Random Regression model for genetic applications

Submitted by:
Marco Biazzo Mathematical Engineering

Supervised by: Prof.ssa Sangalli Laura, Prof. Pigoli Davide

School of Industrial and Information Engineering
Politecnico di Milano

Acknowledgments

This work is the result of a combination of efforts, first of all I want to thank professor Pigoli for his commitment and availability, and professor Laura Sangalli for the support. I want to thank my parents, for the unconditional love and encouragement I have received during these years, my friends and colleagues for the company, and all the people that gave me something valuable during this journey.

Contents

List of Figures

Abstract

In recent years functional data analysis (FDA) have received a great deal of attention, many useful theories and applications have been reported. In a functional setting each data unit is a curve and it is assumed to be a realization of a random process. Quantitative genetics happens to find the functional approach extremely useful for the analysis of functions such as growth and growth rate over time. The main objective of any genetic analysis is the estimation of the covariance function of the generator process and its main eigenfunctions (their role will be explained in detail in chapter 1), these elements can give insights into the dynamics of the underlying process and suggest patterns for selection. More specifically the analysis of these quantities is strictly related to variation and covariation among relatives and to the dynamics of evolutionary change in a particular trait across generations. Some `R` packages (`pedigreemm`, Vazquez et al., 2010 [32]) proposed a fitting strategy for multi-variate and simple genetic models. The first part of this work is dedicated to the development of an algorithmic routine for the fitting and covariance estimation of general functional models, providing a customizable fitting procedure able to deal with several kind of processes. This routine will allow the estimation of the suitable parameters of a model in a general functional setting as opposed to the previous limited multivariate approaches.

Some different techniques have been proposed for the estimation of the quantities of interest, however, the uncertainty around these estimates is generally not examined. The second part of the work proposes an inference criterion for the estimated quantities (covariance function and eigenfunctions) based on a parametric bootstrapping strategy. We will build simultaneous confidence bands, which will prove to be the right choice in a functional setting, and assess the reliability of the estimates generated in the first part of the work. The fitting procedure and inference criterion will be tested first on simulated data, at this stage the inference strategy will prove to be effective for the assessment of uncertainty. Then we will analyze growth trajectories in larval flour beetle *Triboleum Castaneum*, estimating evolutionary constraints and predicting the generational response under different selection regimes.

Chapter 1

Introduction

The study of patterns of growth have always been of special interest to evolutionary biologists, mostly because of their strong correlation with adult fitness (Cheverud et al., 1983 [10]; Mousseau & Roff, 1987 [26]). More specifically, taking into consideration a certain species, growth trajectories (body size over time) are often strongly correlated to grow rate, adult size, pre-adulthood viability and many other factors (Santos et al., 1992 [30]). These quantities have a strong impact on survivorship and mating success (Sokolovska et al., 2000 [31]). For this reason the study of growth curves of a population allows to have great insights into the selection process. Quantities like size or growth rate are more generally referred to as traits, and since they are functions of some indipendent variables they are known as function-valued traits (Kingsolver et al., 2001 [17]). These curves are assumed to be realizations of an underlying stochastic process. In figure 1.1 are displayed two growth curves from the dataset that will be used in the last part of the work regarding the flour beetle *Triboleum Castaneum.*

In a multivariate analysis it is possible, given a dataset, to estimate the covariance matrix of the recorded data; in analogy to the multivariate case, to a functional dataset is associated a covariance *function* which represents the covariance between two records measured at any point in time. Notice that in this work we will refer to the indipendent variable as *time* since in *Triboleum Castaneum* the growth curves are functions of time. In a genetic framework the covariance function is never considered as a whole, it is

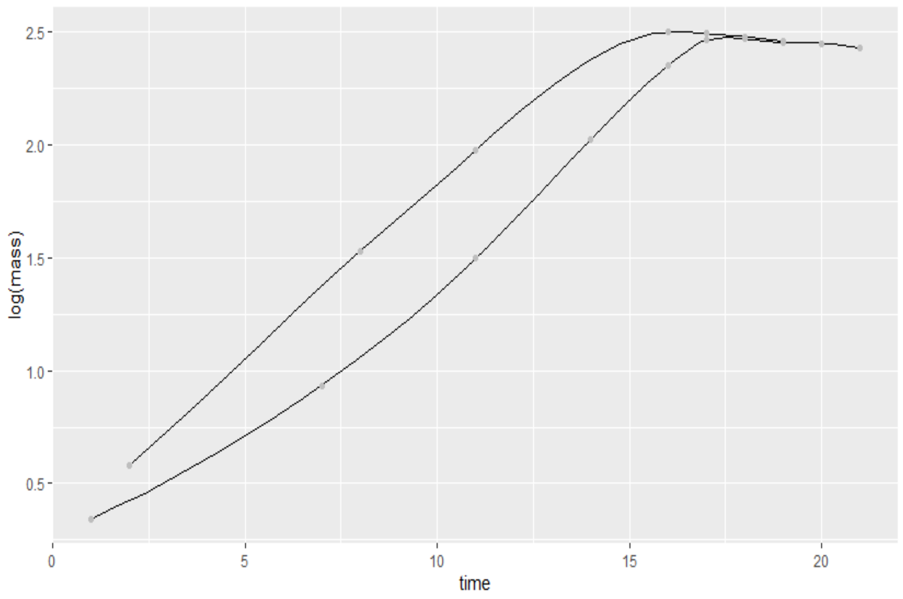

Figure 1.1: Growth curves and sampling points of *Triboleum Castaneum*

indeed modelled as a sum of a genetic part and an environmental one. The genetic part represents how much variation of a trait can be attributed to genetic variability between individuals in the population as opposed to environmental one, which stands for variation coming from non-genetic causes. The ratio between genetic variation and total variation is referred to as *heritability.* The genetic covariance function G is of crucial importance in our analysis, it is indeed possible, knowing G, to predict the mean phenotypical variation of a trait during evolution. More specifically, knowing the covariance function is possible to predict the change in mean of a population's trait from one generation to the following one. This function is also useful for the identification of the so called *genetic constraints*, which are, roughly speaking, limitations or restrictions in the course of adaptive evolution. These evolutionary constraints are tightly related with the eigenfunctions of G for reasons that will be described in detail in chapter 2.
Growth curves are the most commonly analyzed kind of curves, they have been modeled as a functional trait in many occasions. Agricultural studies have used a functional approach for the estimation of the covariance function (Kirkpatrick et al., 1994 [21]; Albuquerque & Meyer, 2001 [1]), several functional models have been employed for the study of growth and strength of selection on body size (mice: (Kirkpatrick et al., 1990 [20]; Kirkpatrick & Lofsvold, 1992 [19]); salamanders: (Ragland & Carter, 2004 [28]); birds: (Bjorklund, 1997 [7]; Badyaev & Martin, 2000 [2])) Our dataset on *Triboleum*

Castaneum (TC) has been considered in various occasions for genetic and development studies for being a worldwide pest for food grains; it showed strong heritability in body mass at different stages of development (Okada & Hardin, 1967 [27]; Conner & Via, 1992 [11]; Campo & Rodriguez, 1986 [9]). In our work we will consider growth curves, i.e. size as function of time. It is intuitive indeed how the size of an animal, for example, can be positively correlated with fitness measures like mating probability, mating success and fecundity. Later on we will estimate the G for TC to find possible genetic constraints for the growth curve and to understand how the mean phenotypic trait changes during selection.

But let us now outline the goals of this work:

- Find an estimation procedure of the G function in a genetic setting based on the random regression model and provide a flexible algorithmic `R` routine for the fit. Indeed at the moment there are no available packages for the fit of a RR model on a functional dataset. This work will provide a function able to deal with generating processes having diverse characteristics. The effectiveness of the procedure will be assessed first in multivariate settings, comparing it with existing packages, and then tested on functional environments through simulations.

- Construct a method to find confidence bands for the covariance G and its eigenfunctions; in quantitative genetic the estimation of the covariance function G is almost never matched with an analysis of its distribution, so the uncertainty around the estimates is not examined. This work proposes a parametric bootstrapping strategy for the assessment of the reliability of the functional estimates found in the previous point. The inference criterion will be also tested in a simulated setting proving to be a satisfactory option for the evaluation of uncertainty.

Vazquez et al., 2010 [32] has proposed a routine for the fitting of linear mixed model in a genetic setting through the `R` package `pedigreemm`, however, this library is only able to fit random intercept models, limiting its use to multivariate applications. The strategy in Vazquez et al., 2010 [32] will be extended to covariate models to handle a functional setting. Secondly we will provide a procedure to make inference on estimates of G and its eigenfunctions building two different kinds of confidence bands, a pointwise and a simultaneous one based on a parametric bootstrap strategy. We will show how the simultaneous bands are suitable for our functional analysis, testing them first in a simulated environment and then analyzing the case study of *Triboleum Castaneum*.

