Nonlinear analysis toolbox for neurodegenerative diseases and aging

A Degree's Thesis submitted to the Faculty of the
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya

by

Santiago Puch Giner

In partial fulfillment
of the requirements for the Degree in

Science and Telecommunication Technologies Engineering

Advisor
Prof. Verónica Vilaplana Besler

Barcelona, September 2016

To my grandparents, Isabel and Pasqual

Abstract

Neurodegenerative diseases impose substantial public health burdens on populations throughout the world. Alzheimer's disease is among the major neurodegenerative diseases, and its causes and treatment are still unknown. Researchers around the world are conducting large data-driven studies in order to unveil the causes and biological mechanisms of such diseases, and for that reason automatic tools that allow to uncover statistically significant findings are needed.

To address this problem we present in this thesis a software toolbox that provides the tools to analyze the linear and nonlinear dynamics of gray-matter and study the statistical significance of such dynamics at the voxel level. The toolbox features various fitting methods and fit evaluation metrics, an automatic hyperparameters look-up algorithm and several visualization and comparison tools.

All the features provided in this toolbox were tested in two real problems provided by the Pasqual Maragall Foundation, and it yielded results that were validated by the findings in the original studies.

Resum

Les malalties neurodegeneratives suposen una càrrega substancial en la sanitat pública de les poblacions d'arreu del món. La malaltia d'Alzheimer es una de les malalties neurodegeneratives més importants, i les seves causes i tractament son encara desconeguts. Investigadors d'arreu del món estan duent a terme estudis impulsats per dades per tal de descobrir les causes i els mecanismes biològics de les malalties abans mencionades, i per aquesta raó es requereixen eines automàtiques que permetin trobar resultats amb rellevància estadística.

Per abordar aquest problema presentem en aquesta tesi un software que proporciona les eines necessàries per analitzar els patrons dinàmics lineals i no lineals de la matèria grisa i estudiar la rellevància estadística d'aquests patrons a escala de *voxel*. El software ofereix diversos mètodes d'ajust de dades i de mètriques de la qualitat de l'ajust, un algoritme automàtic de cerca d'hiper paràmetres i vàries eines de comparació i visualització.

Totes les característiques proporcionades en aquest software van ser provades en dos estudis reals proporcionats per la Fundació Pasqual Maragall, i els resultats generats van ser validats pels resultats dels estudis originals.

Resumen

Las enfermedades neurodegenerativas suponen una importante carga para la sanidad pública de todas las poblaciones del mundo. La enfermedad de Alzheimer se encuentra entre las enfermedades neurodegenerativas mas importantes, y sus causas y tratamiento no son conocidos todavía. Debido a esto, investigadores de todo el mundo están realizando estudios impulsados por datos con el objetivo de descubrir las causas y los mecanismos biológicos de susodichas enfermedades, por lo que se requieren herramientas automáticas que permitan desvelar resultados que sean estadísticamente significativos.

Para afrontar este problema presentamos en esta tesis un software que proporciona las herramientas para analizar los patrones lineales y no lineales de la materia gris y estudiar la significatividad estadística de tales dinámicas a escala de *voxel*. La herramienta de software incluye diversos métodos de ajuste y métricas de evaluación del ajuste, un algoritmo de búsqueda de hiper parámetros y varias herramientas de visualización y comparación.

Todas las características proporcionadas en este software fueron probadas en estudios de la Fundación Pasqual Maragall, y los resultados generados por éste fueron validados por los resultados de los estudios originales.

Acknowledgements

First of all I want to thank my project advisor, Verónica Vilaplana Besler, for giving me the peerless opportunity of being involved in this project, for her wise advice and for trusting me all time.

I also want to thank Asier Aduriz and Adrià Casamitjana for being such amazing colleagues, for the amount of work they've put into this project and for being so easy to work with.

I want to thank Juan Domingo Gispert, Stavros Skouras, Raffaele Cacciaglia and Alan Tucholka too, for their advice and for the time they have dedicated to enrich this project, and also thank all the people in Pasqual Maragall Foundation for working for a future without Alzheimer.

Last but not least, I want to sincerely thank my family and friends for always being there and for their support in the moments of need.

Revision history and approval record

Revision	Date	Purpose
0	17/09/2016 - 18/09/2016	Document creation
1	19/09/2016 - 23/09/2016	Document writing
2	24/09/2016	Document revision
3	25/09/2016	Document correction and writing
3	26/09/2016	Document writing
3	27/09/2016	Document revision and correction
3	28/09/2016	Document approval

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Santiago Puch Giner	santiago.puch@alu-etsetb.upc.edu
Verónica Vilaplana Besler	veronica.vilaplana@upc.edu

Written by:		Reviewed and approved by:	
Date	28/09/2016	**Date**	28/09/2016
Name	Santiago Puch Giner	**Name**	Verónica Vilaplana Besler
Position	Project Author	**Position**	Project Advisor

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Statement of purpose

The Magnetic Resonance Imaging (MRI) imaging modality has enjoyed tremendous growth in the past years, and by now it is firmly established as one of the preferred diagnostic imaging tools. In particular, the brain imaging field has taken advantage of such advances and improvements in MRI, and by now the amount of information we can obtain regarding the anatomy and functionality of the brain is outstanding.

Physicians, statisticians, computer scientists, etc. around the world have seen this step forward in brain imaging as an opportunity to study the brain's pathologies and diseases.

One of the main neurodegenerative diseases that the mankind currently suffers from is Alzheimer's disease (AD). It is characterized by a progressive cognitive decline, and it is (by the time of writing this thesis) the 6th leading cause of death in the United States.

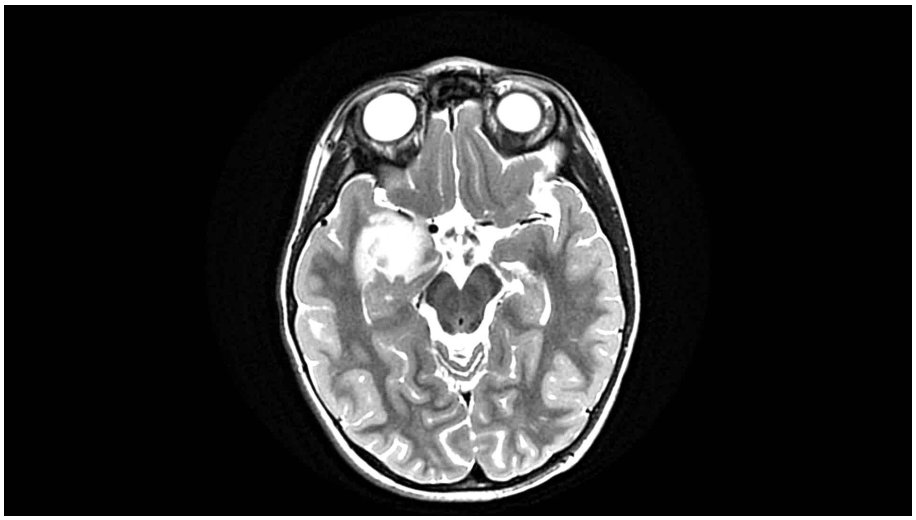

Figure 1.1: Example of a T2 weighted image, a modality of brain imaging using MRI

At Pasqual Maragall Foundation (FPM)[1] they research into how to prevent Alzheimer's disease and how to slow its onset. Particularly, they have studied the cerebral atrophy patterns across the AD continuum, and they are currently studying the atrophy patterns in aging with regards to a certain APOE4 genotype.

What these two studies (and others alike around the globe) have in common is that the researchers that conduct them require several tools to process the raw image data to obtain results that are statistically significant. Even sometimes the main neuroimaging software packages don't provide the functionality they require and they have to code scripts them-

[1] `https://fpmaragall.org/en/`

selves, which are very specific and not reusable at all.

This project tries to address this problem with the creation of a toolbox that provides the functionality required to study the linear and non-linear patterns of brain atrophy in neurodegenerative diseases and aging at the voxel level.

Based on the aforementioned motivation, the main objectives of this project are:

- Research about fitting methods that are convenient to analyze non-linear patterns
- Research about statistical tests and fit evaluation metrics that are relevant to the brain imaging field
- Conceive visualization tools that allow the user to have better insight in the results
- Implement a Command Line Interface (CLI) toolbox that includes the previous items
- Evaluate the performance and usability of the toolbox with the data from the "AD-CSF and the Alzheimer's disease continuum" study
- Evaluate the performance and usability of the toolbox with the data from the "Aging atrophy with regards to the APOE4 genotype" study

1.2 Requirements and specifications

The requirements of this project are the following:

- Provide several fitting methods that are capable of fitting non-linear patterns
- Implement statistical tests and fit evaluators that assess the statistical significance of the fitting
- Develop a method to find hyperparameters for fitters that require it
- Integrate visualization tools that allow the user to visualize the curves and the statistical maps and the statistical distributions of the data
- Include strategies to compare statistical maps
- Implement a method to perform statistical tests to compare between categories of subjects
- Implement a data loader that is capable of optimally loading morphological data of the brain in NIfTI format
- Implement a configurable, modular and extendable software that follows good programming practices and is documented

The specifications are the ones that follow:

- Use Python as programming language

- Use Nibabel[2] to load data in NIfTI format
- Use scikit-learn[3] as a machine learning library
- Use matplotlib[4] and seaborn[5] for the visualization functionality

1.3 Methods and procedures

This project was carried out at the Image and Video Processing Group (GPI) research group from the Signal Theory and Communications Department (TSC) at the Universitat Politècnica de Catalunya (UPC) in collaboration with the Pasqual Maragall Foundation (FPM).

The work presented in this thesis is the natural continuation of the work presented in [Aduriz Berasategi, 2016]. The FPM originally asked the GPI to collaborate in the study of the non-linear patterns across the AD continuum based on the paper they had previously published about this topic ([Gispert et al., 2015]). The idea was to try new fitters and new statistical tests, but instead of rooting for a straightforward but not reusable solution, both the GPI and FPM agreed to start a toolbox that would be flexible, easy to use and extendable.

The base classes and the structure of the toolbox, the GLM based fitters and the majority of the fit evaluation metrics were already integrated in the toolbox by the time I joined the research group. From that moment on Asier Aduriz, Adrià Casamitjana and myself, Santiago Puch, were all involved in the development of the toolbox, and together with the two advisors, Verónica Vilaplana (UPC) and Juan Domingo Gispert (FPM), we created and improved the toolbox little by little until the final version, which is the one that this thesis presents.

1.4 Document structure

In chapter 2 two medical problems are presented, namely *AD-CSF and the Alzheimer's disease continuum* and *Aging atrophy with regards to the APOE4 genotype*, and a survey of the existing literature about these problems is provided.

After introducing the project, the motivation and the problems, the methodology of the project is explained in chapter 3. This chapter is the most extensive one in this document, as it contains detailed explanations of the whole theoretical content upon which the toolbox implementation is based.

The purpose of chapter 4 is two-fold: first it provides statistically significant results of the two previously presented problems that are coherent with the ones found by the FPM; and second it validates the flexibility and usefulness of the toolbox.

Finally, the reached conclusions and the plans for the future development of this project are presented in chapter 5.

[2] `http://nipy.org/nibabel/`
[3] `http://scikit-learn.org/stable/`
[4] `http://matplotlib.org/`
[5] `https://stanford.edu/~mwaskom/software/seaborn/`

Chapter 2

State of the art

2.1 AD-CSF and the Alzheimer's disease continuum

As [Bateman et al., 2012] reports, "well-validated biomarkers of Alzheimer's disease processes are needed to improve the design of clinical trials, develop more effective therapeutics, and offer the opportunity for prevention trials". [Molinuevo et al., 2013] addresses this topic by introducing the AD-CSF-index, an indicator composed by the sum of the normalized CSF concentrations of Aβ42 and t-tau (total tau) that reflects the degree of the pathology and determines where the patient is along the AD continuum.

Regarding the association between the characteristic neurodegeneration of the disease and CSF-related biomarkers, [Sabuncu et al., 2011] evidences the nonlinearities of cortical atrophy with respect to CSF biomarkers. Another nonlinear approach is taken in [Insel et al., 2015], where piecewise-linear splines are used to evaluate the nonlinear nature of the association between CSF Aβ and regional atrophy and to identify points of acceleration of atrophy with respect to Aβ.

Finally, [Gispert et al., 2015] uses a polynomic approach to characterize the nonlinear volumetric changes in gray matter across the disease's spectrum — represented by the AD-CSF-index presented in [Molinuevo et al., 2013] — and the associated impact of the APOE4 genotype, reporting significant nonlinear dependencies in specific memory-related areas of the brain as a result of the analysis.

2.2 Aging atrophy with regards to the APOE4 genotype

Although there are several genetic and environmental variables that determine the cause and progression of AD, there is a large consensus in attributing to a variant in the APOE gene (the ϵ4 allele carriers) the strongest genetic risk factor for both early and late-onset AD development, as reported in [Cherbuin et al., 2007] and [Poirier et al., 1993]. Not only is the presence of APOE ϵ4 allele the main genetic determinant of AD risk but also of age-related cognitive decline during normal aging ([Liu et al., 2013]).

Some studies have been conducted in order to characterize the relationship between the APOE genotype and the brain atrophy in cognitively healthy individuals. [Wishart et al., 2006] reports reduced grey matter volume in several brain regions when comparing ϵ3 homozygote and ϵ3-ϵ4 heterozygote carriers, and concludes that regionally reduced gray matter density is detectable in cognitively intact adults with a single copy of the APOE ϵ4 allele. Another study reports significantly decreased volume in the hippocampus — a major component of the brain that is agreed to have a key role in the formation of new memories — for ϵ4-allele carriers compared to ϵ2-allele carriers ([Alexopoulos et al., 2011]).

More recently, [Cacciaglia et al., 2016] characterizes the impact of the APOE genotype on brain atrophy due to aging in cognitively healthy individuals by modeling the patterns of gray-matter variability across age depending on the APOE status.

Chapter 3

Methodology

3.1 Datasets

In this project two different datasets provided by the FPM have been used to assess the performance of the toolbox and, at the same time, to compare the obtained results with the ones found by the researchers at FPM. The purpose of this section is to summarize their nature. Note that all procedures described in this section were performed by the FPM.

