 - Relation of schedule deviations and buffering actions (if applicable) - Chronogram update

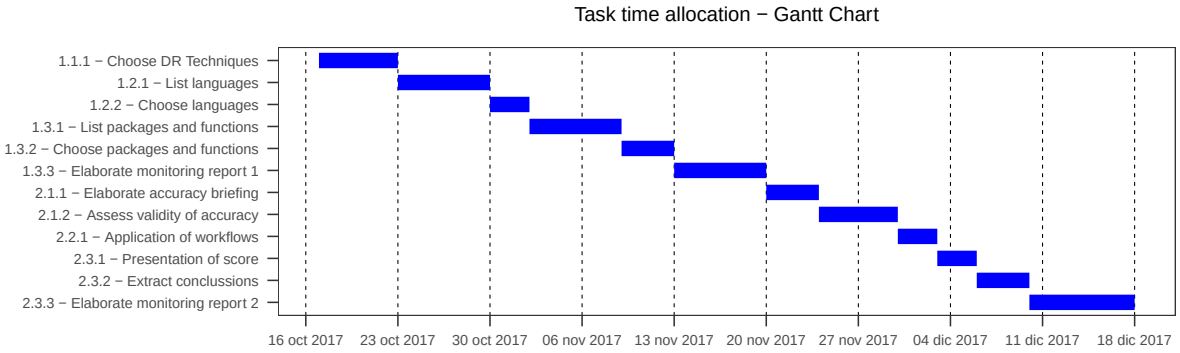
In the original planning, possible sources of deviation and obstacles were proposed as:

**1. Technical problems:** no technical problems have arisen from equipment malfunction, infrastructure breakdown or insufficient resources. The data set being treated is extensive both in number of observations and variables, and some scarcity in processing power could be expected. Even so, no important hindrances have appeared.

**2. Goal overextension:** a judicious selection of techniques and bioinformatics tools have allowed to include not only three of the main, more streamlined reduction techniques(**PCA**, **ICA** and **Factor Analysis**), but also two additions that have been deemed interesting for the current comparison (**Autoencoders** and **T-distributed Stochastic Neighbor Embedding**) along with implementations for them.

**3. Incompatibilities:** no incompatibilities have been found neither in practice nor in literature.

Referring to the original the Gantt diagram presented within the initial plan:




The planned tasks were completed in accordance with the originally scheduled dates. There have been no unforeseen hindrances or problems in the development of such tasks, so the original chronogram still holds without any deviations.

# 4 - List of partial results (with attached products)

**Data analysis script:** Script for the automated analysis of the data input. It presents the reader or reviewer of this project with a *go-to* descriptive statistics briefing of all variables, in which central and dispersion measures can be checked if problems arise and it is suspected that one variable may be anomalous.

**Data quality report:** The output of the data analysis script, it is a dynamic report on the main descriptive traits of the variables involved in the study. As it is stated in the file itself, **it is an information-heavy document**. It is not to be used as reading material of itself, but as a useful source of information about the dataset in case it is needed and a caution measure in any biostatistics project.

## 5 - Project Tutor's comments

## 6 - References


Chun, By Wesley J, and Wesley J Chun. 2006. *Core Python Programming , Second Edition.*

Fodor, Imola. 2002. "A Survey of Dimension Reduction Techniques." doi:10.1.1.8.5098.

Granger, Brian E, and John D Hunter. 2011. "Python : An Ecosystem," 13–21.

Helwig, Author Nathaniel E, and Maintainer Nathaniel E Helwig. 2015. "Package ' ica '."

Hyvärinen, Aapo, and Erkki Oja. 2000. "Independent Component Analysis: Algorithms and Applications." *Neural Networks* 13 (45): 411–30. doi:10.1016/S0893-6080(00)00026-5.

Jolliffe, I T. 2002. "Principal Component Analysis, Second Edition." *Encyclopedia of Statistics in Behavioral Science* 30 (3): 487. doi:10.2307/1270093.

Jones, Eric, Travis Oliphant, and Pearu Peterson. 2001. "SciPy: Open source scientific tools for Python." http://www.scipy.org/.

Maaten, Laurens Van Der, and Geoffrey Hinton. 2008. "Visualizing Data using t-SNE." *Journal of Machine Learning Research 1* 620 (1): 267–84. doi:10.1007/s10479-011-0841-3.

Package, Type. 2016. "Package ' tsne '," 2–5.

Package, Type, Author Qiang Kou, and Yusuke Sugomori. 2015. "Package ' RcppDL '."

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2012. "Scikit-learn: Machine Learning in Python" 12: 2825–30. doi:10.1007/s13398-014-0173-7.2.

Team, R Development Core. 2008. "R: A Language and Environment for Statistical Computing." Vienna: R Foundation for Statistical Computing. doi:3-900051-07-0.

Tobergte, David R., and Shirley Curtis. 2013. "Independent Component Analysis by Minimization of Mutual Information." *Journal of Chemical Information and Modeling* 53 (9): 1689–99. doi:10.1017/CBO9781107415324.004.

University, Stanford. n.d. "Unsupervised Feature Learning and Deep Learning Tutorial." ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/.

Walt, Stefan Van Der, S Chris Colbert, Gaël Varoquaux, Stefan Van Der Walt, S Chris Colbert, Gaël Varoquaux, and The Numpy. 2011. "The NumPy array: a structure for efficient numerical computation." doi:10.1109/MCSE.2011.37.