Chapter 2

Motivations and Problem Statement

This chapter is dedicated to the description of the necessary background notions to understand the goal and the methods of the work. More precisely in the following we will:

- Section 2.1: Explain why a functional approach is a valuable one and set the useful components of our genetic model. We will also explain what are the important quantities to be estimated for genetic purposes and why they are deemed to be useful
- Section 2.2: Describe in detail the model used in this work, anticipate the estimation procedure of the suitable quantities and outline the advantages of the random regression model
- Section 2.3: We will describe in detail the estimation procedure of the quantities of interest, an insight into it is indeed necessary to understand some considerations on the implementation step
- Section 2.4 and Section 2.5: Here we will outline a strategy to fit our genetic model via the `R` package `lme4`, comment the limitations of this strategy and describe how to overcome them

- Section 2.6 is dedicated to the description of the dataset that will be analyzed in the case study and used in this chapter as an example

2.1 Genetic Framework

As anticipated in the introduction, we will be working with a set of functional data called functional-valued (FV) traits. As an example of a functional dataset we show *Triboleum Castaneum* (which will be use in chapter 4), in figure 2.1. The dataset contains growth curve as functions of time, in days, which are the days post hatch of the flour beetle. A detailed description of the dataset is present later in the chapter in section 2.6.

There are many advantages of using a functional approach: first, comparing a functional approach to a multivariate one, the first keeps track of the ordering of measurements (trait values) and the proximity of these measurements in time. Secondly this kind of analysis accounts for traits value that have not been directly sampled, this is done in automatic, by interpolating the observed data (Kirkpatrick & Heckman, 1989 [18]). Third the visualization and conceptualization of a single data unit as a function is way easier than a vector of points, allowing a better understanding and deeper insights. Finally the functional approach allows flexibility in the measurements on the data, since allows to take samples in different times for each record.

Let us now set the model for our functional analysis. Consider a certain time interval $\mathcal{T} = [0, T]$ and a sample space Ω, each function is considered to be a random realization of a real valued process:

$$X(\omega, t) : \Omega \times \mathcal{T} \to \mathbb{R}$$

Given this process it follows the definition of its parameters:

- Covariance function $P : \mathcal{T}^2 \to \mathbb{R}$ s.t. $P(t, s) := Cov(X(t), X(s))$, which is the covariance between two records of the process measured at any time in $[0, T]$. The capital P stands for phenotypical

- Mean function $\mu(t) : \mathcal{T} \to \mathbb{R}$ s.t. $\mu(t) := \mathbb{E}[X(t)]$, μ is called mean process

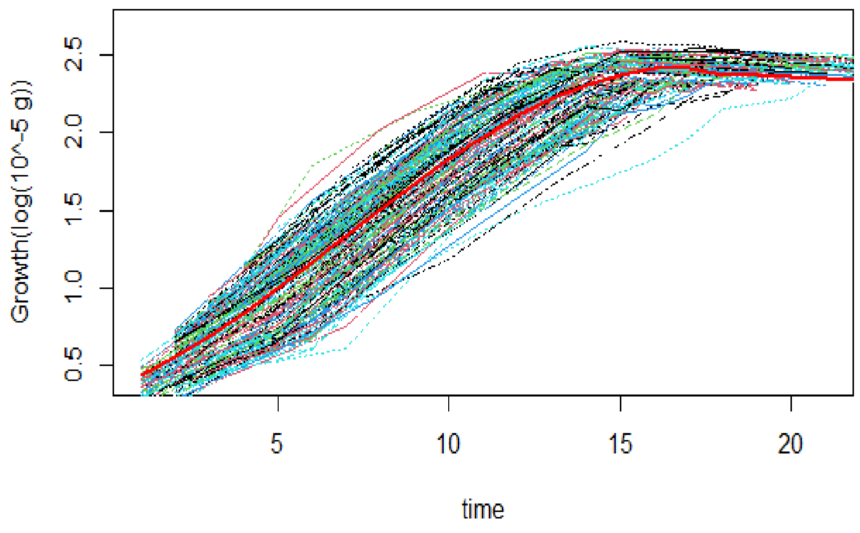

FIGURE 2.1: Growth curves and mean curve of *Triboleum Castaneum*

The goal of the analysis is to explore the evolution of the process under different selection regimes, in other words to examine how the population characteristics change during a generation of selection. For doing this it is of crucial importance the estimation of the covariance function P, more specifically the estimation of its genetic component G. Let us now describe in detail the role of G in quantitative genetics and how it turns out to be so important. As anticipated in the introduction, the total phenotypical variance of a population can be split in a component caused by the genetic variation present in the population, and another one depending on environmental or random chance. Genetic variation is estimated comparing phenotypical variation in related individuals. A tool to model genetic relationship is the additive relationship matrix $\boldsymbol{A}$; let us now briefly elaborate on its structure. $\boldsymbol{A}$ describes the genetic relations between the elements of the dataset, it is a positive definite matrix and each term $\boldsymbol{A}_{ij}$ represents how closely related subjects i and j are (for example a coefficient of 0.5 means that the two subjects either have a sire or dam in common). Since the majority of the terms of $\boldsymbol{A}$ are zero the matrix is a sparse one.

An example of $\boldsymbol{A}$ is the following:

$$\begin{bmatrix} 1 & 0.5 & 0.125 & 0 & .. \\ 0 & 1 & 0.25 & 0 & .. \\ 0 & 0.5 & 1 & 0.25 & .. \\ . & . & . & . & .. \end{bmatrix}$$

$\boldsymbol{A}$ is very useful in modeling the correlation between the random terms correspondent to related individuals. In the final stages of the definition of the model it will be completely clear how this matrix turns out to be useful.

Let us now shed a light on the importance of G in a genetic framework. To do this it is necessary to introduce another useful quantity: the selection gradient β.

The selection gradient is a function describing the relation between a certain character's trait and a species' relative fitness; for example high body size, high growth rate, may improve or reduce fitness. The gradient is useful in evolutionary biology to understand the evolution of a trait over time. Traits that improve an organism's ability to survive tend to increase in frequency in the population through a process called natural selection.

A simple example can be shown: consider a certain trait x and a measure of fitness f, now assume the following holds:

$$f = a + \beta x$$

So we have a precise (naive) expression of fitness as function of the trait x, the derivative of f w.r.t. the trait is the selection gradient β. The relation between fitness and the gradient can of course be more complex than the latter, its estimation goes however beyond the scope of the work. From now on we will call β our selection gradient and consider it as a function of t. It is crucial now to justify the definition of the selection gradient and say why the covariance function is of genetic importance.

Let us introduce the following equation:

$$\Delta\mu(t) = \int_0^T G(t,s)\beta(s)\, ds \tag{2.1}$$

The phenotypical mean change in a generation can be predicted by applying the formula

2.1, $\Delta\mu(t)$ is the mean phenotypical change, G is the genetic covariance function and β is the selection gradient.

The operator:

$$y(t) \rightarrow \int_0^T G(t,s)y(s)\,ds$$

is a functional operator with the genetic covariance function G as kernel (hereafter we will write G meaning the covariance function and the operator interchangeably). Applying G to a given selection gradient, the result is the change in mean phenotype between the next generation of organisms and the current one (Kirkpatrick et al., 1989 [18]).

Notice how $\Delta\mu(t)$ can be on average maximized across the range of independent index t when $\beta(s)$ is taken to be the eigenfunction associated with the largest eigenvalue of the genetic operator G. A selection gradient corresponding to the first eigenfunction will then result in the greatest phenotipic response. On the contrary a gradient collinear to eigenfunctions with small eigenvalues will result in a small phenotypical change, these direction are what are referred to as *evolutionary constraints*.

The genetic covariance function G, and its ratio with the phenotypical covariance P can also be source of information. The ratio between the G and P represents how much of the phenotipical variance is due to genetics and is commonly referred to as *heritability*.

Sometimes the analysis of the derivative of growth rates can be of genetic importance, this can be performed with the same strategy described before, but an important remark has to be made: derivatives as functional data, and the results of the their analysis, are strictly dependent on the kind of smoothing involved. We will talk briefly about this problem without getting too much into it since it goes beyond the purpose of this work.

2.2 Genetic random effect model

Let us now introduce the genetic model. Let $\boldsymbol{C}^G$ be a genetic covariance matrix of dimension $k_G \in \mathbb{N}$ and $\boldsymbol{C}^E$ an environmental covariance matrix of dimension $k_E \in \mathbb{N}$. We assume each realization of the random process to be an $\mathbb{L}^2([0,T])$ function and let us choose a basis $(\phi(t)_m)_{m\in\mathbb{N}}$ for this functional space, i.e. a countable dense set in the space. After having chosen a basis it is necessary to cut down its number of elements, we will consider two truncated basis made of the first k_G and k_E elements. At this point we are ready to model the stochastic process defined above in the following way:

$$X(t) = \mu(t) + \sum_{m=0}^{\infty} \alpha_m \phi_m(t) + \sum_{m=0}^{\infty} \gamma_m \phi_m(t)$$

And since we will not work with infinite series we perform the following approximation:

$$X(t) \cong \mu(t) + \sum_{m=0}^{k_G-1} \alpha_m \phi_m(t) + \sum_{m=0}^{k_E-1} \gamma_m \phi_m(t) \tag{2.2}$$

Where:

- $\alpha_m = \alpha_m(\omega) : \Omega \to \mathbb{R} \ \ \forall m$
- $\gamma_m = \gamma_m(\omega) : \Omega \to \mathbb{R} \ \ \forall m$

And setting the following:

$$\boldsymbol{\alpha}^* = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ .. \\ \alpha_{k_G-1} \end{bmatrix}, \boldsymbol{\gamma}^* = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ .. \\ \gamma_{k_E-1} \end{bmatrix}$$

we can complete the definition of the model, setting the distribution of the random parameters:

$$\boldsymbol{\alpha}^* \sim N(\mathbf{0}, \boldsymbol{C}^G)$$

$$\boldsymbol{\gamma}^* \sim N(\mathbf{0}, \boldsymbol{C}^E)$$

Notice that the approximately equal in equation 2.2 is necessary since we are not using the complete basis but only a truncation. We will call $F(t)$, the non-random term, fixed effect term, while the remaining parts will be called mixed/random effect terms.