3.1.1 ADCSF and the Alzheimer's disease continuum

The cohort for this study consisted of 129 participants that were divided in 4 groups depending on the stage of the disease: 62 belonged to the control group, which means that they didn't present any evidence of cognitive impairment and were CSF $A\beta$ negative; 18 were categorized as preclinical AD due to presence of positive CSF $A\beta$ values (despite the lack of evidence of cognitive impairment); there were 28 MCI (mild cognitive impairment) participants; and the remaining 21 participants belonged to the AD group, which means that they were already diagnosed with Alzheimer's disease. All the scans were processed using Voxel Based Morphometry as implemented in SPM[1]. Briefly, all T1[2] images were normalized to a reference template and the gray matter was segmented. A denoising procedure was performed and as a result voxels were assigned a value between 0 and 1. These values were modulated by the spatial normalization deformation, and finally all images were smoothed with a 6-mm full-width at the half maximum Gaussian kernel (FWHM) [3]. For a detailed explanation of the dataset and the image processing of the images refer to [Gispert et al., 2015].

3.1.2 Aging atrophy with regards to the APOE4 genotype

The cohort for this study consisted of 533 cognitively healthy participants aged between 45 and 76 years that were categorized depending on the APOE genotype: 261 non-carriers of ϵ4-alleles; 207 ϵ4-heterozygotes, meaning that they only have one ϵ4-allele; and 65 ϵ4-homozygotes, which have two ϵ4-alleles.

The image processing of the brain scans was similar to the one described in subsection 3.1.1: the images were segmented into gray matter tissue using the SPM software, and located into a common space for subsequent normalization using a 9-affine parameter transformation. Grey matter images were then used to generate a reference template object of the sample, which was warped into a standard Montreal Neurological Institute (MNI) template. Finally, the images were modulated with the spatial normalization deformation and were smoothed with a 6-mm full-width at half maximum Gaussian kernel. Then again, for a detailed explanation of the dataset and the image processing of the images refer to [Cacciaglia et al., 2016].

[1] `http://www.fil.ion.ucl.ac.uk/spm/`

[2] One of the commonest MRI sequences that is characterized by its short Repetition Time (TR) and Time to Echo (TE)

[3] `https://en.wikipedia.org/wiki/Full_width_at_half_maximum`

3.2 The toolbox

3.2.1 General description

The toolbox comprises an independent *fitting library*, made up of different *curve fitting* and *fit evaluation* methods, a *processing* module that interacts with the aforementioned *fitting library* providing the formatted data obtained from the *file system*, several *visualization* tools and a *CLI interface* that allows the interaction between the user and the *processing* module, supported by a *configuration file*.

On top of that, a *data loader* is required for two reasons: the first is the heterogeneity of the data to be loaded (Excel file for metadata, NIfTI files for morphometric data, etc.); and the second is the large amount of data the system is required to handle (a typical gray-matter volume of a subject has more than 1M voxels, and the toolbox is required to handle studies with hundreds of subjects).

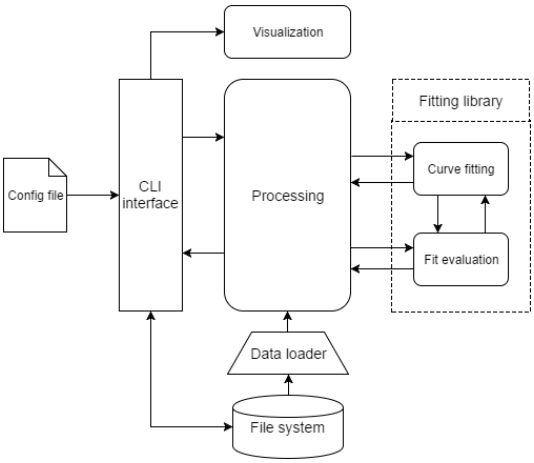

Figure 3.1: Block diagram of the toolbox

3.2.2 Curve fitting module

One of the main modules of the toolbox is the *curve fitting* library. This library is totally decoupled from the toolbox and is in charge of the actual fitting of the data.

In the context of this project a fitter is a method that finds a parametric function of one or more predictors and possibly one or more correctors that fits the target variable with the minimum loss (or alternatively the maximum quality measure). That is, if we assume:

$$\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{X}_{correctors}, \boldsymbol{X}_{predictors}) + \boldsymbol{\epsilon} \tag{3.1}$$

A fitter estimates the parameters $\boldsymbol{\theta}$ $(\theta_0, \theta_1, ..., \theta_{n-1})$ of the parametric function that minimizes the *loss* (L) between the *target variable* and the *prediction*:

$$\hat{\theta}_0, \hat{\theta}_1, ..., \hat{\theta}_{n-1} = argmin_{\boldsymbol{\theta}}(L(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{X}_{correctors}, \boldsymbol{X}_{predictors})) \tag{3.2}$$

Moreover, if we assume additive contributions of the covariates and the explanatory variables we can formulate Equation 3.1 as:

$$\boldsymbol{y} = f_{\boldsymbol{\alpha}}(\boldsymbol{X}_{correctors}) + f_{\boldsymbol{\beta}}(\boldsymbol{X}_{predictors}) + \boldsymbol{\epsilon} \tag{3.3}$$

And then we can estimate $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ separately:

$$\hat{\alpha}_0, \hat{\alpha}_1, ..., \hat{\alpha}_{c-1} = argmin_{\boldsymbol{\alpha}}(L(\boldsymbol{y}, f_{\boldsymbol{\alpha}}(\boldsymbol{X}_{correctors})) \tag{3.4}$$

$$\boldsymbol{r} = \boldsymbol{y} - f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{X}_{correctors}) \tag{3.5}$$

$$\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_{p-1} = argmin_{\boldsymbol{\beta}}(L(\boldsymbol{r}, f_{\boldsymbol{\beta}}(\boldsymbol{X}_{predictors})) \tag{3.6}$$

Here $\boldsymbol{y}$ is a random vector of n observations with potentially unknown statistical properties, $\boldsymbol{X}_{correctors}$ and $\boldsymbol{X}_{predictors}$ are matrices of dimensions $n \times m$ and $n \times k$ respectively, and $\boldsymbol{r}$ is the residuals vector obtained after substracting the contribution of the covariates to the observations.

This mathematical formulation translates into a execution pipeline like the one depicted in Figure 3.2

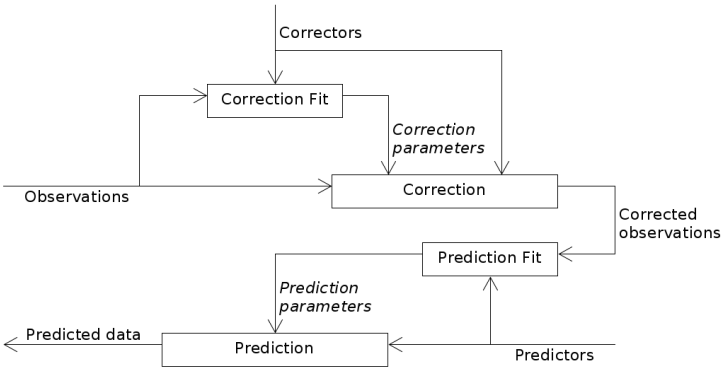

Figure 3.2: The curve fitting pipeline. The correctors are often called *nuisance variables* or *covariates* in the literature, while the predictors are called *variables of interest* or *explanatory variables*.

3.2.3 Processing module

Another core module of the toolbox is the *Processing* module. The main role of this module is to act as an intermediary between the CLI interface and the fitting library whilst using the *DataLoader* module to load the metadata from the Excel files, the NIfTI template and the morphometric data for all subjects. It is designed in a way that each *Processor* interacts with a *Fitter* to ask the user to input the required parameters for it to properly work. In order to integrate all processors and allow the user to use different fitting techniques for correction and for prediction we included the *MixedProcessor*, which adds a new level of abstraction and delegates the underlying correction and prediction related functions to the corresponding *Processor* instance.

Predictor separation by category

One important feature that the toolbox offers is the possibility to categorize the subjects of the study. For instance, if a particular study has a categorical variable that indicates the diagnosis of the subject, this can be added in the configuration file and modeled by the toolbox. This allows the user to separately fit the observations of each category (for example to study the differences in trends of brain atrophy due to the diagnostic), or to perform a statistical test that assesses how different the trends of brain atrophy are between one categorical variable and the rest (for example to compute a statistical map that shows which are the regions where these differences are more noticeable).

The former is trivial to implement: just select the observations and the samples of the predictors and correctors that belong to the selected category. But the latter involves a non-trivial implementation, which is going to be explained in the following paragraphs.

The idea is that the predictor used to model the data is split into M predictors, where M is the total number of categories, and each predictor has the same number of observations as the original one (n) by putting zeros on the samples that don't belong to its category.

If P is the predictor with n samples, and $s^{(i)}$ is a sample belonging to the ith category:

$$P = \begin{bmatrix} s^{(1)} \\ s^{(1)} \\ s^{(1)} \\ s^{(2)} \\ s^{(2)} \\ \vdots \\ s^{(i)} \\ \vdots \\ s^{(M)} \end{bmatrix} \Rightarrow S = \begin{bmatrix} s^{(1)} & 0 & \cdots & 0 & \cdots & 0 \\ s^{(1)} & 0 & \cdots & 0 & \cdots & 0 \\ s^{(1)} & 0 & \cdots & 0 & \cdots & 0 \\ 0 & s^{(2)} & \cdots & 0 & \cdots & 0 \\ 0 & s^{(2)} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & s^{(i)} & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & s^{(M)} \end{bmatrix} \tag{3.7}$$

Once the separation is performed, the toolbox asks the user which is the category that should be compared to the others, and then considers the selected category (say, i) as the predictor P, and adds the remaining categories to the correctors. Then, assuming that the correctors matrix C has as columns the correctors of the model, the final model (i vs the rest) would be

$$C = \{C \mid S_1 \mid S_2 \mid \cdots \mid S_M\} \tag{3.8}$$

$$P = S_i \tag{3.9}$$

where S_j is the jth column of S and $\mid$ is the concatenation operator.

An example of an application of this feature can be found in section 4.2.

3.2.4 Software implementation

As stated in the specifications section (1.2), the programming language used to implement this toolbox is Python. The reasons why we chose this particular programming language were several: it is an easy to learn but feature rich programming language, it supports several programming paradigms —object-oriented programming (OOP) or functional programming, for instance—, has mature scientific computing libraries such as Numpy or Scipy and is widely used by the community for statistical analysis, as it is R or MATLAB.

The whole toolbox has been implemented almost exclusively with an OOP paradigm in mind, which has allowed us to use techniques like inheritance or polymorphism to make it modular and extendable. The latter is one of the strongest points of the implementation, as it allows the current developers and future ones to add new fitters, new processors or new fit evaluation methods by overriding the corresponding abstract classes.

Some unit tests[4] have been implemented to automatically test the toolbox, hence providing an added layer of quality assurance (QA). Despite having only a few unit tests that only cover a small percentage of the code, the implementation of such unit tests provides a starting point for the developers to start covering more classes and methods, and denotes that the toolbox is ready to be automatically tested.

Furthermore, a lot of effort has been put in documenting the main classes of the toolbox and the toolbox itself, and also to be compliant to the PEP8[5] code style guideline, in order to increase the readability of the code and therefore make the code base maintainable and extendable.

For further details on the implementation of the toolbox refer to [Aduriz Berasategi, 2016].

3.3 Fitters

3.3.1 General Linear Model (GLM)

The General Linear model is a generalization of multiple linear regression to the case of more than one dependent variable. Having in mind the multiple linear regression equation

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k + \epsilon \tag{3.10}$$

the GLM differs from it in that the Y vector of n observations of a single Y variable can be replaced by a Y matrix of n observations of m different Y variables. Similarly, the β vector of regression coefficients for a single Y variable can be replaced by a β matrix of regression coefficients, with one vector of β coefficients for each of the m dependent variables.

The matrix representation of this formulation is the following:

$$\underset{(n\times m)}{\mathrm{Y}} = \underset{(n\times(c+p))}{\mathbf{X}} \underset{((c+p)\times m)}{\beta} + \underset{(n\times m)}{\varepsilon} \tag{3.11}$$

In our case, the variables correspond to the concepts in Table 3.1.

But how do we estimate the β coefficients? The most popular estimation method for the GLM is Least Squares, in which we pick the β coefficients that minimize the Residual Sum of Squares (RSS).

[4] `https://docs.python.org/2/library/unittest.html`
[5] `https://www.python.org/dev/peps/pep-0008/`

Variable	Description
Y	Matrix of target variables, that is, the n observations of gray-matter values for each of the m voxels
X	Matrix of p predictors and c correctors, which is commonly known as the *design matrix*
β	Unknown parameters that must be estimated in order to explain the target variables as a linear combination of the predictors and the correctors
ϵ	Matrix of m random errors

Table 3.1: Description of the General Linear Model variables

$$RSS(\beta_m) = \sum_{i=1}^{n} [y_{i,m} - (\beta_{0,m} + \sum_{j=1}^{c+p} \beta_{j,m} x_{i,j})]^2 \tag{3.12}$$

where β_m is the vector of coefficients of the mth target variable, that is, the mth column vector of β.

For the sake of simplicity let us assume that we only have one target variable Y, that we will express from now on as y. We can write Equation 3.12 as:

$$RSS(\beta) = (y - X\beta)^T (y - X\beta) \tag{3.13}$$

Differentiating this equation with respect to the β coefficients we obtain:

$$\frac{\partial RSS(\beta)}{\partial \beta^T} = -2X^T(y - X\beta) \tag{3.14}$$

To find the minimum we put Equation 3.14 equal to 0, and solve the equation for the β coefficients:

$$-2X^T y + 2X^T X\beta = 0 \iff X^T X\beta = X^T y \iff \beta = (X^T X)^{-1} X^T y \tag{3.15}$$

If we interpret these results geometrically, we observe that $\hat{y} = X\beta$ is a vector in the subspace defined by the columns of X (the predictors and the correctors). Furthermore, what Least Squares does is minimize the distance between this vector and the observations of the target variable y, which is accomplished when $\hat{y}$ is the projection of y in such subspace, i.e., when $\epsilon \perp \hat{y}$ (see Figure 3.3)

However, here we have implicitly assumed that $X^T X$ is not singular and therefore there is a unique solution for the β coefficients. This assumption does not held when the columns of X are not linearly independent, hence X is not of full rank. This is the case for correlated error among samples.