From now on we will set $k_G = k_E = k$ and costruct a fitting procedure for the data based on equation 2.2.

Let us now introduce a further step to model measurement noise through the following equation:

$$Y_{ij} = F(t_{ij}) + \sum_{m=0}^{k-1} \alpha_{im}\phi_m(t_{ij}) + \sum_{m=0}^{k-1} \gamma_{im}\phi_m(t_{ij}) + \epsilon_{ij} \tag{2.3}$$

Where the quantities used in the model are the following:

- i : subject-specific index
- $t_{ij} \in \mathcal{T}$: measurement j regarding the subject i (measurements are ordered w.r.t. j)
- N : number of subjects
- $\boldsymbol{\phi}(t) = [\phi_0(t), \phi_1(t), ..., \phi_{k-1}(t)]^T$: the first k elements of the functional basis
- l_i : number of measurements for the subject i
- Y_{ij} : measurement j relative to the subject i
- $\boldsymbol{Y}_i = [Y_{i1}, Y_{i2}, .., Y_{il_i}]$: values sampled from subject i
- $\boldsymbol{\alpha}_i^*$: the same α^* of before correspondent to a particular subject (the same holds for $\boldsymbol{\gamma}_i^*$)
- $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1^*, \boldsymbol{\alpha}_2^*, .., \boldsymbol{\alpha}_N^*]^T$
- $\boldsymbol{\beta} = [\beta_0, \beta_1, .., \beta_{k-1}]^T$: the fixed effect predictor
- ϵ_{ij} : gaussian measurement error
- $\boldsymbol{A}$: $N \times N$ additive relationship matrix

The fixed effect F models the mean process $\mu(t)$, usually estimated with the same basis chosen for the mixed effect.

Equation 2.3 can be also written in the form:

$$Y_{ij} = \boldsymbol{\beta}^T\boldsymbol{\phi}(t_{ij}) + (\boldsymbol{\alpha}_i^*)^T\boldsymbol{\phi}(t_{ij}) + (\boldsymbol{\gamma}_i^*)^T\boldsymbol{\phi}(t_{ij}) + \epsilon_{ij} \tag{2.4}$$

Where the fixed term is modeled as a linear combination of elements of the basis (the order is the same as the random effect terms). Once reached the equation in 2.4 we need to take the last step forward to set up a notation in matrix form. Calling n the total number of observations we can define $\boldsymbol{J}$, an $n \times N$ matrix of indicator columns for the subjects, a matrix in the form:

$$\boldsymbol{J} := \begin{bmatrix} 1 & 0 & 0 & . \\ 1 & 0 & 0 & . \\ 0 & 1 & 0 & . \\ 0 & 1 & 0 & . \\ . & . & . & . \end{bmatrix}$$

Each element $\boldsymbol{J}_{hi}$ is equal to 1 if the observation h in the dataset belongs to the i-th subject.

And using the basis vector evaluated in the sampling points we define:

$$\boldsymbol{X} := \begin{bmatrix} \boldsymbol{\phi}(t_{11}) \\ \boldsymbol{\phi}(t_{12}) \\ . \\ . \\ \boldsymbol{\phi}(t_{Nl_N}) \end{bmatrix}$$

Where $\boldsymbol{X}$ is a $n \times k$ matrix. At last the random effect matrix $\boldsymbol{Z}$ undergoes the following definition:

$$\boldsymbol{Z} := (\boldsymbol{J}^T * \boldsymbol{X}^T)^T = \begin{bmatrix} \boldsymbol{J}_1^T \otimes \boldsymbol{X}_1^T \\ \boldsymbol{J}_2^T \otimes \boldsymbol{X}_2^T \\ . \\ . \\ \boldsymbol{J}_N^T \otimes \boldsymbol{X}_N^T \end{bmatrix}$$

In this formula $*$ is the Katri-Rao product (Khatri et al., 1968 [16]), $\otimes$ is the Kronecker product and $\boldsymbol{J}_j^T$ and $\boldsymbol{X}_j^T$ are respectively the j-th rows of $\boldsymbol{J}$ and $\boldsymbol{X}$.

A matrix $\boldsymbol{Z}$ has to be built for each random effect term, in our case two of them. Calling $\boldsymbol{Z}^G$ and $\boldsymbol{Z}^E$ the random effect matrices relative to, respectively, the genetic and environmental term we can state the final model:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}^E\boldsymbol{\alpha} + \boldsymbol{Z}^G\boldsymbol{\gamma} + \boldsymbol{\epsilon} \tag{2.5}$$

This is called random regression model (RR), the random terms indeed have the following distributions:

$$\boldsymbol{\alpha} \sim N(\boldsymbol{0}, \boldsymbol{A} \otimes \boldsymbol{C}^G)$$

$$\boldsymbol{\gamma} \sim N(\boldsymbol{0}, \boldsymbol{I}_N \otimes \boldsymbol{C}^E)$$

$$\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2_{res} * \boldsymbol{I}_n)$$

Needs to be specified that $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are uncorrelated. Notice how the additive relationship matrix models is useful to model the correlation between related subjects' random coefficients.

Covariance Estimation

Keeping in mind the goal of our genetic analysis it is necessary now to explain how to estimate the covariance matrix of the process $\boldsymbol{X}$ and why it is so convenient to use a random regression model (Meyer, 1998 [23]). Consider the formula in 2.3, the goal is to have an expression of the covariance between any two records taken at $(t_{ij}, t_{ij'})$. With few computational steps it is possible to derive the following equation:

$$\begin{aligned} Cov(\mathbf{Y}_{ij}, \mathbf{Y}_{ij'}) = &\sum_{m=0}^{k-1}\sum_{l=0}^{k-1} \phi_m(t_{ij})\phi_l(t_{ij'})Cov(\alpha_{im}, \alpha_{il}) \\ &+ \sum_{m=0}^{k-1}\sum_{l=0}^{k-1} \phi_m(t_{ij})\phi_l(t_{ij'})Cov(\gamma_{im}, \gamma_{il}) + Cov(\epsilon_{ij}, \epsilon_{ij'}) \end{aligned} \tag{2.6}$$

Given this expression it is straighforward to write down the general covariance function once known the covariances between the random coefficients.

At this point the estimation of G and P reduces to the estimation of its coefficients over the chosen basis. The idea is to set a random effect model as in equation 2.5 with the `R` package `lme4`, fit this model and extract an estimate for the covariance of the random effects coefficients.

Computational requirements for this RR are reduced to the order of fit of the covariance function. Other strategies have been explored before with higher computational costs: Kirkpatrick et al., 1994 [21] estimated covariance functions using a generalized least square approach, Meyer and Hill, 1997 [24] proposed a different estimation procedure via REML, this, however, required a mixed model matrix with dimension proportional to the number of sampling points, which is very limiting. Our approach instead aims at estimating a random effect matrix of dimension proportional to the product (#*random coefficients*)∗(# *levels*), where, in our case, the levels are the subjects.

Now that we have derived the final form of the model we can discuss some of its characteristics and complications. One of the most common packages for fitting mixed effect model in `R` is `lme4`, which provides an interface similar to the one of linear models through the command `lmer`. The fitting is achieved via an iterative routine involving two main steps (which will be described in detail in section 2.3 (Bates et al., 2014 [4])):

- Penalized least squares (PLS), for the derivation of a temporary random effect and fixed effect vector estimate
- Restricted maximum likelihood (REML), in this step the deviance is profiled and minimized through a customizable optimizer

In linear mixed models REML or maximum likelihood are common, indeed, under gaussian assumptions, the marginal likelihood of the data has a closed form and maximum likelihood or REML estimation can be performed conveniently. In the next section we will discuss about the parameter estimation performed in package `lme4` which will be the procedure underlying our strategy.