Fortunately the GLM is also able to treat this case. If we have a non-singular covariance matrix $C_{nxn} = \sigma^2 V_{nxn}$, where $V_{nxn} = \sum_i \lambda_i Q^i_{nxn}$ and Q^i_{nxn} is the matrix corresponding to the ith covariance component of the model, the Ordinary Least Squares (OLS) problem becomes a Weighted Least Squares (WLS), whose solution is:

$$\beta = (X^T V^{-1} X)^{-1} X^T V^{-1} y \tag{3.16}$$

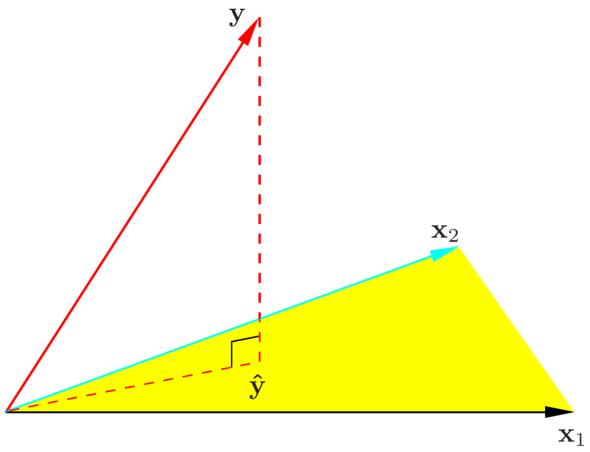

Figure 3.3: Geometric interpretation of the Least Squares solution in a 2-dimensional space. The red dotted line that is orthogonal to the yellow plane represents the error ϵ.
Figure taken from [Hastie et al., 2009]

.

Assuming $V^{-1} = W^T W$, we can write Equation 3.13 as:

$$\beta = (X^T W^T W X)^{-1} X^T W^T W y = ((WX)^T WX)^{-1} (WX)^T W y = (X_w^T X_w)^{-1} X_w^T y_w \quad (3.17)$$

Which is the OLS equation (see Equation 3.12) but with the *whitened* versions of X and y, which are X_w and y_w respectively.

Curve fitting definition of GLM

Regarding the mathematical definition of the curve fitting block in Equation 3.4 and Equation 3.6, we have the following table of correspondence:

Curve fitting	**GLM**
$L(\cdot)$	$RSS(\cdot)$
$f_\theta(X)$	$X\beta = \beta_0 + \beta_1 X_1 + ... + \beta_{(c+p)} X_{(c+p)}$

Table 3.2: Correspondence between the curve fitting and the GLM formulation

3.3.2 Generalized Additive Model (GAM)

Although appealingly simple, the General Linear Model often fails in real-life examples due to the constraint of being linear. Even though that a basis expansion and non-linear inter-dependencies between the regressors can be added, sometimes an inherently non-linear approach is needed.
The Generalized Additive Model is a Generalized Linear Model (not to be confused with General Linear Model, GLM) in which the observations depend linearly on unknown smooth functions of some predictor variables. It was first introduced in [Hastie and Tibshirani, 1990] as a novel technique that is capable of uncovering nonlinear covariate effects and has the advantage of being completely automatic, i.e., no exploratory work is needed on the part of the statistician.

A generalized additive model has the form

$$Y = \alpha + f_1(X_1) + f_2(X_2) + ... + f_k(X_k) + \epsilon \tag{3.18}$$

Here the $X_1, X_2, ..., X_k$ represent the explanatory variables and Y the target variable, the $f_1, f_2, ..., f_k$ are non-parametric smooth functions, and the error ϵ has zero mean. The difference between the GAM and the Generalized Linear Models is that, instead of modeling each function as a basis expansion and then fitting the model using OLS, each function is fitted using a scatterplot smoother (e.g. a cubic spline smoother) by means of an algorithm (see algorithm 1) that simultaneously estimates the k functions.

Given observations (x_i, y_i) a criterion like the penalized residual sum of squares (PRSS) can be specified for the problem of estimating the non-parametric functions:

$$PRSS(\alpha, f_1, f_2, ..., f_k) = \sum_{i=1}^{n}(y_i - \alpha - \sum_{j=1}^{k} f_j(x_{ij}))^2 + \sum_{j=1}^{k} \lambda_j \int [f_j''(\tau_j)]^2 d\tau_j \tag{3.19}$$

Here the λ_j are regularization parameters that penalize the roughness of the non-parametric function, being the two limit conditions $\lambda_j = 0$, which means that f_j can be any function that interpolates the observations, and $\lambda_j \to \infty$, which means that f_j is the linear function — as no second derivative is tolerated — that best fits the data.

It can be shown ([Hastie et al., 2009]) that the functions that minimize this criterion are cubic splines in the component X_j, with knots at each of the unique values of x_{ij}. However, there are some conditions that must be fulfilled in order for the solution to be unique:

1. $\sum_{i=1}^{n} f_j(x_{ij}) = 0 \ \forall j$: the functions average zero over the data.
2. The matrix of input values (ijth entry is x_{ij}) X has full column rank.

The iterative algorithm that finds the solution is known as the *backffiting* algorithm:

Algorithm 1 Backfitting algorithm

$\hat{\alpha} \leftarrow \frac{1}{N}\sum_{i=1}^{n} y_i$ ▷ Initialize elements
for $j = 1 \dots k$ **do**
  $\hat{f}_j \leftarrow 0$
end for
while $\hat{f}_j$ has not converged for some j **do**
  for $j = 1 \dots k$ **do**
    $\hat{f}_j \leftarrow \mathbf{S_j}\left[y - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(X_k)\right]$ ▷ Smooth function
    $\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N}\sum_{i=1}^{N} \hat{f}_j(x_{ij})$
  end for
end while

Here the $\boldsymbol{S}_j$ is a cubic smoothing spline, but other fitting methods can be accommodated in this algorithm by choosing the appropriate smoothing operator, such as:

- Local polynomial regression
- Kernel smoothing methods
- Surface smoothers

Curve fitting definition of GAM

As it has been previously explained, the *backfitting* algorithm does not find the parameters that define a parametric function but instead finds the function itself, so if we assume that the formula in Equation 3.2 has no θ parameters, the following correspondence can be defined:

Curve fitting	**GAM**
$L(\cdot)$	$PRSS(\cdot)$
$f(X)$	$\alpha + \sum_{j=1}^{k} f_j(X_j)$

Table 3.3: Correspondence between the curve fitting and the GAM formulation

3.3.3 Support Vector Regression (SVR)

The basic idea behind Support Vector Regression is simple: assuming we have n training examples $(x_1, y_1), ..., (x_n, y_n)$, where $x_i \in \mathbb{R}^k$, the goal is to find a function $f(x)$ that has at most ϵ deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible.

For the sake of simplicity we begin by describing the linear case of the Support Vector Regression, and later on the non-linear case will be introduced via the implicit mapping with kernels. The linear function f may be expressed as:

$$f(x) = \langle w, x \rangle + b \tag{3.20}$$

Then we can write this problem as a convex optimization problem by requiring

$$\text{minimize } \frac{1}{2}\|w\|^2 \text{ subject to } \begin{cases} y_i - \langle w, x_i \rangle - b \leqslant \epsilon \\ \langle w, x_i \rangle + b - y_i \leqslant \epsilon \end{cases} \tag{3.21}$$

This assumes that a function f exists that approximates all pairs (x_i, y_i) with ϵ precision. Sometimes, however, this may not be the case, so to relax the unfeasible constraints of the optimization problem we can introduce slack variables ζ_i and ζ_i^*

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) \text{ subject to } \begin{cases} y_i - \langle w, x_i \rangle - b \leqslant \epsilon + \zeta_i \\ \langle w, x_i \rangle + b - y_i \leqslant \epsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \geqslant 0 \end{cases} \tag{3.22}$$

The C hyperparameter acts as a regularizer, and adjusts the trade off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated.

The formulation above corresponds to the so called ϵ-insensitive loss function, which is depicted in figure Figure 3.4.

We can now formulate the Lagrangian function from both the objective function (*primal* objective function) and the corresponding constraints, by introducing a dual set of variables:

$$\begin{aligned} L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) - \sum_{i=1}^{n}\alpha_i(\epsilon + \zeta_i - y_i + \langle w, x_i \rangle + b) - \\ \sum_{i=1}^{n}\alpha_i^*(\epsilon + \zeta_i^* - y_i + \langle w, x_i \rangle + b) - \sum_{i=1}^{n}(\eta_i\zeta_i + \eta_i^*\zeta_i^*) \end{aligned} \tag{3.23}$$

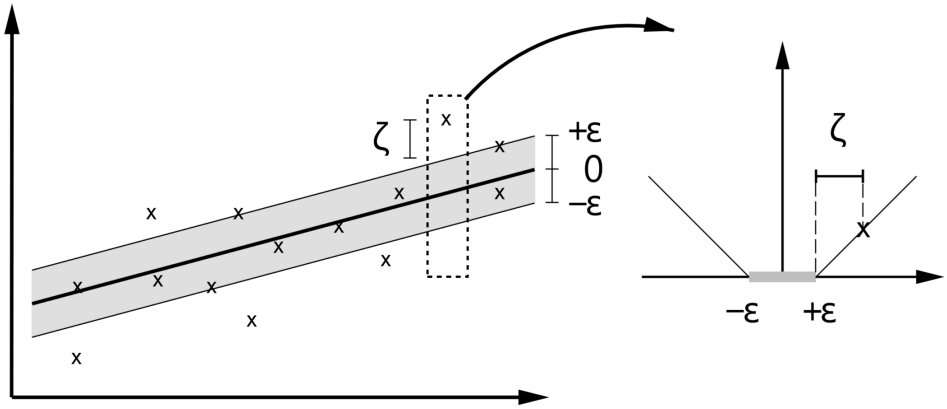

Figure 3.4: ϵ-insensitive loss function for Support Vector Regression

Now we take the partial derivatives of L with respect to the primal variables $(w, b, \zeta_i, \zeta_i^*)$ and we equal them to 0:

$$\begin{aligned} \frac{\partial L}{\partial b} &= \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0 \\ \frac{\partial L}{\partial w} &= w - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i = 0 \\ \frac{\partial L}{\partial \zeta_i^*} &= C - \alpha_i^* - \eta_i^* = 0 \end{aligned} \tag{3.24}$$

Combining all equations in Equation 3.24 into Equation 3.23 we get the dual optimization problem:

$$\text{maximize} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \end{cases} \quad \text{subject to} \begin{cases} \sum_{i=1}^{n}(\alpha_i + \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases} \tag{3.25}$$

The middle equation in Equation 3.24 can be written as

$$w = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i \tag{3.26}$$

Finally, if we substitute Equation 3.26 into Equation 3.20 we obtain the SVR solution:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b \tag{3.27}$$

where α_i and α_i^* are known as *dual coefficients*.

For now we have only dealt with the linear version of Support Vector Regression, but one of the main objectives of this project is to study non-linearities in the data, so now we are going to study how to make the SVR algorithm nonlinear. There are two options:

- Mapping functions Φ that explicitly map the inputs from their original input space $\mathfrak{X}$ into another feature space $\mathfrak{F}$ ($\Phi : \mathfrak{X} \to \mathfrak{F}$). A typical example is a polynomial expansion of degree d.
- Kernel functions $k(\cdot)$ that implicitly map the inputs from their original space into another (potentially high-dimensional) feature space. A kernel is defined as $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$

The former is a reasonable approach in problems where higher order polynomials and high dimensional spaces are not required, as it is computationally unfeasible for such cases (the number of different features is $\binom{k+d-1}{k}$, being d the degree of the polynomial expansion).

On the other hand, kernel functions are computationally cheaper in SVR, as its solution only depends on dot products between the input vectors. Thus, it suffices to know and use $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ instead of $\Phi(\cdot)$ explicitly. The SVR formulation using kernels is therefore:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) k(x_i, x) + b \tag{3.28}$$

Some typical examples of kernel functions used with Support Vector Regression and Support Vector Machines are:

- Polynomial of degree d: $k(x_i, x_j) = (\langle x_i, x_j \rangle + r)^d$
- Radial basis function (Gaussian): $k(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2)$
- Sigmoid: $k(x_i, x_j) = tanh(\gamma \langle x_i, x_j \rangle + r)$

Curve fitting definition of SVR

Curve fitting	**SVR**
$L(y, f(X))$	$\begin{cases} 0 & \text{if } \lvert y - f(X) \rvert \leqslant \epsilon \\ \lvert y - f(X) \rvert - \epsilon & \text{otherwise} \end{cases}$
$f_\theta(X)$	$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) k(x_i, x) + b$

Table 3.4: Correspondence between the curve fitting and the SVR formulation

3.4 Fit evaluation methods

Once a particular model has fitted the data an evaluation of the goodness of the fit can be performed. As this is done at the voxel-level one can compute statistical maps — a 3D image where each voxel has the score of the fit evaluation for that particular voxel — for each of the fit evaluation methods presented in this section.

3.4.1 MSE

Mean Square Error is easily one of the simplest and commonest fit evaluation metric. It simply assesses how close is the prediction to the actual observations in terms of quadratic

distance, and does not take into account the complexity of the model used to fit the observations.

MSE is defined with the following formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.29}$$

where y_i is the ith observation and $\hat{y}_i$ is the prediction of the model for the ith sample.

Considering the curve fitting pipeline in Figure 3.2, y_i is the ith corrected observation and $\hat{y}_i$ is the prediction of the model for the ith predictor sample.

3.4.2 Coefficient of determination (R^2)

The coefficient of determination is an evaluator that indicates the proportion of the variance of the target variable that is predictable from the explanatory variables. Put it in plain language, it provides a measure of how well future samples are likely to be predicted by the model.

If $\hat{y}_i$ is the predicted value of the ith sample, y_i is the corresponding observation and $\bar{y}$ is the mean of the observations defined as $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, then the R^2 score is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.30}$$

As it happened with MSE, the coefficient of determination does not penalize the complexity of the model that fits the observations, so it is possible to arbitrarily increase the score by increasing the complexity of the model, i.e., the more overfitted the model the greater is this coefficient.

3.4.3 Akaike Information Criterion (AIC)

The Akaike Information Criterion is a measure of relative quality of a model: given a collection of models AIC estimates the quality of each model relative to each of the other models, which means that it will tell which is the best model even if all of them fit the data poorly.

It is a criterion founded on information theory, as it offers a relative estimate of the information lost when a given model is used to represent the process that generates the data. In doing so, it deals with the trade-off between the goodness of fit of the model and the complexity of the model.

If k is the number of estimated parameters in the model (e.g. the number of regressors and the intercept term in multiple linear regression) and L is the maximum value of the likelihood function of the model, AIC can be computed as:

$$AIC = 2k - 2ln(L) \tag{3.31}$$

Particularly, the AIC score for a GLM model with k explanatory variables is

$$AIC = 2(k+1) + n \cdot (ln(2\pi) + ln(RSS) - ln(n) + 1) \tag{3.32}$$

that if we simplify by removing the constants (as it will be used to compare models that fit the same n observations) we obtain

$$AIC = 2(k+1) + n \cdot ln(RSS) \tag{3.33}$$

3.4.4 F-test

A t-test is a useful hypothesis testing statistic for both sample means and regression coefficients. Unfortunately, when we have more complicated hypotheses, this test no longer works. Hypotheses involving multiple regression coefficients require a different test statistic and a different null distribution.

The F-test can be used in regression problems to determine whether a particular part of a model is significantly improving the overall performance of the rest of the model. Consider two models, the restricted model $M_{restricted}$ and the full model M_{full}, such that $M_{restricted}$ is nested within M_{full}, that is, $M_{restricted}$ is a submodel of M_{full}. Then, we can measure how much the inclusion of $M_{full} - M_{restricted}$ in the model is improving the fit by comparing the variance of the fitting error when using the complete model, M_{full}, as opposed to using the restricted model, $M_{restricted}$.

Typically, model M_{full} will yield better results (a smaller error) than $M_{restricted}$, but what we want to measure is whether the difference is significant or not. Thus, we will perform an F-test where the null hypothesis (H_0) states that the variances of both errors are equal, while the alternative hypothesis (H_1) claims that the variance of the error when using M_{full} is smaller than when using $M_{restricted}$:

$$\begin{aligned} H_0 &: Var(MSE_{M_{restricted}}) = Var(MSE_{M_{full}}) \\ H_1 &: Var(MSE_{M_{restricted}}) \neq Var(MSE_{M_{full}}) \end{aligned} \tag{3.34}$$

Then the F-statistic is defined as:

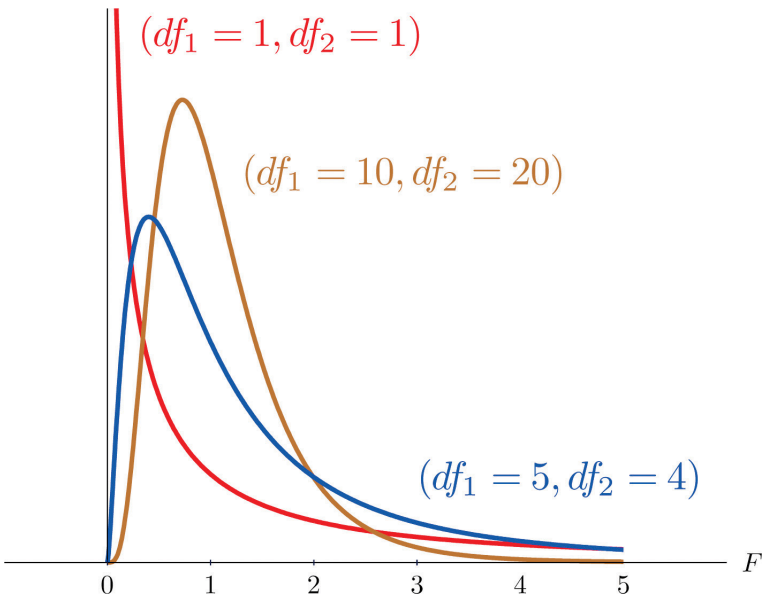

Figure 3.5: Examples of the probability density functions of several F-distributions with different degrees of freedom

$$F = \frac{(\frac{RSS_{restricted} - RSS_{full}}{dfrestricted - df_{full}})}{(\frac{RSS_{full}}{df_{full}})} \tag{3.35}$$

where $RSS_{restricted} = \sum_{i=1}^{n}(y_i - \hat{y}_i^{restricted})^2$ is the Residual Sum of Squares of the restricted model $M_{restricted}$ and $\hat{y}_i^{restricted}$ is the prediction of the ith sample using the restricted model, $RSS_{full} = \sum_{i=1}^{n}(y_i - \hat{y}_i^{full})^2$ is the Residual Sum of Squares of the full model M_{full} and $\hat{y}_i^{full}$ is the prediction of the ith sample using the full model, $df_{restricted}$ are the *degrees of freedom* of $M_{restricted}$ and df_{full} are the *degrees of freedom* of M_{full}.

Under the null hypothesis F will follow a F-distribution of parameters $(df_{restricted} - df_{full}, df_{full})$, that is, $F_{df_{restricted}-df_{full},df_{full}}$.

To reject the null hypothesis at the α significance level we need to compute the F-test from the obtained F-statistic (Equation 3.35) in order to obtain a p-value:

$$p\text{-}value = Pr(F_{df_{restricted}-df_{full},df_{full}} \geqslant F\text{-}statistic \mid H_0) \tag{3.36}$$

If the resulting p-value is less than the significance level, that is p-value $\leqslant \alpha$, then we can reject the null hypothesis.

The degrees of freedom of a model

In statistics the degrees of freedom may be defined as the number of values in the calculation of a statistic that are free to vary.

For a multiple linear regression model, the number of degrees of freedom is the number of observations of the data n minus the number of parameters to be estimated, that for the case of a multiple linear regression with p explanatory variables (including the intercept term) would be $df = n - p - 1$. Despite being trivial in the case of multiple linear regression or GLM, it is in general non-trivial for the other fitting methods.

The F-test is one of the most widely used statistical tests in the neuroimaging field, so there was a strong interest in generating statistical maps with this metric for all fitters.

The case of GAM using smoothing splines is explained in [Hastie et al., 2009], where the *effective degrees of freedom* are defined as the trace of the smoother matrix, $df_\lambda = tr(\boldsymbol{S}_\lambda)$.

The case of the SVR is even more complex. [Dinuzzo et al., 2007] introduces the concept of the *equivalent degrees of freedom* based on the notion of pseudoresiduals and the use of subdifferential calculus. As the mathematical background required to fully understand this article is out of the scope of this project, I will only present the formula to compute the pseudoresiduals, the sets of marginal support vectors and the final formula to compute the degress of freedom.

Considering the notation used in subsection 3.3.3 and $I = \{1, 2, 3, ..., n\}$, the pseudoresiduals are defined as

$$\eta_i = y_i - \sum_{j \in I, j \neq i} (\alpha_j - \alpha_j^*) k(x_i, x_j) = y_i - f(x_i) + a_i k(x_i, x_i) \tag{3.37}$$

From now on the dual coefficients $(\alpha_i - \alpha_i^*)$ will be expressed as a_i

The set of the marginal support vectors is defined as:

$$\begin{aligned} I_M^+ &= \{i \in I : f(x_i) - y_i = \epsilon\} \\ I_M^- &= \{i \in I : y_i - f(x_i) = \epsilon\} \end{aligned} \tag{3.38}$$

$$I_M = I_M^+ \cup I_M^- \tag{3.39}$$

And finally, the degrees of freedom of the SVR is the number m of marginal support vectors

$$m = |\{i \in I : \epsilon \leq |\eta_i| \leq \epsilon + Ck(x_i, x_i)\}| \tag{3.40}$$

3.4.5 Penalized Residual Sum of Squares (PRSS)

As we saw in Equation 3.19, Penalized Residual Sum of Squares is a fit evaluation method based on the fitting curve:

$$PRSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \int [f''(x)]^2 dx \tag{3.41}$$

Instead of penalizing the number of parameters of the model like the AIC or the degrees of freedom like the F-test, the PRSS penalizes the roughness of the fitting, thus making it a fit evaluation metric available for any fitter.

However, PRSS presents the problem of specifying the λ, which can be solved either heuristically (i.e. trial and error) or using some sort of hyperparameters searching method.

Furthermore, this method does not take into account the underlying variance of the predicted curve. For example, if we have n observations with 0 mean, low variance and a flat trend, the PRSS score will be low for that fitting as even a linear regression will fit the data almost perfectly, while if we have n observations with non-zero mean, high variance and a non-flat trend, no matter how good is our fitting that we will never obtain the same PRSS score as in the previous case. That poses a real problem in the context of morphological neuroimaging analysis, as a great percentage of the voxels are 0 valued for all subjects (e.g. voxels corresponding to areas with white matter or cerebrospinal fluid).

For that reason, we propose a variance normalized version of the PRSS, which is presented in the next section.

3.4.6 Variance Normalized - Penalized Residual Sum of Squares (VN-PRSS)

This new fit evaluation method intends to favor not only the best smooth fits, but also the ones that fit observations with non-flat trends, or conversely, it penalizes fits that do not adjust well to the observations, are rough, or are performed over observations with a flat trend.

The formulation is the same as PRSS but adding the normalization factor of the variance of the predicted curve:

$$VN\text{-}PRSS = \frac{1}{\frac{1}{n-1}\sum_{i=1}^{n}(\hat{y}_i - \frac{1}{n}\sum_{i=1}^{n}\hat{y}_i)^2}(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \int [f''(x)]^2 dx) \tag{3.42}$$

3.5 Hyperparameters search: GridSearch

The insightful reader may have noticed that the Support Vector Regression not only has normal parameters that are included in the optimization problem (i.e. the dual coefficients), but also depends on several hyperparameters to properly fit the observations, which are C and ϵ by default, and may include others depending on the kernel used (e.g. γ for the Gaussian kernel).

One possible approach to find such hyperparameters is the so called *trial and error*. The problem is that the search space is so big even with 2 parameters that this approach is unfeasible to do it manually.

For that reason we proposed an automatic method to find the optimal hyperparameters in the considered search space by sampling it in a grid, hence the name *GridSearch*. Here by optimal hyperparameters we mean the hyperparameters that minimize an error function (or loss function) that outputs how great is the error of the fit with respect to the observations (see subsection 3.5.3).

3.5.1 Description

Assuming there are p hyperparameters to be optimized, the idea of the GridSearch algorithm is to sample the p-dimensional search space in a grid, try to fit a subset of the data with the combination of hyperparameters of each sample in the grid, and find the combination that minimizes the error function of choice. Despite being a simple idea, it still presents some problems that need to be addressed:

1. The lack of validation data forces the searching algorithm to work with the training data itself, the morphological data of all subjects where the linear and nonlinear patterns ought to be found. This poses the problem of biasing the search towards overfitting the data, so in order to generalize a subset of m voxels is randomly selected (without repetition) for each iteration, and the search is performed over N iterations.

2. In voxels with low-variance data and flat trends (mentioned in subsection 3.4.5) the fitting will be easier than in voxels with high-variance data and non-flat trends, hence the error for such voxels will be smaller. Suppose then that in the ith iteration out of N we select m voxels like the former ones: the optimal hyperparameters will be found in this iteration, but such hyperparameters will bias the fit towards underfitting in voxels with high-variance and non-flat trends. Due to that, there is a minimum variance that a voxel must have in order to be randomly selected for a particular subset of m voxels, and the error for each voxel is weighted by the inverse of its variance, $\frac{1}{Var(y_j)}$.

The sampling procedure to obtain the grid considers four approaches. Considering the example of one hyperparameter that should be optimized using $start = a$, $end = b$ and $samples = s$, these are the four mentioned approaches:

- *Deterministic linear spacing*: the interval $[a, b]$ is partitioned into s sub-intervals with the same length, or equivalently, we obtain s equidistant samples between a and b.

- *Deterministic logarithmic spacing*: the interval $[10^a, 10^b]$ is partitioned into s sub-intervals with the same length, or equivalently, we obtain s equidistant samples between 10^a and 10^b.

- *Random linear spacing*: we find s random values between a and b and we order them in increasing order.
- *Random logarithmic spacing*: we find s random values between 10^a and 10^b and we order them in increasing order.

3.5.2 Algorithm

Algorithm 2 GridSearch algorithm

```
tot_err ← 0                                          ▷ Initialize total error to 0
optimal_hparam ← null                                ▷ Initialize optimal hyperparameters to null
fitter ← new Fitter()                                ▷ Initialize fitter
hparams ← get_hyperparameters()                      ▷ Initialize hyperparameters grid
for i = 1...N do
    cnt ← 0                                          ▷ Initialize voxel counter
    while cnt < m do                                 ▷ Select a subset of m voxels
        voxel ← select_random_voxel()
        if Var(voxel) > thr and voxel ∉ voxel_subset then
            voxel_subset ← voxel
            cnt ← cnt + 1
        end if
    end while
    obs ← get_observations(voxel_subset)             ▷ Observations of the m voxels
    for hparam in hparams do
        fitter.fit(obs, hparam)                      ▷ Fit the observations
        prediction ← fitter.predict()                ▷ Predict the fitted data
        df ← fitter.df()                             ▷ Get the degrees of freedom
        err ← error_function(obs, prediction, df)
        err ← 1/Var(obs) err
        err ← sum(err)
        if err < tot_err then
            tot_err ← err
            optimal_hparam ← hparam
        end if
    end for
end for
```

3.5.3 Error functions

Three error functions have been considered in order to optimize the hyperparameters:

1. MSE: Mean Squared Error optimizes the goodness of the fit but does not penalize the complexity of the model. It can be used with a linear SVR with polynomial basis expansion (PolySVR) if the degree of the polynomial is low, but it will cause overfitting when used with a SVR with Gaussian kernel (GaussianSVR).
2. C_p statistic: formulated as $C_p = MSE + 2\frac{df}{n}\hat{\sigma}^2$, it assesses the adjustment of the fit via MSE and the complexity of the model via degrees of freedom, df. It can be use with both PolySVR and GaussianSVR as it won't allow overfitting.

3. ANOVA-based error: it computes the p-value as in subsection 3.4.4. This error function is similar to the C_p statistics in terms of penalizing the complexity of the model, but it is more useful if we want to obtain F-test statistical maps afterwards, as it finds the optimal hyperparameters for that fit evaluation method.

3.6 Methods to compare statistical maps

Four methods to compare statistical maps have been conceived in the context of this project.