2.3 Random effect parameter estimation

In this section we will outline the estimation procedure performed in Bates et al. [4](`lme4`), using a more compact notation than the one discussed in the genetic model. It is important to know the layers underlying the command `lmer` for many reasons:

- In chapter 3 we will construct a fitting procedure based on `lme4`'s routine. It will be important to determine if a model is singular via several convergence checks involving some of the quantities defined in this section
- An understanding of the penalized least square procedure (2.3.1) is important to familiarize with some of the constraints we will impose to the fitting routine

The model to be estimated in this section will be the following:

$$(\boldsymbol{Y}|\mathcal{B}=b) \sim \mathcal{N}(\boldsymbol{X\beta}+\boldsymbol{Zb}, \sigma^2\mathbf{W}^{-1}) \tag{2.7}$$

$$\mathcal{B} \sim \mathcal{N}(0, \boldsymbol{\Sigma}) \tag{2.8}$$

$\boldsymbol{\Sigma}$ will be then expressed as a function of the relative covariance factor $\Lambda_{\boldsymbol{\theta}}$ which is defined as follows:

$$\boldsymbol{\Sigma_\theta} = \sigma^2\boldsymbol{\Lambda_\theta}\boldsymbol{\Lambda}_{\boldsymbol{\theta}}^T \tag{2.9}$$

Some comments are in order:

- The formulation with $\boldsymbol{\Lambda_\theta}$ allows to find a relatively easy expression for the profiled log-likelihood as can be seen in the following steps
- This kind of definition allows the model to comprehend the fitting of singular covariance matrices instead of a definition of $\boldsymbol{\Lambda_\theta}$ as a function of $\boldsymbol{\Sigma}$
- the matrices $\boldsymbol{Z}$ and $\boldsymbol{\Lambda_\theta}$ are tipically sparse, implying some model assumptions and certain matrix representation techniques which will be clear in chapter 3

The model recently described, is reparametrized as follows for computational and efficiency matters. This reparameterization is made using a spherical random effect:

$$\mathcal{U} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_q)$$

and setting:

$$\mathcal{B} = \boldsymbol{\Lambda_\theta} \mathcal{U}$$

In this way $\mathcal{B}$ has the distribution in 2.8. The expected conditional value of the response will then be:

$$\mathbb{E}(\boldsymbol{Y}|\mathcal{U}) = \boldsymbol{X\beta} + \boldsymbol{Z\Lambda}_\theta \boldsymbol{u}$$

It is now important to understand the nature of this parameter $\boldsymbol{\theta}$ and the relative covariance factor $\boldsymbol{\Lambda_\theta}$.
$\boldsymbol{\Lambda_\theta}$ is a block diagonal matrix (in the simple case in which we have a single random effect term), each lower-triangular block is a copy of a so called *template matrix*, $\boldsymbol{T}$. The covariance parameter vector $\boldsymbol{\theta}$ consists of the lower triangle of the template matrix $\boldsymbol{T}$.
The understanding of the definition of the parameter $\boldsymbol{\theta}$ will be useful later in the analysis on singular fits.
The estimation of the random effect vector and the parameter $\boldsymbol{\theta}$ will be the outcome of an iterative routine consisting of a penalized least square step and an optimization procedure.

2.3.1 Penalized Least Squares

The penalized least square method (PLS) involves minimizing the following quantity:

$$r^2(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{u}) = \rho^2(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{u}) + ||\boldsymbol{u}||^2 \tag{2.10}$$

over $[\boldsymbol{u}, \boldsymbol{\beta}]^T$, where:

$$\rho^2(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{u}) = ||\boldsymbol{W}^{1/2}[\boldsymbol{y}_{obs} - \mathbb{E}(\boldsymbol{Y}|\mathcal{U})]||^2 \tag{2.11}$$

Notice that r^2 and ρ^2 depend on three arguments, notice also that the penalization is the euclidean norm of the vector $\boldsymbol{u}$ in the objective function. This optimization problem, under reparameterization, can be thought as a standard least square problem. Skipping some of the steps, the PLS estimates satisfy the equation:

$$\begin{bmatrix} \boldsymbol{\Lambda_\theta X}^T\boldsymbol{W}(\boldsymbol{y}_{obs}) \\ \mathbf{X}^T\mathbf{W}(\boldsymbol{y}_{obs}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Lambda_\theta X}^T\boldsymbol{WZ\Lambda_\theta} + \boldsymbol{I} & \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^T\boldsymbol{Z}^T\boldsymbol{W} \\ \boldsymbol{X}^T\boldsymbol{WZ\Lambda_\theta} & \mathbf{X}^T\boldsymbol{WX} \end{bmatrix} \begin{bmatrix} \mathbb{E}(\mathcal{U}|\boldsymbol{Y}) \\ \hat{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \end{bmatrix} \tag{2.12}$$

We may now rewrite the term r^2 as:

$$r^2(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{u}) = r^2(\boldsymbol{\theta}) + ||\boldsymbol{L}_{\boldsymbol{\theta}}^T(\boldsymbol{u} - \mathbb{E}(\mathcal{U}|\boldsymbol{Y})) + \boldsymbol{R}_{ZX}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_{\boldsymbol{\theta}})||^2 + ||\boldsymbol{R}_X(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_{\boldsymbol{\theta}})||^2 \tag{2.13}$$

where $\boldsymbol{R}_X$ and $\boldsymbol{L}_{\boldsymbol{\theta}}$ are factors deriving from the Cholesky factorization of the matrix on the right in 2.12. It is easy now to see how the minimizing terms of the extended penalized least square problem are the one satisfying equation 2.12.

2.3.2 Profiled deviance

In the following section we describe the REML criterion estimation which is the one exploited throughout the work. As an alternative also a ML criterion is available with different profiling steps.

It is possible to write the probability densities of $\boldsymbol{Y}|\mathcal{U}$ and $\mathcal{U}$, through these densities we can compute, using the Bayes Theorem:

$$f_{\boldsymbol{Y},\mathcal{U}}(\boldsymbol{y}_{obs}, \boldsymbol{u}) = \frac{|\boldsymbol{W}|^{1/2}}{(2\pi\sigma^2)^{(n+q)/2}} \exp \frac{-r^2(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{u})}{2\sigma^2} \tag{2.14}$$

Let us now compute the quantity:

$$f_{\boldsymbol{Y}}(\boldsymbol{y}_{obs}) = \int f_{\boldsymbol{Y},\mathcal{U}}(\boldsymbol{y}_{obs}, \boldsymbol{u})\, d\boldsymbol{u} \tag{2.15}$$

The REML criterion uses as objective function the integral of the marginal distribution of $\boldsymbol{Y}$ w.r.t. $\boldsymbol{\beta}$:

$$\int f_{\boldsymbol{Y}}(\boldsymbol{y}_{obs})\, d\boldsymbol{\beta} \tag{2.16}$$

Which under change of variable, applying minus twice the *log* to the integral and after some steps reduces to:

$$-2\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\sigma}^2|\boldsymbol{y}_{obs}) = log\frac{|\boldsymbol{L_\theta}|^2|\boldsymbol{R}_X|^2}{|\boldsymbol{W}|} + (n-p)log(2\pi\sigma^2) + \frac{r^2(\theta)}{\sigma^2} \tag{2.17}$$

The REML estimate for σ^2 turns out to be:

$$\hat{\sigma}^2_{\boldsymbol{\theta}} = \frac{r^2(\boldsymbol{\theta})}{n-p} \tag{2.18}$$

Which leads to the following fully profiled REML criterion:

$$-2\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}_{obs}) = log\frac{|\boldsymbol{L_\theta}|^2|\boldsymbol{R}_X|^2}{|\boldsymbol{W}|} + (n-p)(1 + log(\frac{2\pi}{n-p})) \tag{2.19}$$

Once obtained this form, `lme4` uses a number of optimization techniques to find the minimum for this function.

2.4 Reparameterization procedure

The package `lme4` for mixed models handles an arbitrary number of random effect terms (even if we need just two), and uses a combination of sparse and dense matrix representation to boost efficiency. Unfortunately its use has been limited in genetic applications due to the fact that does not allow correlation between different subjects, so the estimation of the model described in 2.3 is not possible. A suitable model for `lme4`'s estimation has a covariance of the random effect matrix in the form of a block diagonal matrix, while in the case of model 2.3 the covariance of the vector $\boldsymbol{\alpha}$ is not block diagonal. A strategy has been developed by Harville and Callanan, 1989 [8] for the fitting of a correlated random intercept model, and implemented in the `R` package `pedigreemm` (Vazquez et al., 2010 [32]). In the following we describe this strategy which is developed for the case of a multi-dimensional setting.

Consider the following simple mixed effect model:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{\epsilon} \tag{2.20}$$

Notice that this notation is not related to the one of the previous sections, it is used here to avoid cumbersome notational steps. This is not a functional model, but a simple multi-dimensional one. In a standard gaussian linear model the marginal distribution of the data is:

$$Cov(\boldsymbol{Y}) = \boldsymbol{Z}(\sigma_u^2)\boldsymbol{Z}^T + \boldsymbol{R}$$

Where $Cov(\boldsymbol{Y})$ is the variance of the response, σ_u^2 the variance of the random factor, and $\boldsymbol{R}$ the variability of the residuals. This model represents a set of uncorrelated subjects, which is not appropriate for a genetic setting. For genetic purposes we have to set a variance of the random effect vector in the form $\sigma_u^2\boldsymbol{A}$, which implies:

$$Cov(\boldsymbol{Y}) = \boldsymbol{Z}(\boldsymbol{A}\sigma_u^2)\boldsymbol{Z}^T + \boldsymbol{R}$$

where $\boldsymbol{A}$ is the additive relationship matrix of the data. Since subjects are related one to each other, levels have to be correlated in the model. The methodology developed by Harville and Callanan, 1989 [8] consists of post multiplying the model matrix $\boldsymbol{Z}$ by the Cholesky decomposition of the numerator relationship matrix. The factorization is possible since $\boldsymbol{A}$ is positive definite (unless identical twins or clones are in the pedigree, in this case is positive semi-definite).