3.6.1 Description of the *BEST*, *RGB*, *absdiff* and *SE* methods

Name	**# of comparable maps**	**Description**
BEST	$2 \rightarrow \infty$	Creates a new map with the best fit score of all compared statistical maps for each voxel, and also creates an additional map where each voxel has the numerical label of the map with the best fit score. This means that if we compare m maps, and we number them in increasing order as $\{1, 2, ..., i, ..., m\}$, the vth voxel where the best fit score fs corresponds to the ith map, the value assigned to that voxel would be fs for the former and i for the latter.
RGB	2 or 3	Creates a statistical map where each compared map is assigned to an RGB channel. That way, if you compare 2 statistical maps, the first would be assigned to the red channel (R) and the second to the green channel (G), and if you add a third map to the comparison it would be assigned to the blue channel (B). The resulting is a 4-dimensional array with dimensions $2 \times dim1 \times dim2 \times dim3$ for the case of 2 inputs and $3 \times dim1 \times dim2 \times dim3$ for the case of 3 inputs.
Absolute difference (*absdiff*)	2	Computes the absolute difference between the two maps: $Y = \lvert X_1 - X_2 \rvert$, where X_1 and X_2 are the 3D maps to be compared with dimensions $dim1 \times dim2 \times dim3$ and the subtraction is element-wise.
Squared Error (*SE*)	2	Computes the squared error or distance between the two maps: $Y = \lVert X_1 - X_2 \rVert^2$, where X_1 and X_2 are the 3D maps to be compared with dimensions $dim1 \times dim2 \times dim3$ and the subtraction is element-wise.

Table 3.5: Description of the *BEST*, *RGB*, *absdiff* and *SE* methods to compare statistical maps

3.6.2 Use cases

There are different use cases in which each of this comparison methods are useful. Here we present some of them:

- **Select the best fitting model**

  If fitting the same observations of the same study with different fitters and/or different models for the same fitter, the *BEST* method is the method that will ease your model selection task. As it generates a label map with the model that best fits a particular voxel, you can see which regions are best fitted by which models.

- **Visualize the relative contribution of different models**

  If you have 2 or 3 models to fit your data and want to visually inspect which is the relative contribution of each model (in terms of fitting scores), then the *RGB* method is the one you should use. The output map can be visualized with tools like *FSLView*[6] in RGB mode in order to see the contribution of each compared statistical map as a combination of the primary colors for each region of interest.

- **Validate the similarity between 2 maps**

  If you want to ensure that two statistical maps are equal or almost equal, or otherwise you want to know where the two maps differ the most, then you can use the *SE* and the *absdiff* methods. The *SE* method is better for equality comparisons, as it enhances the big differences and lessens the small ones (due to the nature of quadratic distance), whilst the *absdiff* is better to visually inspect the regions where a difference between the 2 maps can be found.

3.7 Visualization tools

While having a variety of fitters to model your data and fit evaluation methods to assess the goodness of the fit is important, it is equally important to have tools that let you easily inspect the fitting and fit evaluation results. For that reason, two visualization methods have been considered in this project:

1. **Curve visualization**

   This method allows the user to visualize the fitted curves with respect to the observations for a given voxel. The shown figure consists of a scatter plot of the corrected observations of the specified voxel and a curve plot of the prediction of the model over such corrected observations. This tool can show overlapping or independent plots of different models: the former allows the user to inspect the differences between the fitted curves and the latter provides a detailed plot of a specific model.

   An example of the curve visualization can be found in Figure 4.7.

2. **Graphical visualizer**

   While the previous tool is useful to inspect the curves for a particular voxel, it is somewhat rigid in the sense that you can't quickly change the voxel that you are visualizing and the fact that you have no visual reference of the position of that particular voxel with respect to the brain. On top of that, a user can have better insight of the results

[6] `http://fsl.fmrib.ox.ac.uk/fsl/fslview/`

by visualizing the curves of the voxels where the fit scores are higher. These previous reasons led us to conceive a visualization tool that, in fact, isn't provided in any major neuroimaging software package: a graphical visualizer that shows the coronal, sagital and frontal views of the reference template with an statistical map overlaying it, and a figure on the side with the corrected observations and the predicted curve (or curves) for the selected voxel.

So, in a sense, this tool allows you to visualize what the *curve visualization* does with the addition of having a visual reference of the brain and the statistical map and the ability to select with the cursor the voxel to visualize.

One example of the graphical visualizer can be found in Figure 4.4.

Chapter 4

Results

4.1 AD-CSF and the Alzheimer's disease continuum

In this section the results obtained from the toolbox for the dataset *AD-CSF and the Alzheimer's disease continuum* will be presented. As [Aduriz Berasategi, 2016] presented significant results computed with a previous version of the toolbox that were coherent with the ones found in [Gispert et al., 2015], we will orient this section in presenting alternative methods to uncover other interesting findings, while providing examples of the capabilities of the toolbox. Despite that, the models used to fit the data will be as similar as possible to the ones used in [Gispert et al., 2015], that is, all the observations will be corrected by a polynomial GLM with *age* and *gender* as correctors, being the former a 2nd degree polynomial and the latter linear, and the predictor, which is the *AD-CSF index*, will be modeled as a 3rd degree polynomial in the polynomial-based fitters.

The first experiment consisted of fitting the two SVR based fitters to the data and compute an F-test statistical map on both of them. The goal of this experiment was to see how the SVR based fitters behave (in terms of non-linearity and statistical significance) and also see if they provide more information than the common polynomial GLM, which is the one used in [Gispert et al., 2015]. In order to fit the polynomial SVR and the Gaussian SVR a previous step was taken: the hyperparameters search, which yielded the following hyperparameters when using the *ANOVA* error function, deterministic linear spacing for ϵ and deterministic logarithmic spacing for C and γ (see Figure A.1):

Name	**C**	**ϵ**	**γ**
Polynomial SVR	1.65	0.078	-
Gaussian SVR	0.301	0.1	1.995

Table 4.1: Polynomial and Gaussian SVR hyperparameters for the *AD-CSF and the Alzheimer's disease continuum*

For this hyperparameters the resulting heat-maps of the F-test are Figure 4.1 for the Polynomial SVR and Figure 4.2 for the Gaussian SVR.

One can easily notice the difference in the size and number of activated regions between the two fitters, as it is clear that the polynomial SVR has statistical significance in several regions that the Gaussian SVR has not. This may be due to the incremented non-linearity of the Gaussian SVR that causes the model to have more degrees of freedom and, therefore, to have less statistical power. However, here the actual value of the statistical test cannot be easily distinguished between one and other.

The second experiment tried to overcome the mentioned limitation by producing a *BEST fit model* map between the transformed Z-scores of the polynomial GLM, the polynomial SVR and the Gaussian SVR, and then visualizing the voxels in which each of the SVR-based fitters have better fitting scores. Again, the compared maps are Z-scores obtained as a transformation of the p-values produced by the F-test, which have been filtered by a significance level of 0.001 and by minimum cluster size of 100 voxels.

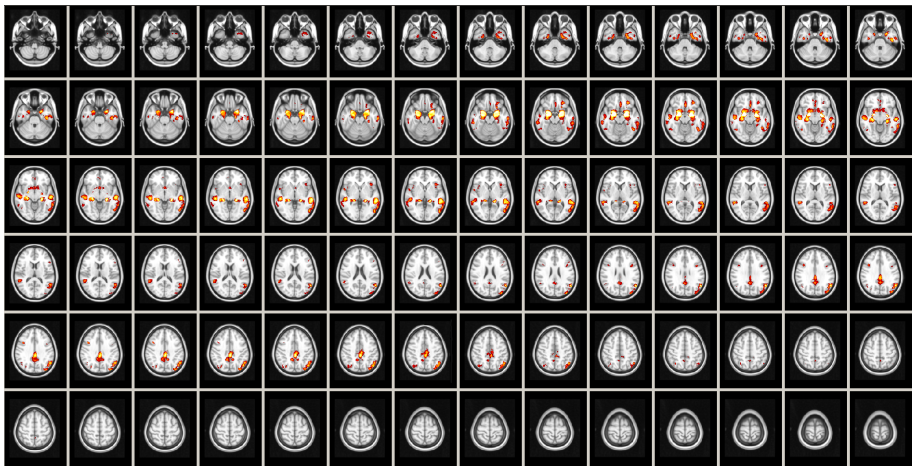

Figure 4.1: F-test statistical map for the polynomial SVR, with significance level (α) filtering at 0.001, minimum cluster size of 100 voxels and transformed into Z-scores for improved visualization.

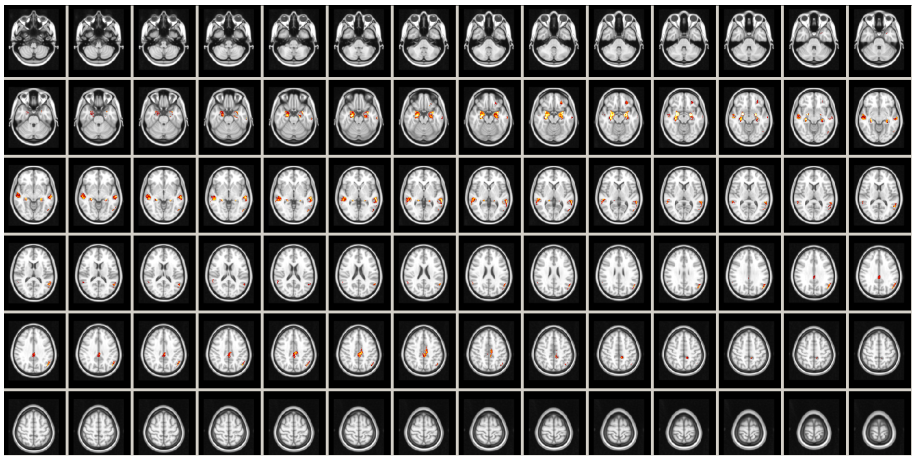

Figure 4.2: F-test statistical map for the Gaussian SVR, with significance level (α) filtering at 0.001, minimum cluster size of 100 voxels and transformed into Z-scores for improved visualization.

In Figure 4.3 we see that the selected voxel is orange, which corresponds to the polynomial SVR model, and in Figure 4.4 we see that the selected voxel is white, which corresponds to the Gaussian SVR model. Again, we see a lot of voxels that are either black (polynomial GLM, the remaining model) or orange, indicating that the polynomial models (GLM and SVR) provide great statistical power in order to find the regions of interest. But we also see this time that the Gaussian SVR has the greatest fitting scores in specific regions, concretely the right hippocampus — as reported by the Harvard-Oxford Subcortical Structural Atlas —, which is known to be one of the most important memory-related structures of the brain. This may indicate that the inherent non-linearity of the atrophy in this region may not be explained only by a third degree polynomial, but by an even more nonlinear function.

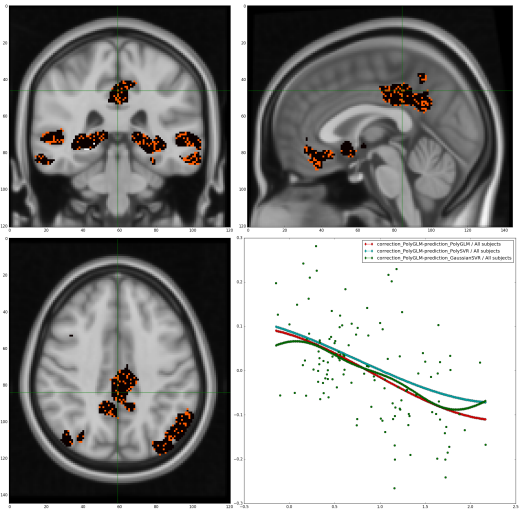

Figure 4.3: *BEST* model of polynomial GLM vs polynomial SVR vs Gaussian SVR with the corresponding curves for a voxel in which the best fitting score belongs to the polynomial SVR model.

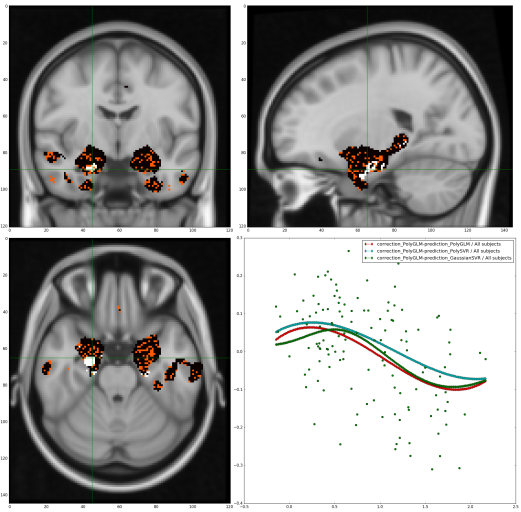

Figure 4.4: *BEST* model of polynomial GLM vs polynomial SVR vs Gaussian SVR with the corresponding curves for a voxel in which the best fitting score belongs to the Gaussian SVR model.

Finally the third experiment was conceived to compare the F-test and the VN-PRSS fit evaluation methods and see which kind of curve patterns they favor. The goal was to find whether the new VN-PRSS evaluation function discovers new regions of accelerated atrophy that the F-test is not capable of. For this experiment to be easily reproducible the polynomial GLM was used as the fitting method.

Figure 4.5 shows the activation map of the F-test while Figure 4.6 shows the activation map of the VN-PRSS fit evaluation method.

We see that the VN-PRSS overlaps with the majority of the activated regions in the F-test, meaning that they capture similar information. Figure 4.7 shows the fitted polynomial GLM curve for a voxel where both the VN-PRSS and the F-test scores are high. Although there is no evidence of the VN-PRSS method providing more information than the F-test, the method still proves itself useful as it can provide similar results as the widely accepted F-test, but it does not require the computation of the degrees of freedom, so it can be applied to any parametric and non-parametric fitter.

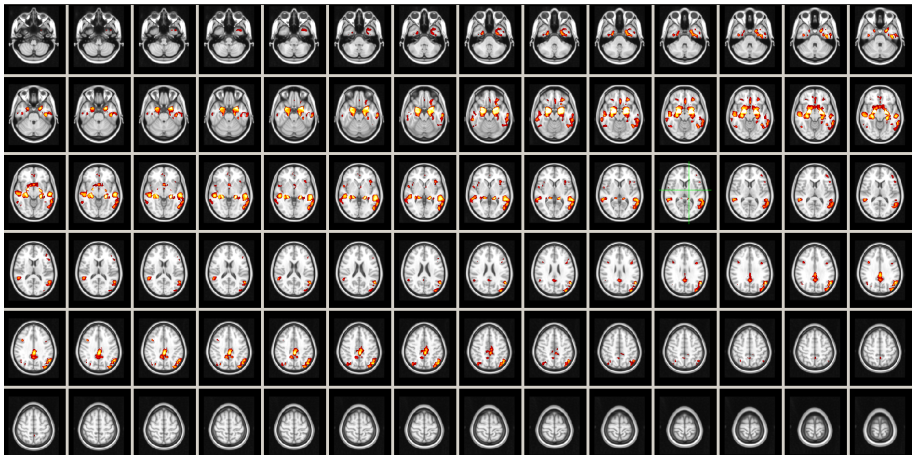

Figure 4.5: F-test statistical map for the polynomial GLM, with significance level (α) filtering at 0.001, minimum cluster size of 100 voxels and transformed into Z-scores for improved visualization.

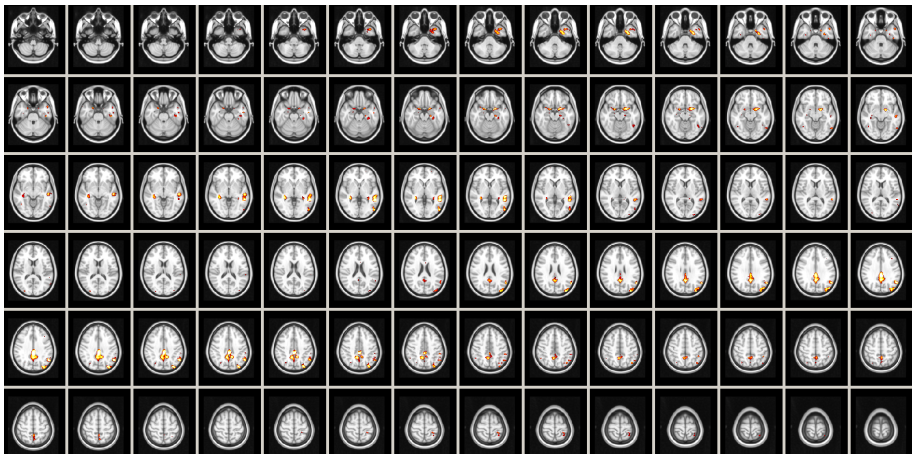

Figure 4.6: VN-PRSS statistical map for the polynomial GLM, with a γ value of 0.0003 and filtering out the values outside the 0.5% percentile

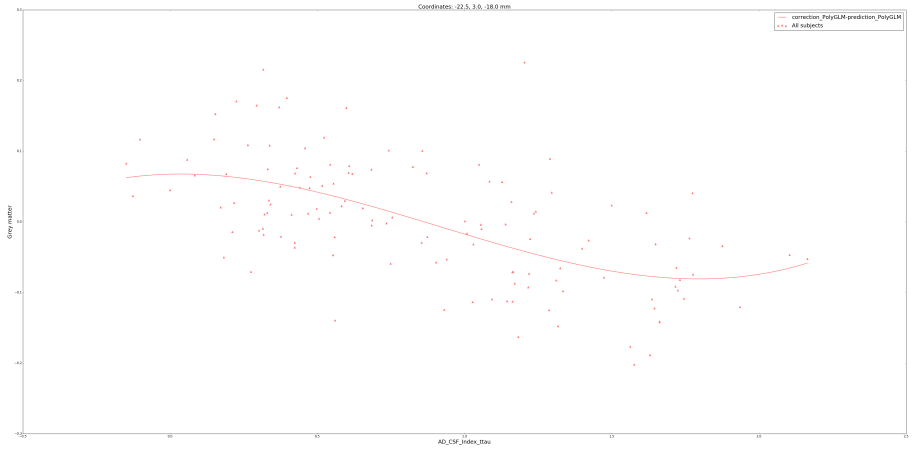

Figure 4.7: Polynomial GLM curve for the voxel in coordinates -22 mm, 3 mm, -18 mm in MNI space.

4.2 Aging atrophy with regards to the APOE4 genotype

The results provided in this section will try to reproduce the ones in [Cacciaglia et al., 2016]. To do so, the same polynomial model used in the article will be used here: the correctors will be *gender*, *years of education* and *total intracranial volume*, all of them fitted by a linear GLM, and the predictor will be *age*, which will be fitted using a polynomial GLM of degree 2 whenever the fitter allows polynomial basis expansion (that is, polynomial GLM, GAM with polynomial smoother and polynomial SVR).

In this problem we want to find results that show differences in the atrophy patterns across normal aging depending on the APOE4 genotype. Hence if we encode the APOE4 genotype as a categorical variable that represents the number of $\epsilon 4$-alleles, we can categorize the subjects of the study as 0, 1 or 2, and then we can use the feature explained in section 3.2.3 in order to fit the observations separately depending on the category and also to perform statistical tests to compare between categories.

A quick inspection of the boxplots of the *Age* variable for each category in Figure A.2 reveals that category 2 does not have the same representation in terms of range, and also that the distribution of this variable is quite different in category 2 from the rest of categories.

The first experiment consisted in fitting the categorically-separated observations with the polynomial GLM using the aforementioned model for the correctors and predictors. The goal of this experiment was to compare the results generated by the toolbox to the ones presented in [Cacciaglia et al., 2016].

We see in Figure 4.8 the original F-test statistical map when comparing $\epsilon 4$-$\epsilon 4$ allele carriers vs the rest. Beside it we have Figure 4.9, which is the F-test statistical map computed by the toolbox when also comparing category 2 vs the rest. Although they are not completely different, it can be easily noticed that the two maps don't match perfectly. A possible explanation of this mismatch is the way in which the category is encoded: in our case, we only encode whether the subject is non-carrier, heterozygote or homozygote for the $\epsilon 4$-allele, while in the paper they encode the APOE4 categorical variable using 5 categories: $\epsilon 2$-$\epsilon 3$, $\epsilon 2$-$\epsilon 4$, $\epsilon 3$-$\epsilon 3$, $\epsilon 3$-$\epsilon 4$ and $\epsilon 4$-$\epsilon 4$ allele carriers.

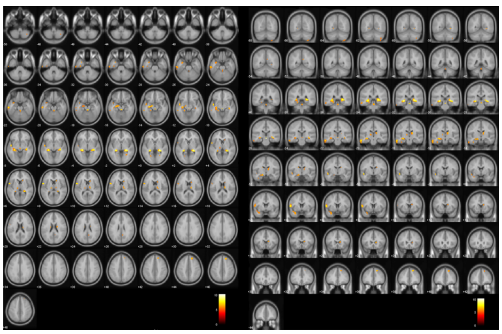

Figure 4.8: F-test statistical map from [Cacciaglia et al., 2016] comparing the $\epsilon 4$-homozygotes versus the rest, with significance level (α) filtering at 0.001 and minimum cluster size of 30 voxels.

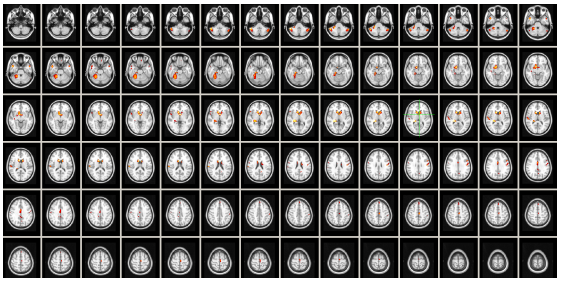

Figure 4.9: F-test statistical map for the polynomial GLM comparing the $\epsilon 4$-homozygotes vs the rest, with significance level (α) filtering at 0.001 and minimum cluster size of 30 voxels.

In Figure 4.10 we see the curves for the significant regions provided in the original paper. Some of these significant regions are derived from a ROI-based analysis — the F-test comparison is performed locally in a region of interest —. Because of that we only plot the

curves for the significant regions at whole-brain level (that is, the posterior hippocampus for both hemispheres of the brain), which can be seen in Figure 4.11 and Figure 4.12.

Here the fitted curves are rather similar despite being fitted with two different methods. We can also see that the right posterior hippocampus is not significant when evaluated with the F-test while the left is, and that is due to the difference in the fitted curves, as the left region presents an abrupt decay around 57 years that the other categories don't present, but the right one follows a similar trend to the other categories.

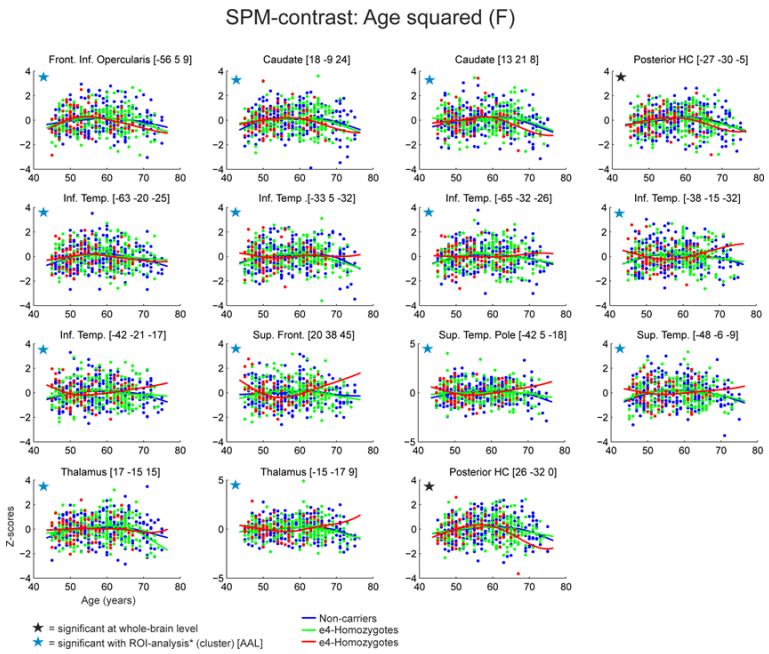

Figure 4.10: Fitted curves using an spline-smoother for the significant regions at ROI level and at whole-brain level provided in [Cacciaglia et al., 2016]

The second and last experiment for this problem consisted in computing the VN-PRSS fit scores for the same comparison as in the first experiment, category 2 vs the rest. The goal in mind when performing this experiment was to see whether this fit evaluation metric uncovered new regions or if it behave similarly to the F-test instead.

To see the differences between the F-test statistical map and the VN-PRSS fit scores map we compared them overlaying one map to the other and putting them in different color modes in FSLView. The resulting colored map (Figure 4.13) shows that the two fit-evaluation methods are almost identical, as the resulting color (purple-ish) is due to the fact that the VN-PRSS map is using a hot colormap with opacity value of 0.5 and the F-test map is using a cool colormap with opacity value of 1. An example of a curve for a voxel where both statistical maps have a high fit score is provided in Figure A.3.

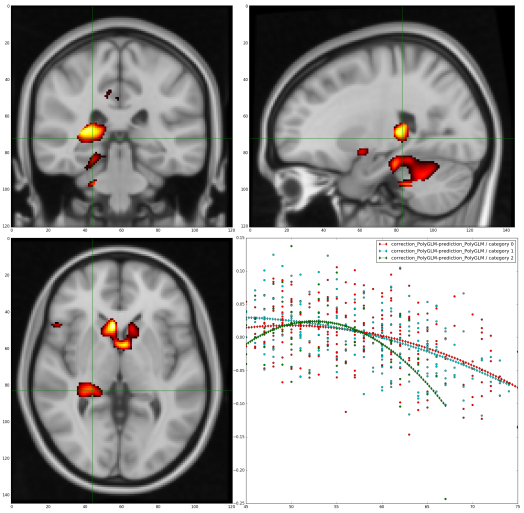

Figure 4.11: F-test activation map for category 2 versus the rest and the corresponding polynomial GLM curve in the voxel with coordinates 26 mm, -32 mm, 0 mm in MNI space, which belongs to the left posterior hippocampus.

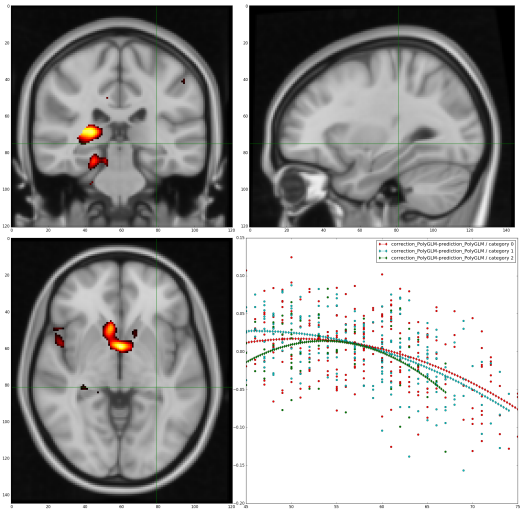

Figure 4.12: F-test activation map for category 2 versus the rest and the corresponding polynomial GLM curve in the voxel with coordinates -27 mm, -30 mm, 5 mm in MNI space, which belongs to the right posterior hippocampus.

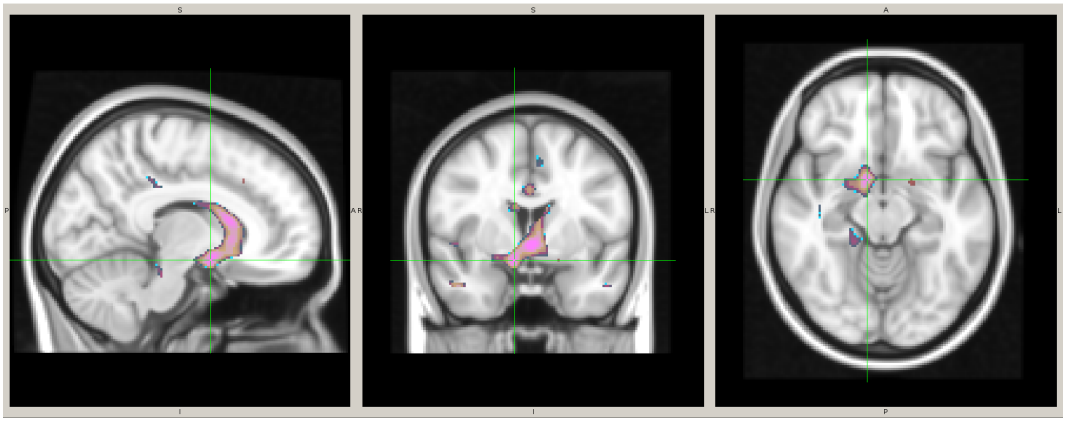

Figure 4.13: Combination of an F-test and a VN-PRSS statistical map for the same model (polynomial GLM) when comparing category 2 vs all. The former is colored in *cool* mode while the latter is colored in *hot* mode and has 0.5 opacity.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis we have presented a software toolbox that allows researchers to study linear and nonlinear patterns of gray-matter variability.