Let $\boldsymbol{A} = \boldsymbol{LL}^T$ where $\boldsymbol{L}$ is the Cholesky factor of $\boldsymbol{A}$ which can be directly extracted from the pedigree information. Define now:

$$\boldsymbol{Z}^* := \boldsymbol{ZL}$$

and write then:

$$\begin{aligned} Cov(\boldsymbol{Zu}) &= \boldsymbol{ZAZ^T}\sigma_u^2 \\ &= \boldsymbol{ZLL}^T\boldsymbol{Z}^T\sigma_u^2 \\ &= \boldsymbol{(Z^*)(Z^*)}^T\sigma_u^2 \end{aligned} \tag{2.21}$$

define then $\boldsymbol{u^*} = \boldsymbol{L^{-1}u}$ and reparametrize the model:

$$\begin{aligned} \mathbb{E}(\boldsymbol{Y}) &= \boldsymbol{ZLL^{-1}u} + \boldsymbol{X\beta} \\ &= \boldsymbol{Z^*u^*} + \boldsymbol{X\beta} \end{aligned} \tag{2.22}$$

this change of variable let us now have a model which has a simpler random effect covariance, indeed:

$$\begin{aligned} Cov(\boldsymbol{u}^*) &= \boldsymbol{L}^{-1}Cov(\boldsymbol{u})(\boldsymbol{L}^{-1})^T \\ &= \boldsymbol{ZL}^{-1}(\boldsymbol{L}^{-1})^T\sigma_u^2 \\ &= \boldsymbol{I}\sigma_u^2 \end{aligned} \tag{2.23}$$

In this setting the elements of $\boldsymbol{u}^*$ are mutually independent. The `lme4` has now the possibility to fit such a model. Vazquez et al., 2010 [32] develops this factorization in the package `pedigreemm` for random intercept models. The inputs of the fitting procedure are the records and a pedigree, necessary to extract the additive relationship matrix. Unfortunately this reparametrization procedure is not ideal for the kind of model displayed in 2.3. Making use of a more compact notation for 2.3 and using the same structure of 2.20 we can write the following:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{\epsilon} \tag{2.24}$$

where:

$$\boldsymbol{Z} = [\boldsymbol{Z}^E, \boldsymbol{Z}^G]$$

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \end{bmatrix}$$

Where $\boldsymbol{u}$ has the following covariance structure:

$$Cov(\boldsymbol{u}) = \begin{bmatrix} \boldsymbol{A} \otimes \boldsymbol{C}^G & 0 \\ 0 & \boldsymbol{I} \otimes \boldsymbol{C}^E \end{bmatrix}$$

Where $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are independent. It is impossible to fit this kind of model with the strategy in Vazquez et al., 2010 [32] or with the package `pedigreemm` unless the matrix $\boldsymbol{C}^G$ is a scalar.

In the following section we will outline the problem statement and anticipate what will be the proposed approach.

2.5 Problem and purpose statement

The problem that will be faced in this work was briefly anticipated in the previous sections. The first main goal is to provide a routine to fit the model in 2.3 using the strategy described in 2.4, but extending it to any (or almost any) degree k, or number of random effect terms. This means, using the previous notation, being able to fit a model with a vector of random effects $\boldsymbol{\alpha}^*$ of dimension greater than one. The package `pedigreemm` is not able to deal with such complex models.

Once found an estimate for the covariance function (through expression 2.6) it will be interesting, from a genetic point of view, to extract the eigenfunctions and eigenvalues of G. The analysis of the shape of these elements will be helpful to recognize the pattern of growth and evolutionary constraints of the modeled process.

At this point the important question is how reliable these estimates are, it will be then necessary to detect some confidence region/bands for the G function and its eigenvectors, this is the second main goal. The estimation of G and its eigenfunctions for genetic purposes is often performed in literature but it is still missing a procedure to have confidence bands for this estimates. Since we are working with functional data we will

Trait	x	id	sire	dam
2.2	1	10001	1	101
8.6	7	10001	1	101
225.9	16	10001	1	101
3.1	2	10002	1	102
9.1	8	10002	1	102
318.3	16	10002	1	102
12.4	6	10003	2	103
..	..	..	..	..

TABLE 2.1: *Triboleum Castaneum* dataset

need a uniform criteria to estimate these intervals. In the following a bootstrap procedure is implemented to show the good quality of a descriptive simultaneous confidence band instead of a point-wise interval estimation.

2.6 *Triboleum Castaneum* dataset

It is worth to dedicate a section to the description of the dataset which will be used in the following chapters. *Triboleum Castaeum* is a common insect model for population genetic and development studies. The trait reported is the body mass measured in $10^{-5}g$, while the independent variable is time, more specifically days post-hatch, of each of the individuals. The curves in the dataset are displayed in figure 2.1. The dataset is called `TRFUN25PUP4` and was built for Irwin and Carter's experiment [15]. It contains 873 individuals and 6860 measurements, with an average of approximately 8 measurements per individual. Sampling points are not taken at fixed times as they vary in number and location, the range of days measured is 1-25 days.

All of the individuals were derived from a stock population of the cSM++ strain. The stock population contained approximately 300 individuals initially divided into 17 population and allowed to breed freely. Offspring were collected, their sex recorded and then isolated into separate vials to create a stock of virgin adults. The full experiment (in terms of temperatures, feeding information, humidity) is described in Irwin and Carter, 2013 [15], from which we are also taking the dataset.

The dataset comes in the form displayed in table 2.1:

In the simulation studies (chapter 4) we will use, only the `x`, `id`, `sire` and `dam` columns, while in chapter 4, the actual case study, we will use also the response vector `Trait`.

The columns `sire` and `dam` on the right are used to build the pedigree and the additive relationship matrix. `sire` values go from 1 to 100 while `dam` ones go from 100 to 1000.

Chapter 3

Proposed Approach

In the following sections we will describe the proposed approach to the problem. The chapter is divided in:

- Model reparameterization, in this part we reparameterize the model 2.3 in order to be able to use the package `lme4` for our genetic fitting
- Parametric Bootstrapping Inference, a bootstrapping inference strategy is proposed in this section. More specifically we propose a point-wise and a simultaneous confidence bands approach for the covariance function G and its first eigenfunctions.

3.1 Model Extension

In this section we will use the same notation and quantities defined in section 2.4. Let p be the dimension of the random effect vector $\mathbf{u}$ and consider the matrix:

$$\boldsymbol{M}^{-1} := \boldsymbol{L}^{-1} \otimes \boldsymbol{I}_p \tag{3.1}$$

where $\boldsymbol{L}$ is the Cholesky factor of the additive relationship matrix $\boldsymbol{A}$.
Now consider the model in 2.20 and set:

$$\boldsymbol{Z}^* := \boldsymbol{Z}\boldsymbol{M}$$

$$\boldsymbol{u}^* := \boldsymbol{M}^{-1}\boldsymbol{u}$$

Then we can reparametrize in the following way:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{\epsilon}$$

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{ZMM}^{-1}\boldsymbol{u} + \boldsymbol{\epsilon}$$

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{Z}^*\boldsymbol{u}^* + \boldsymbol{\epsilon}$$

Let us now check the covariance structure of the new random effect vector $\boldsymbol{u}^*$:

$$\begin{aligned} Cov(\boldsymbol{u}^*) &= Cov(\boldsymbol{M}^{-1}\boldsymbol{u}) \\ &= \boldsymbol{M}^{-1}Cov(\boldsymbol{u})(\boldsymbol{M}^{-1})^T \end{aligned} \tag{3.2}$$

For the sake of simplicity we will assume here the vector $\boldsymbol{\gamma}$ to be zero, so $Cov(\boldsymbol{u}) = (\boldsymbol{A} \otimes \boldsymbol{C}^G)$, we will later describe the general strategy.
Substituting then this value to $Cov(\boldsymbol{u})$:

$$\begin{aligned} Cov(\boldsymbol{u}^*) &= \boldsymbol{M}^{-1}(\boldsymbol{A} \otimes \boldsymbol{C}^G)(\boldsymbol{M}^{-1})^T \\ &= (\boldsymbol{L}^{-1} \otimes \boldsymbol{I}_p)(\boldsymbol{A} \otimes \boldsymbol{C}^G)(\boldsymbol{L}^{-1} \otimes \boldsymbol{I}_p)^T \end{aligned} \tag{3.3}$$

Using the properties of the Kronecker product:

$$\begin{aligned} Cov(\boldsymbol{u}^*) &= (\boldsymbol{L}^{-1}\boldsymbol{A} \otimes \boldsymbol{I}_p\boldsymbol{C}^G)(\boldsymbol{L}^{-1} \otimes \boldsymbol{I}_p)^T \\ &= (\boldsymbol{L}^{-1}\boldsymbol{L}\boldsymbol{L}^T \otimes \boldsymbol{I}_p\boldsymbol{C}^G)(\boldsymbol{L}^{-1} \otimes \boldsymbol{I}_p)^T \\ &= (\boldsymbol{L}^T \otimes \boldsymbol{C}^G)(\boldsymbol{L}^{-1} \otimes \boldsymbol{I}_p)^T \\ &= (\boldsymbol{L}^T \otimes \boldsymbol{C}^G)((\boldsymbol{L}^{-1})^T \otimes \boldsymbol{I}_p) \\ &= \boldsymbol{L}^T(\boldsymbol{L}^{-1})^T \otimes \boldsymbol{C}^G \\ &= \boldsymbol{I} \otimes \boldsymbol{C}^G \end{aligned} \tag{3.4}$$

We can see then that covariance matrix of the random effect vector is the Kronecker product of the identity and the matrix $\mathbf{C}^G$. This covariance structure represents correlation between components of each subject whilst indipendence between one subject and another one. `lme4` is capable to fit such a model.