Several fitting methods with their own sub-variants have been introduced and their mathematical formulation has been explained. Also, some of the most widely accepted fit evaluation methods have been presented in this work, as well as others that are more innovative and less traditional. In the context of this work an algorithm to find the hyperparameters of the Support Vector Regression fitters based on a grid search approach has been designed and implemented. The presented algorithm has been designed with a neuroimaging point of view in mind, but it also provides flexibility by the inclusion of several error functions. Finally, visualization tools and methods to compare statistical maps have been included in this work in order to ease the task of inspecting the results and uncovering interesting findings in them.

Having said that, we can conclude that the toolbox has been successfully implemented, as the requirements specified in section 1.2 have been fulfilled and the main objectives of the project in section 1.1 have been achieved. Proof of the successful implementation are the results presented in chapter 4, which are coherent with the ones presented in both articles from FPM and, at the same time, provide added insight of the data due to the diversity of methods that can be applied to it. Furthermore, the toolbox is already fully operational, the code is easily readable as it is all documented and PEP8 compliant, and the CLI interface is also well documented (Appendix F), which makes it user-friendly, even for non-technical users. Another important fact to be mentioned is the increasing interest that the Pasqual Maragall Foundation has put in the publication of the code — as an open-source tool — accompanied by a technical publication.

Finally, although the toolbox incorporates many features that have been proved useful, there is still some development to be done, which is going to be summarized in section 5.2.

5.2 Future development

- **Curve pattern clustering**: One of the most innovative features in the field of neuroimaging that both the research group at GPI and the FPM have conceived is a clustering method that groups voxels depending on the pattern of the fitted curve, i.e., voxels with similar curve trends are clustered together.
- **Extending the data loader for FreeSurfer data**: another interesting feature that has been planned for this toolbox is the extension of the Data loader module to accept FreeSurfer data, so that researchers that are used to FreeSurfer can easily use their data with this toolbox.
- **Implementing more unit tests**: with the implementation of more unit tests (ideally to cover 100% of the code) we would improve the QA of the toolbox.

Bibliography

[Aduriz Berasategi, 2016] Aduriz Berasategi, A. (2016). *Analysis of the dynamics of gray matter reduction in Alzheimer's Disease*. Bachelor's degree thesis, ETSETB.

[Alexopoulos et al., 2011] Alexopoulos, P., Richter-Schmidinger, T., Horn, M., Maus, S., Reichel, M., Sidiropoulos, C., Rhein, C., Lewczuk, P., Doerfler, A., and Kornhuber, J. (2011). Hippocampal volume differences between healthy young apolipoprotein E $\epsilon 2$ and $\epsilon 4$ carriers. *J. Alzheimers Dis.*, 26(2):207–210.

[Bateman et al., 2012] Bateman, R. J., Xiong, C., Benzinger, T. L. S., Fagan, A. M., Goate, A., Fox, N. C., Marcus, D. S., Cairns, N. J., Xie, X., Blazey, T. M., Holtzman, D. M., Santacruz, A., Buckles, V., Oliver, A., Moulder, K., Aisen, P. S., Ghetti, B., Klunk, W. E., McDade, E., Martins, R. N., Masters, C. L., Mayeux, R., Ringman, J. M., Rossor, M. N., Schofield, P. R., Sperling, R. A., Salloway, S., and Morris, J. C. (2012). Clinical and biomarker changes in dominantly inherited alzheimer's disease. *The New England Journal of Medicine*, 367:795–804.

[Cacciaglia et al., 2016] Cacciaglia, R., Domingo Gispert López, J., Falcón, C., and Molinuevo, J. L. (2016). Impact of apoe genetic variant on brain morphology in cognitively healthy individuals with enriched genetic risk for alzheimer's disease. Draft of the manuscript.

[Cherbuin et al., 2007] Cherbuin, N., Leach, L. S., Christensen, H., and Anstey, K. J. (2007). Neuroimaging and APOE genotype: a systematic qualitative review. *Dement Geriatr Cogn Disord*, 24(5):348–362.

[Dinuzzo et al., 2007] Dinuzzo, F., Neve, M., Nicolao, G. D., and Gianazza, U. P. (2007). On the representer theorem and equivalent degrees of freedom of svr. *Journal of Machine Learning Research*, 8:2467–2495.

[Gispert et al., 2015] Gispert, J. D., Rami, L., Sánchez-Benavides, G., Falcon, C., Tucholka, A., Rojas, S., and Molinuevo, J. L. (2015). Nonlinear cerebral atrophy patterns across the alzheimer's disease continuum: impact of apoe4 genotype. *Neurobiology of Aging*, 36(10):2687–2701.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, second edition edition.

[Hastie and Tibshirani, 1990] Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*, volume 43 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC.

[Insel et al., 2015] Insel, P. S., Mattsson, N., Donohue, M. C., Mackin, R. S., Aisen, P. S., Jack, C. R., Shaw, L. M., Trojanowski, J. Q., and Weiner, M. W. (2015). The transitional association between β-amyloid pathology and regional brain atrophy. *Alzheimers Dement*, 11(10):1171–1179.

[Johnson et al., 2012] Johnson, K. A., Fox, N. C., Sperling, R. A., and Klunk, W. E. (2012). Brain imaging in alzheimer disease. *Cold Spring Harbor Perspectives in Medicine*, 2(4):a006213.

[Liu et al., 2013] Liu, C. C., Liu, C. C., Kanekiyo, T., Xu, H., and Bu, G. (2013). Apolipoprotein E and Alzheimer disease: risk, mechanisms and therapy. *Nat Rev Neurol*, 9(2):106–118.

[Molinuevo et al., 2013] Molinuevo, J. L., Gispert, J. D., Dubois, B., Heneka, M., Lleo, A., Engelborghs, S., Pujol, J., de Souza, L. C., Alcolea, D., Jessen, F., Sarazin, M., Lamari, F., Balasa, M., Antonell, A., and Rami, L. (2013). The ad-csf-index discriminates alzheimer's disease patients from healthy controls: a validation study. *Journal of Alzheimer's Disease*, 36(1):67–77.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Poirier et al., 1993] Poirier, J., Davignon, J., Bouthillier, D., Kogan, S., Bertrand, P., and Gauthier, S. (1993). Apolipoprotein E polymorphism and Alzheimer's disease. *Lancet*, 342(8873):697–699.

[Sabuncu et al., 2011] Sabuncu, M. R., Desikan, R. S., Sepulcre, J., Yeo, B. T. T., Liu, H., Schmansky, N. J., Reuter, M., Weiner, M. W., Buckner, R. L., Sperling, R. A., and Fischl, B. (2011). The dynamics of cortical and hippocampal atrophy in alzheimer disease. *Archives of Neurology*, 68(8):1040–1048.

[Smola and Schölkopf, 2004] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

[Wishart et al., 2006] Wishart, H. A., Saykin, A. J., McAllister, T. W., Rabin, L. A., McDonald, B. C., Flashman, L. A., Roth, R. M., Mamourian, A. C., Tsongalis, G. J., and Rhodes, C. H. (2006). Regional brain atrophy in cognitively intact adults with a single APOE epsilon4 allele. *Neurology*, 67(7):1221–1224.

Appendix A

Additional results

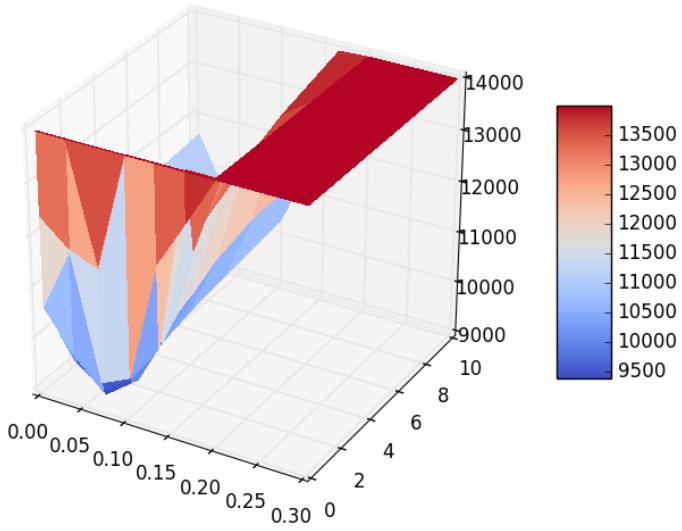

Figure A.1: Error surface for the polynomial SVR when using the ANOVA error function in the *AD-CSF and the Alzheimer's disease continuum* problem.

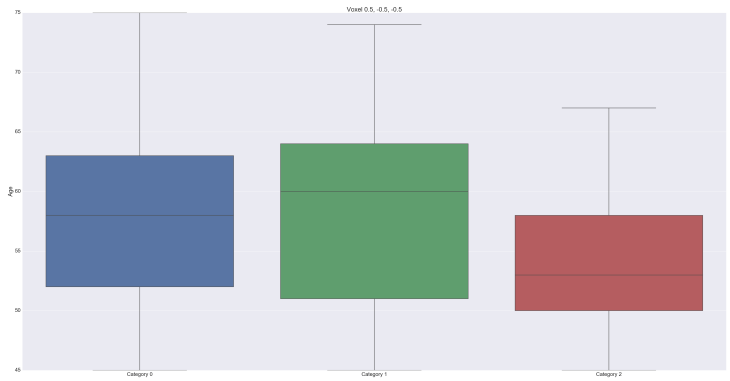

Figure A.2: Categorical boxplot for categories 0, 1 and 2 that presents the minimum and maximum, the median and the first and third quartiles of the *Age* variable for each category of the *Aging atrophy with regards to the APOE4 genotype* problem.

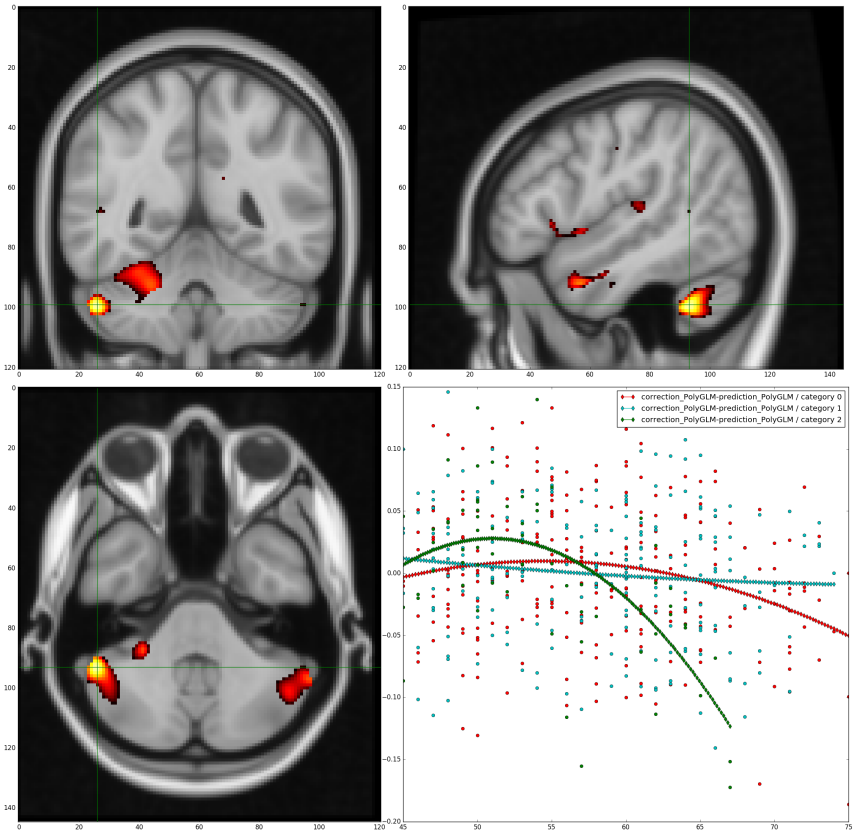

Figure A.3: VN-PRSS statistical map for the polynomial model in *Aging atrophy with regards to the APOE4 genotype* problem and the associated curve in a voxel with a high fit score.

Appendix B

List of Acronyms

FPM Fundació Pasqual Maragall

GPI Grup de Processat d'Imatge

AD Alzheimer's Disease

CSF Cerebrospinal fluid

NIfTI Neuroimaging Informatics Technology Initiative

CLI Command Line Interface

GLM General Linear Model

GAM Generalized Additive Model

SVR Support Vector Regression

VBM Voxel Based Morphometry

RSS Residual Sum of Squares

PRSS Penalized Residual Sum of Squares

Appendix C

Gantt diagram

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**WP1: Project proposal and workplan**	**20 días**	**lun 01/02/16**	**dom 28/02/16**
2		Project description	6 días	lun 01/02/16	dom 07/02/16
3		Project development plan	11 días	lun 08/02/16	dom 21/02/16
4		Document revision and approval	6 días	lun 22/02/16	dom 28/02/16
5		**WP2: Information research and documentation**	**25 días**	**lun 08/02/16**	**dom 13/03/16**
6		Neuroimaging techniques in AD	3 días	lun 08/02/16	mié 10/02/16
7		Neuroimaging data processing	3 días	jue 11/02/16	dom 14/02/16
8		Tools in neuroimaging (SPM)	11 días	lun 15/02/16	dom 28/02/16
9		Toolbox documentation	6 días	lun 22/02/16	dom 28/02/16
10		Non-linearities of AD biomarkers	11 días	lun 29/02/16	dom 13/03/16
11		Nonlinear curve fitting	11 días	lun 08/02/16	dom 21/02/16
12		Statistical tests and evaluation methods	16 días	lun 22/02/16	dom 13/03/16
13		**WP3: Software development**	**65 días**	**lun 29/02/16**	**dom 29/05/16**
14		Neuroimaging structures, formats and methods	6 días	lun 29/02/16	dom 06/03/16
15		Study of nonlinear curve fitting and application to AD	6 días	lun 07/03/16	dom 13/03/16
16		Implementation of nonlinear regression methods	41 días	lun 14/03/16	lun 09/05/16
17		Study of statistical inference and model validation techniq	15 días	lun 18/04/16	vie 06/05/16
18		Implementation of statistical inference and model validatio	16 días	lun 09/05/16	dom 29/05/16
19		**WP4: Critical Review**	**10 días**	**lun 25/04/16**	**dom 08/05/16**
20		Writing about progress to date	6 días	lun 25/04/16	dom 01/05/16
21		Review of the work plan	3 días	lun 02/05/16	mié 04/05/16
22		Document revision and approval	3 días	jue 05/05/16	dom 08/05/16
23		**WP5: Improvements and generalization**	**71 días**	**lun 30/05/16**	**lun 05/09/16**
24		Improvement of grid search by the addition of error functi	3 días	lun 30/05/16	mié 01/06/16
25		Creation of the CLI interface	14 días	jue 02/06/16	mar 21/06/16
26		Parameters computation generalization	7 días	mié 22/06/16	jue 30/06/16
27		Curve visualization tool integration	3 días	vie 01/07/16	mar 05/07/16
28		Generalization of the statistical maps computation	5 días	mié 06/07/16	mar 12/07/16
29		Integration of methods to compare statistical maps	3 días	mié 13/07/16	vie 15/07/16
30		Implementation of statistical distribution visualization tools	5 días	lun 18/07/16	vie 22/07/16
31		Implementation of graphical visualizer	7 días	lun 25/07/16	mar 02/08/16
32		Abstraction of hyperparameter searching functionality	14 días	mié 03/08/16	lun 22/08/16
33		Documentation of the code and the CLI toolbox	10 días	mar 23/08/16	lun 05/09/16
34		**WP6: Testing and generation of results**	**17 días**	**lun 22/08/16**	**mar 13/09/16**
35		Resarch about testing practices in Python	5 días	lun 22/08/16	vie 26/08/16
36		Implementation of unit tests	5 días	lun 29/08/16	vie 02/09/16
37		Generate results	7 días	lun 05/09/16	mar 13/09/16
38		**WP7: Final report**	**12 días**	**mié 14/09/16**	**jue 29/09/16**
39		Write thesis	10 días	mié 14/09/16	mar 27/09/16
40		Thesis revision and approval	2 días	mié 28/09/16	jue 29/09/16
41		**WP7: Oral Presentation**	**11 días**	**vie 30/09/16**	**vie 14/10/16**
42		Draft slides	7 días	vie 30/09/16	lun 10/10/16
43		Presentation rehearsal	3 días	mar 11/10/16	jue 13/10/16
44		Presentation revision and approval	1 día	vie 14/10/16	vie 14/10/16

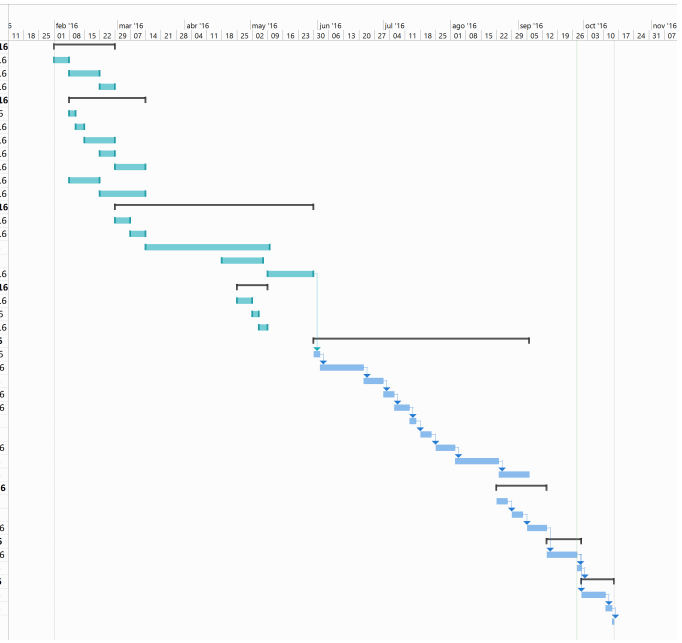

Figure C.1: Gantt diagram of the project.

Appendix D

Budget

Despite being a research project and therefore not involving a service or product to be sold, this section tries to estimate the budget of the project.

The hardware used for this project were the computational resources provided by the GPI, so there has not been any cost in terms of hardware.

The software used for the development of the toolbox and the visualization of results is all open-source, so again there are no costs added.

The only cost that can be accounted for this project is the salary of the members involved in it. Considering the amount of time that each member has put into this project and the *standard* salary for junior engineers, senior engineers and technical advisors, the costs can be summarized as follows:

Role	**Weeks**	**Wage/hour**	**Dedication**	**Total**
Junior engineer	35	10,00 €/h	25 h/week	8.750 €
Junior engineer	24	10,00 €/h	25 h/week	6.000 €
Junior engineer	24	10,00 €/h	15 h/week	3.600 €
Senior engineer	35	20,00 €/h	4 h/week	2.800 €
Technical advisor	-	20,00 €/h	10h	200 €
			Total	21.350 €

Table D.1: Budget of the project

Appendix E

Incidents and modifications

The development of this project hasn't occurred without any incidences or modifications.

As the primary stakeholders of the project were the Pasqual Maragall Foundation, we have had to adapt the requirements and specifications throughout the project in order to provide the best outcome for them. Initially the goal of the project was to develop a software toolbox that allowed researches to study the neurodegenerative patterns across the Alzheimer's disease, and to do so we planned the features that would be useful for that task. By periodically meeting with the FPM we were able to narrow down the initially planned features, so a few of them that were initially considered in the project plan were finally rejected.

The inflection point was around the beginning of June, when the goal of the project was changed for a more ambitious one: to develop a toolbox that would allow researchers to study nonlinear patterns of gray-matter variability and assess their statistical significance for any study, thus not being limited to the Alzheimer's disease continuum problem. This dramatic change of scope was primarily motivated by their need to study a new problem, the *Aging atrophy with regards to the APOE4 genotype* problem.

Regarding the incidents, the most important one was the underestimation of the timing to implement the F-test for one of the newly introduced fitters, the SVR. We conducted an in-depth research over the topic, and some experiments were made in order to provide a robust implementation of the degrees of freedom for the SVR, but it took quite more time than we initially planned.

Appendix F

CLI toolbox documentation

The *Requirements*, *How can I use it?* and *CLI documentation* sections from the README.md file — the file that contains the toolbox documentation in Markdown format — in the Git repository (as of September of 2016 and without images) are included in this appendix. The original file has been converted to PDF in order to be visible in this document.

Nonlinear analysis toolbox for neurodegenerative disease and aging

This toolbox written in Python provides the tools to analyze the linear and non-linear dynamics of gray-matter and study the statistical significance of such dynamics at the voxel level.

Authors

Name	Position / Role
Asier Aduriz Berasategi	Author
Santiago Puch Giner	Author
Adrià Casamitjana Díaz	Contributor
Verónica Vilaplana Besler	Advisor (UPC)
Juan Domingo Gispert	Advisor (PMF)

Institutions

- UPC (Universitat Politècnica de Catalunya)
- PMF (Pasqual Maragall Foundation)

Requirements

In order for this toolbox to properly parse and obtain the data to be processed there are some requirements that should be fullfilled. These requirements are the following:

- **Excel file (.xls)** with all the metadata This file should contain the unique identifier for each subject, an optional categorical value for each subject, and one or several fields with metadata to be used as predictor and/or correctors. The data must be enclosed in the first sheet of the xls book, and this sheet must have the first row as a header with the names identifying the fields to be used. An example of a Excel file with the required format can be found in `tests/mock_data/mock_excel.xls`.
- **Folder** containing all the **NIFTIs** (gzipped or not) This folder must containt one NIFTI file for each subject, and it should be identified with the unique identifier specified in the excel file, with the option to have a study prefix for everyone of them.
- **Template** file in NIFTI format The template into which all the subjects have been registered to compute the VBM (e.g. MNI template)
- **Configuration file (.yaml)** for this study In this configuration file you specify where to find your previous requirements (Excel, data folder and template), where to store the results, the model (predictor and correctors), and other parameters, such as the processing parameters and the configuration parameters for the *GridSearch*.You can find a template of this configuration file in `config/exampleConfig.yaml`.

How can I use it?

The interaction between the user and the software is done through a Command Line Interface (CLI).

As the toolbox is written in Python you must have **python 2.7** previously installed in order to use it (instructions on how to install python can be found here).

First you just have to clone this repository:

```
$ git clone https://santi-puch@bitbucket.org/imatge-upc/neuroimatge.git
$ cd Neuroimatge
```

After that you must install all the dependencies, specified in the **requirements.txt** file:

```
$ pip install -r requirements.txt
```

After all that is done you can execute the scripts using the python executable. This is the pattern that you'll be using to execute the scripts:

```
$ python nln-script.py --options
```

CLI documentation

nln-compute_fitting.py

Computes the fitting parameters for the data provided in the configuration file. This fitting parameters can be computed for all subjects in the study (default behaviour) or you can specify for which categories should the parameters be computed

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
--categories	Yes	Space-separated integers	None	Category or categories (as they are represented in the Excel file) for which the fitting parameters should be computed
--parameters	Yes	Path	None	Path to the txt file within the results directory that contains the user defined parameters to load a pre-configured correction and prediction processor
--prefix	Yes	String	Empty string	Prefix used in the result files

nln-generate_user_parameters.py

Generates user defined parameters for a specific correction and prediction processor so you can use them in compute_fitting.py using the --parameters option

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
--prefix	Yes	String	Empty string	Prefix used in the result files

nln-compute_statistical_maps.py

Computes statistical maps for the fitting results computed by compute_fitting.py. By default uses all computed parameters inside the results folder specified in the configuration file.

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
--method	Yes	mse, r2, fstat, ftest, aic, prss, vnprss	ftest	Method to evaluate the fitting score per voxel and create a statistical map out of these fitting scores
--dirs	Yes	Space-separated paths	All computed parameters within the results directory	Specify one or several directories within the results directory from which the parameters should be loaded
--cluster_size	Yes	Integer	100	Value of the minimum cluster size (in voxels) that should survive after thresholding
--p_thresholds	Yes	Space-separated floats	0.01 0.005 0.001	One or more values representing the maximum acceptable p-value, so that all voxels with greater p-value are put to the default value
--gamma	Yes	Float	5e-3	Value of the percentile used to determine the upper threshold for PRSS and vnPRSS methods
--gm_threshold	Yes	Float	0.1	Mean grey-matter lower threshold
--labels	Yes	Boolean	True	Produce a map that has one label per cluster

nln-compare_statistical_maps.py

Compares statistical maps generated by compute_statistical_maps.py using four possible techniques: RGB map, best-fitting map, absolute difference or squared error. You must specify the specific maps to compare and ensure that they are comparable (Z-score map vs Z-score map, p-value map vs p-value map, etc.)

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
files	No	Space-separated paths	-	Specify two or more files within the results directory to be compared
--method	Yes	best, rgb, absdiff, se	best	Method to compare the fitting score per voxel and create a new statistical map out of this comparison
--name	Yes	String	Empty string	Name to be prepended to the output file

nln-search_hyperparameters.py

Finds the hyper parameters of the PolySVR or GaussianSVR using a grid search approach and using several error functions

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
--parameters	Yes	Path	None	Path to the txt file within the results directory that contains the user defined parameters to load a pre-configured correction and prediction processor
--categories	Yes	Space-separated integers	None	Category or categories (as they are represented in the Excel file) for which the hyperparameters should be found
--prefix	Yes	String	Empty string	Prefix used in the result files
--error	Yes	mse, anova, Cp	anova	Error function to be minimized in order to find the optimal hyperparameters
--iterations, -i	Yes	Integer	5	The number of iterations to perform
--voxels, -v	Yes	Integer	100	The number of voxels to be used to compute the error and therefore find the optimal hyperparameters. In general, more voxels used may imply better generalization, but also more computation time and use of resources
--voxel-offset	Yes	Integer	10	Number of voxels that will not be taken into account in all directions, both at the beginning and at the end. That is, for a voxel offset of v, and volumes with dimensions (x_dim, y_dim, z_dim), only the following voxels will be taken into account: [v:x_dim-v, v:y_dim-v, v:z_dim-v]

nln-show_curves.py

Shows the curves for the fitting results computed by compute_fitting.py. By default shows all computed parameters inside the results folder specified in the configuration file

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
--dirs	Yes	Space-separated paths	All computed parameters within the results directory	Specify one or several directories within the results directory from which the parameters should be loaded
--compare	Yes	Boolean	True	Plots the curves in the same figure so that you are able to compare the different curves. The program does not recognize whether the data has been corrected with the same fitter or not, so you must ensure this to have coherent results

nln-show_visualizer.py

Shows the graphical visualizer to display a statistical map and the curves for the selected voxel

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
map	No	Path	-	Path relative to the output directory specified in the configuration file to the statistical map to be loaded
dirs	No	Space-separated paths	-	Specify one or more directories within the results directory specified in the configuration file from which the fitting parameters should be loaded
--colormap	Yes	hot, rainbow	hot	Color map used to paint the statistical maps' values. By default it is 'hot', useful for statistical based measures (F-stat, p-values, Z-scores, etc.), but you can use 'rainbow' for labeled maps
--n-points	Yes	Integer	100	Number of points used to plot the curves. More points means a smoother curve but requires more computational resources

nln-show_data_distribution.py

Shows the data distribution of the observations, the predictors, the correctors and the residuals

Parameter name	Optional	Possible value/s	Default value	Description
configuration_file	No	Path	-	YAML configuration file for the study, as specified in the requirements section
plot	No	univariate_density, bivariate_density, boxplot, categorical_boxplot	-	Type of plot to be used. For the categorical_boxplot it is assumed that the dirs specified belong to different categories of the data. Otherwise, only the last data retrieved from a specific category will be taken into account
--dirs	Yes	Space-separated paths	All computed parameters within the results directory	Specify one or several directories within the results directory from which the parameters should be loaded

---

You can get help about the required parameters using the `--help` option, which is supported by all the scripts.

For example, if you want to know how to execute the *nln-show_visualizer.py* script, you can use:

```
$ python nln-show_visualizer.py --help
```