The strategy is then to fit a mixed effect model with a modified random effect matrix $\boldsymbol{Z}$ and get an estimate of the components of $\mathbf{C}^G$.

Extending this strategy to the case of an additional environmental effect term we will fit (via `lme4`) the following model:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}^*\boldsymbol{u}^* + \boldsymbol{\epsilon} \tag{3.5}$$

where the matrix $\boldsymbol{Z}$ will depend on $\boldsymbol{Z}^E$ and $\boldsymbol{Z}^G$ defined in chapter 2, and $\boldsymbol{Z}^*$, $\boldsymbol{u}^*$ will be:

$$\boldsymbol{Z}^* = [\boldsymbol{Z}^E\boldsymbol{M}, \boldsymbol{Z}^G]$$

$$\boldsymbol{u}^* = \begin{bmatrix} \boldsymbol{M}^{-1}\boldsymbol{\alpha} \\ \boldsymbol{\gamma} \end{bmatrix}$$

More specifically, once fixed a basis for the functional space, the matrix $\boldsymbol{X}$ will be in the form:

$$\boldsymbol{X} := \begin{bmatrix} \phi(t_{11}) \\ \phi(t_{12}) \\ . \\ . \\ \phi(t_{Nl_N}) \end{bmatrix}$$

while $\boldsymbol{Z}$ will be built as a function of $\boldsymbol{X}$ as explained in chapter 2. The vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are the coefficients of the random regression whose covariance matrix is needed for the estimation of the covariance function of the process as stated in formula 2.6.
This reparameterization let us exploit the stability of the largely used package `lme4` fitting models that the package alone is not able to manage independently.
In chapter 4 we will develop our version of the simple multivariate routine and compare it to the one of the `R` package `pedigreemm` in the random intercept case, testing the discrepancy in the results. We will then extend the strategy to a general model.

3.2 Parametric Bootstrapping Inference

Once found an estimate for the covariance function $\hat{G}$ one purpose of this work is to build a tool to make inference around this estimate. The goal of this section is to find a confidence region/band of level α around $\hat{G}$.
We want to construct a pointwise confidence band for the covariance G and its main eigenfunctions. Later we will find a suitable statistic for our bootstrap sample to let us find simultaneous confidence bands for the same elements. First we briefly describe the bootstrap procedure used to build the sample. For more information about the general bootstrap strategy and its justification see Davison et al., 1997 [13] and Efron et al., 1994 [14], and for bootstrap in FDA see Cuevas, 2014 [12].

Bootstrap Procedure

In the following we will go through the bootstrap algorithm 1 stated below. Given a certain functional dataset, let $\hat{G}$ and $\hat{E}$ be the estimated genetic and environmental covariance functions (line 3). To this functions correspond a couple of covariance matrices $\hat{\boldsymbol{C}}^G$ and $\hat{\boldsymbol{C}}^G$ (line 4), which are the coefficients of the functions $\hat{G}$ and $\hat{E}$ over the chosen basis of the space.

From $\hat{\boldsymbol{C}^G}$ and $\hat{\boldsymbol{C}^E}$ is possible to generate a new functional response vector using these matrices as terms of the covariance structure of the random effect $\boldsymbol{u}$ (line 8).

Let's say we generate S functional datasets, from each one of these we can estimate the covariance functions G_i and E_i, $i \in \{1, .., S\}$ (line 9). We will call $\{(G_1, E_1), .., (G_S, E_S)\}$ the *bootstrap sample.*

Follows the algorithmic breakdown of the bootstrap routine:

Algorithm 1 Parametric Bootstrap Procedure

```
 1: procedure BOOTSTRAP
 2:     $Y \leftarrow$ response vector
 3:     $\hat{G}, \hat{E} \leftarrow$ estimates using the algorithm fme($Y$)
 4:     $\hat{\boldsymbol{C}^G}, \hat{\boldsymbol{C}^E} \leftarrow$ estimated variances of the random effect vectors
 5:     Set a model $M$ using the matrices $\hat{\boldsymbol{C}^G}, \hat{\boldsymbol{C}^E}$
 6:     Set $S$ as bootstrap sample
 7: loop:
 8:     Generate a response vector $Y_{(i)}$ from model $M$
 9:     $G_i, E_i \leftarrow$ estimated covariance functions from $Y_{(i)}$.
10:     $S \leftarrow add(S, (G_i, E_i))$
11:     close;
12:     return $S$.
```

Pointwise vs Simultaneous intervals

Given the bootstrap sample we have a set of S couples of genetic and enviromental covariance functions. We will first describe pointwise and simultaneous intervals for the covariance function, then the ones for the eigenfunctions.

Let us consider our estimate for the covariance function $\hat{G}$, it needs to be pointed out that this estimate is a random process:

$$\hat{G}(\omega, \boldsymbol{t}) : \Omega \times [0, T]^2 \to \mathbb{R}$$

Now given a fixed point $\bar{\boldsymbol{t}}$ we would like to find a couple of coefficients $c_{1-\alpha/2}$ and $c_{\alpha/2}$ s.t.:

$$P(c_{1-\alpha/2} \leq \hat{G}(\omega, \bar{\boldsymbol{t}}) - G(\bar{\boldsymbol{t}}) \leq c_{\alpha/2}) = 1 - \alpha$$

Since the distribution of the r.v. $\hat{G}(\omega, \bar{\boldsymbol{t}}) - G(\bar{\boldsymbol{t}})$ is not known we need to approximate this probability via the bootstrap sample described in the previous paragraph. Reminding that also the element of the bootstrap sample are random processes in the form:

$$G_i(\omega, \boldsymbol{t}) : \Omega \times [0,T]^2 \to \mathbb{R}$$

let us set the following quantities:

- $V(\omega) := \hat{G}(\omega, \bar{\boldsymbol{t}}) - G(\bar{\boldsymbol{t}})$
- $\widetilde{V}(\omega) := G_i(\omega, \bar{\boldsymbol{t}}) - \hat{G}(\omega, \bar{\boldsymbol{t}})$

we will perform the following approximation:

$$1 - \alpha = P(c_{1-\alpha/2} \leq V \leq c_{\alpha/2}) \cong P(c^b_{1-\alpha/2} \leq \widetilde{V} \leq c^b_{\alpha/2}) \tag{3.6}$$

At this point, given a sample of observations $G_i(\omega, \bar{\boldsymbol{t}})$ with $i \in 1,..,S$ we can find the quantiles of order $\alpha/2$ and $1-\alpha/2$ of $\widetilde{V}$ using its empirical distribution, this quantities are the $c^b_{\alpha/2}$ and $c^b_{1-\alpha/2}$ displayed in the previous formula.
This leads to the following pointwise band:

$$P\{\ \hat{G}(\omega, \bar{\boldsymbol{t}}) - c^b_{\alpha/2}(\bar{\boldsymbol{t}}) \leq G(\bar{\boldsymbol{t}}) \leq \hat{G}(\omega, \bar{\boldsymbol{t}}) - c^b_{1-\alpha/2}(\bar{\boldsymbol{t}})\ \} = 1 - \alpha \tag{3.7}$$

It is important to point out why this approximation is legit, in the following we qualitatively describe the rationale behind (for a more detailed approach see Davison et al., 1997[13] and Efron et al., 1994 [14]):

- First thing we use the fact that the distributions of V and $\widetilde{V}$ are similar, this is ensured by the bootstrap principle. It is intuitive indeed that the difference of the estimate from the true covariance function has the same distribution of the distance between a bootstrap element and the "center" of its distribution, i.e. the estimate $\hat{G}$
- We need also a result of convergence of the estimate to the real G to make the approximation in 3.6. We know that fixing a basis for our functional space to

represent the covariance function allows us to represent each function with a finite number of coefficients. These coefficients are the elements of the matrices $\hat{\boldsymbol{C}}^G$ and $\hat{\boldsymbol{C}}^E$. Through restricted maximum likelihood procedure we have results on convergence of these coefficients to the true ones, which are the elements of the matrices $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$.

A very similar procedure can be built for the simultaneous confidence bands, in this case we want to find a scalar d_α s.t.

$$P(\sup_{t\in[0,T]^2} |\hat{G}(\omega,\boldsymbol{t}) - G(\boldsymbol{t})| \leq d_\alpha) = 1-\alpha$$

using the same approximation as before we can write:

$$\begin{aligned} 1-\alpha &= P(\sup_{\boldsymbol{t}\in[0,T]^2} |\hat{G}(\omega,\boldsymbol{t}) - G(\boldsymbol{t})| \leq d_\alpha\ ) \\ &\cong P(\sup_{\boldsymbol{t}\in[0,T]^2} |G_i(\omega,\boldsymbol{t}) - \hat{G}(\omega,\boldsymbol{t})| \leq d^b_\alpha\ ) \end{aligned} \tag{3.8}$$

And as before we can approximate the distribution of the random variable

$$\sup_{\boldsymbol{t}\in[0,T]^2} |G_i(\omega,\bar{\boldsymbol{t}}) - \hat{G}(\omega,\bar{\boldsymbol{t}})|$$

with the empirical one computed from the bootstrap sample.
Once we have the empirical quantile d^b_α of order α we can compute the simultaneous confidence band:

$$P\{\ \hat{G}(\omega,\boldsymbol{t}) - d^b_\alpha \leq G(\boldsymbol{t}) \leq \hat{G}(\omega,\boldsymbol{t}) + d^b_\alpha \quad \forall \boldsymbol{t}\in[0,T]^2\} = 1-\alpha \tag{3.9}$$

The exact same procedure can performed using the bootstrap sample of the eigenfunctions. Here after we will consider the eigenfunctions ordered w.r.t. the correspondent eigenvalue in a decreasing trend. Using the following notation:

- $\hat{e}^{(i)}$: the i-th eigenfunction of the estimated covariance function $\hat{G}$

- $\{e_1^{(i)}, e_2^{(i)}, .., e_S^{(i)}\}$ will be the bootstrap sample of eigenfunctions, each $e_j^{(i)}$ is the i-th eigenfunction of the j-th genetic covariance function G_j of the bootstrap sample

Following the same strategy of above it is now possible to extract a pointwise confidence band for $e^{(i)}$:

$$P\{\ \hat{e}^{(i)}(\omega, \bar{\boldsymbol{t}}) - h^b_{\alpha/2}(\bar{\boldsymbol{t}}) \leq e^{(i)}(\bar{\boldsymbol{t}}) \leq \hat{e}^{(i)}(\omega, \bar{\boldsymbol{t}}) - h^b_{1-\alpha/2}(\bar{\boldsymbol{t}})\ \} = 1 - \alpha \tag{3.10}$$

and a simultaneous confidence interval:

$$P\{\ \hat{e}^{(i)}(\omega, \boldsymbol{t}) - l^b_\alpha \leq e(\boldsymbol{t}) \leq \hat{e}^{(i)}(\omega, \boldsymbol{t}) + l^b_\alpha \quad \forall \boldsymbol{t} \in [0, T]\ \} = 1 - \alpha \tag{3.11}$$

where $h^b_{\alpha/2}(\bar{\boldsymbol{t}})$, $h^b_{1-\alpha/2}(\bar{\boldsymbol{t}})$ and l^b_α are the pointwise and simultaneous bootstrap quantiles for the i-th eigenvector.

Notice that the coverage for the estimated eigenfunction bands is not simultaneous for multiple eigenvectors, for this reason each simultaneous band is different for distinct eigenfunctions. This choice is given by the fact that in a genetic analysis we could be interested in the confidence band of only one of the eigenvectors, this can happen, for example, when we want to produce a specific effect in the mean phenotypic response (the one caused by a specific direction).

Chapter 4

Simulation studies

In this chapter we will build an algorithmic solution to the problem in section 2.5 in several steps. The starting point will be fitting a simple random intercept model, we will then gradually increase the complexity until reaching the final functional formulation. Here an outline of the sections of this chapter:

- Section 4.1: In the first part of this section we will build a procedure able to fit the simple multi-variate model described in section 2.4, i.e. a random intercept model (which is manageable by already existing packages), and compare our result with the estimates of `pedigreemm`. In the second part the complexity will be extended to random slope models (see section for more details), and the quality of the estimates will be measured through a simulation

- Section 4.2: Here a full functional model is set up using the dataset from *Triboleum Castaneum*. We will perform a simulation and apply the bootstrap inference criteria (Section 3.2) on the estimated covariance and eigenfunctions

It is important to introduce, before starting to describe the implementation, the syntax of the formulas used to fit mixed models in `lme4` package for `R`. Indeed knowing this syntax let us have an extremely compact representation of any mixed effect model.
Let `Y` be a response column, `x1` a column of covariates and `id` an identifier relative to a certain subject. Now consider the following formula:

```
Y ~ 1 + x1 + (1 + x1 | id)
```

The first two terms on the right side of the relation are responsible for the construction of the fixed effect matrix $\boldsymbol{X}$ which is built in the same way as in `lm`. Talking about the last term, and using the notation of section 2.2, `lmer` will build a matrix $\boldsymbol{J}$ based on `id` and construct a random effect matrix $\boldsymbol{Z}$ as the Khatri-Rao product of $\boldsymbol{J}$ and $\boldsymbol{X}$ (the detailed explanation is present in section 2.2).

4.1 Preparatory simulations

Random intercept model

The goal of this part is to set a certain simple model, perform a simulation using as fitting procedures `pedigreemm` and our algorithm to detect any significant differences in the result. Our algorithm will be called hereafter `fme` (from fit-mixed-effect). We will analyze the dataset `milk`, a milk yield dataset obtained from existing databases at the USDA-ARS Animal Improvement Programs Laboratory (Beltsville, MD), it is also used by Vazquez et al., 2010 [32] in their descriptive paper on the package `pedigreemm`.
The dataset is in the following form:

Sd milk production	# lactations	Days in Milk	ID
6.371997	1	242	4930
4.624097	2	420	5906
6.458307	3	506	6222
..	..	..	..

There are 3397 records from 1359 different cows. Different records can come from the same cow considered in different lactations (i.e number of different times a cow has produced milk). These cows are daughters of 38 sires in 57 herds and the pedigree information is available in the package `pedigreemm`. The relevant elements for our analysis are:

- Sd milk production (`SDM`), milk produced in kg, then standardized. This will be our response variable

- Number of lactations (`lact`): the number of calvings a cow has had, this is reasonably a relevant covariate for the prediction of the milk production
- Days in milk (`DIM`), number of days of milk production
- Identification number of the cow (`ID`)

We set at first the following model:

$$\texttt{SDM} \sim \texttt{lact + log(DIM) + (1|ID)}$$

Here the matrix $\boldsymbol{X}$ is a $n \times 3$ matrix while the random effect term's $\boldsymbol{Z}$ is an $n \times N$. The variance of the random effect vector has the following form:

$$\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \sigma^2_{rand-eff} * \boldsymbol{A})$$

We know choose some values for $\sigma^2_{rand-eff} = 2$ and $\sigma_{res} = 0.1$. Through these parameter we simulate 10000 response vectors and, for each response vector, estimate 2 couples of parameters, one from `pedigreemm`, the other from `fme`. The sample will contain a total of 10000 couples estimates for the variance of the random effect $\sigma^2_{rand-eff}$ and the residuals σ^2_{res}.

In figure 4.1 are displayed two boxplots: the first one shows the comparison between the variance of the random effect $\sigma^2_{rand-eff}$ estimated by respectively `pedigreemm` and `fme`, the second shows the same comparison for the variance of the residuals $\sigma^2_{residuals}$.

In figures 4.2 and 4.3 instead we can see two scatterplots from the same sample: the first shows the variance of the random effect, the estimate of `fme` against `pedigreemm`'s, the second shows the variances of the residuals compared.

In this first stage the results seem to point in the right direction. As a last visualization tool we show the distributions of the two couples of parameters in figure 4.4.

One of the purposes of this work is to quantify the quality of the estimates of our algorithm. In this section we will use simple indicators for the goodness of our fit, like RMSE, and perform some tests. In the following section we will develop a proper inference strategy for our functional data, which is impossible to adopt now since we are

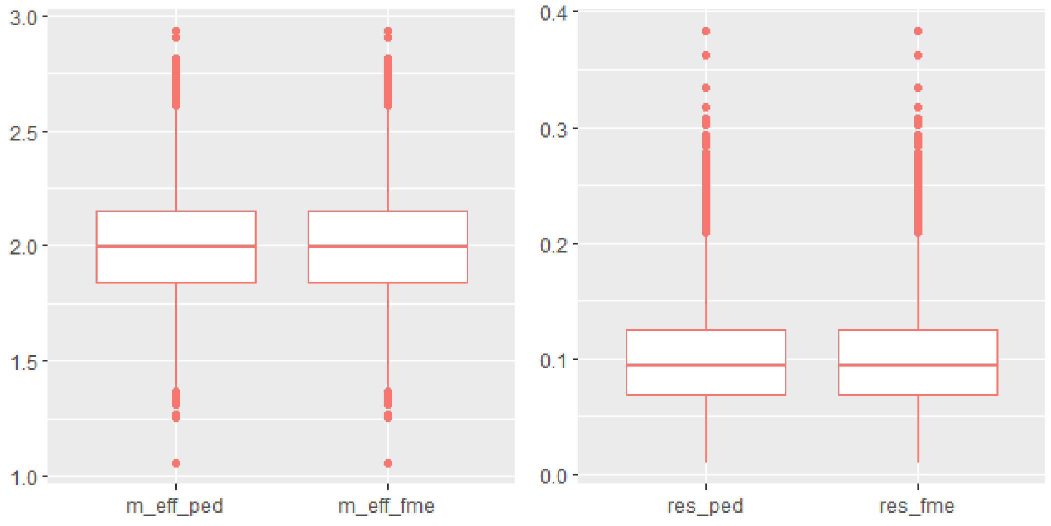

FIGURE 4.1: Mixed effect and residuals variances for the two fits

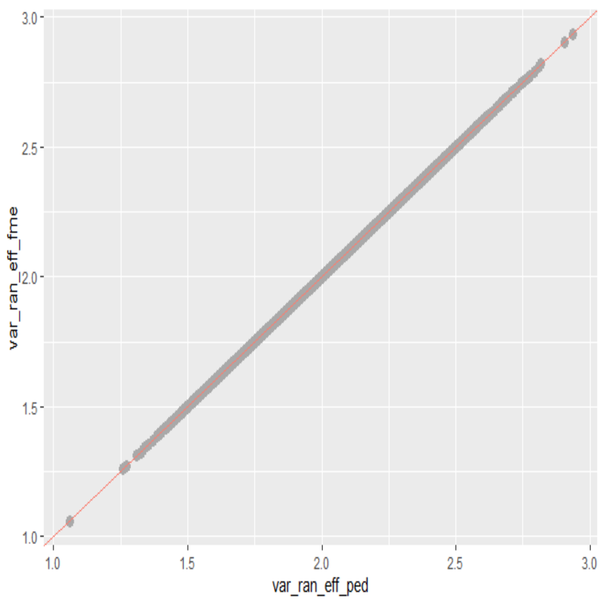

FIGURE 4.2: Mixed effect and residuals variances for the two fits

testing `fme` on multi-variate models.

A way to compare `pedigreemm` and `fme` at this stage is confronting the two RMSE computed w.r.t. the true value. The results are displayed in table 4.1, along with the ratio between the difference of the two and one of the RMSE:

	RMSE fme	RMSE pedigreemm	Abs diff/RMSE
$\sigma^2_{rand-eff}$	0.22473907	0.22473906	3.5e-08
σ^2_{res}	0.04393078	0.04393079	2.6e-07

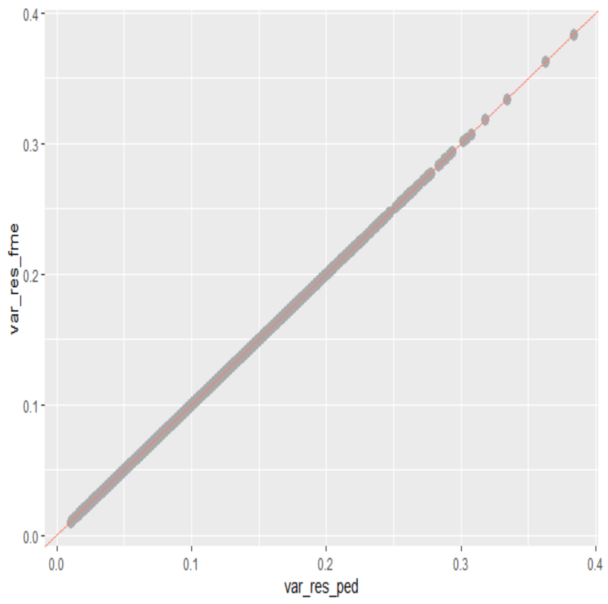

FIGURE 4.3: Mixed effect and residuals variances for the two fits

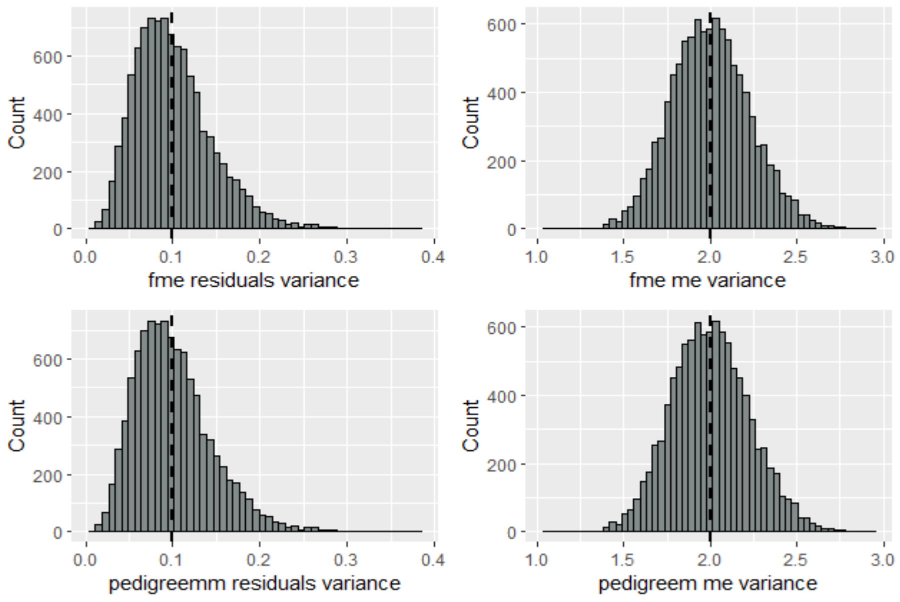

FIGURE 4.4: Mixed effect and residuals variances for the two fits

It will be useful also to keep an eye on `fme`'s indicator $RMSE/TrueValue$ which is 0.11 for $\sigma^2_{rand-eff}$ and 0.44 for σ^2_{res}. This index will be useful to assess to some extent the precision of the estimates computed in the second part of this section.

Let us now conduct some tests to compare the distribution of the two samples:

- **Mann-Whitney**, this test is a non parametric, uncoupled test with alternative hypothesis a difference in the two distributions. The two couples of samples were tested, the outcome is a p-value of 1 for both tests

- **Signed Rank**, this is a non-parametric, coupled test with the same null hypothesis as the Mann-Whitney, its outcome is a p-value of 0 (later the discussion of the possible causes)
- **Kolmogorov-Smirnov** a non parametric test for the same aim of the previous but using a different statistic, the result is a p-value of 1 for both tests

The outcomes of the first and third test are positive, however it is appropriate to mention that the Kolmogorov-Smirnov alternative is less indicated for large samples (10000 is quite a large number), but it is still worth mentioning the result. The coupled test gives a negative result instead, the possible reason is that given such a large sample, even a slight difference in the two distributions happens to be strongly statistically significant. In this part of the work we withhold our conclusion waiting to have a full functional model (section 4.2) and making inference on it.

Random slope model

In this second part of the section we extend the strategy described in 2.4 as explained in 3.1 and apply `fme` to the following model:

```
SDM ~ lact + log(DIM) + (1+log(DIM)|ID) + (1+log(DIM)|ID)
```

This `R` formula correspond to the following matrix form:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{\epsilon}$$

and random effect covariance structure:

$$\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{A} \otimes \boldsymbol{C}^G & 0 \\ 0 & \boldsymbol{I} \otimes \boldsymbol{C}^E \end{bmatrix})$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_{res} * \boldsymbol{I}_n)$$

Where $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$ are 2×2 and A is a square matrix with the same dimension as the number of subjects (1359). This is the step where we go beyond the strategy described in Vazquez et al., 2010 [32] and generalize to any number of covariates inside the random effect term. For this reason, from now on it will not be possible to compare the results of `fme` with the ones of `pedigreemm`, since this package is not able to fit such models. It is then necessary to set up a simulation fixing some known parameters, simulating, using `fme` to fit those parameters and see how good is the fit w.r.t. the true initial values.
Let us set a couple of arbitrary matrices $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$, parameters of the model above, then each iteration i of the simulation will have the following workflow:

1. simulate $\boldsymbol{u}$ and $\boldsymbol{\epsilon}$ from the upper model and build a response vector $\boldsymbol{Y}_{(i)}$
2. fit the dataset with response vector $\boldsymbol{Y}_{(i)}$ and extract the estimates for the matrices $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$

In the end of the procedure we will have a sample of dimension S in the form

$$\{\{\boldsymbol{C}^G_{(1)}, \boldsymbol{C}^E_{(1)}, \sigma^2_1\}, \{\boldsymbol{C}^G_{(2)}, \boldsymbol{C}^E_{(2)}, \sigma^2_2\}, .., \{\boldsymbol{C}^G_{(S)}, \boldsymbol{C}^E_{(S)}, \sigma^2_S\}\}$$

where σ^2_i stands for the variance of the residuals in the i-th sample.
Let us now set some numerical values and perform the simulation, we fix the matrices:

$$\boldsymbol{C}^G = \begin{bmatrix} 2 & 0 \\ 0 & .2 \end{bmatrix}$$

$$\boldsymbol{C}^E = \begin{bmatrix} 4 & 0 \\ 0 & .4 \end{bmatrix}$$

and a variance of the residuals $\sigma^2_{res} = 0.01$
This model was more complex than the previous one and its fitting during the simulation gave some issues. Referring to section 2.3 we need to underline some of the problems

that can come out. In the following we describe the kind of issues that can arise in a fitting (and in a simulation) and how to detect and avoid them:

1. Unidentifiability: this problem comes out in the PLS step, when the dimension of $\boldsymbol{u}$ is higher than the number of observations. This happens when the model has too many covariates in the random effect terms, the result is unidentifiability in the PLS step of the estimation procedure

2. Singularity of the fit: this is a very common issue, singularity comes out in the non linear optimization step, when the optimum parameter $\hat{\boldsymbol{\theta}}$ turns out to be on the boundary of the constrained optimization domain. More precisely when the diagonal of the template matrix $\boldsymbol{T}$ has some zero elements, in this case even the estimated covariance matrix happens to be singular.

The singularity problem is a serious one and applies most of the times when we don't have information about the right number of random covariates to use for the fitting. This is not the case since we know the model that is generating the data, the problem will be of central importance in the last chapter, where will be discussed in more detail. In the following we show some actions that can be taken even in this multi-variate setting to improve the fit:

1. To prevent unidentifiability just lower the number of covariates until the model is identifiable

2. Reduce the variance of the noise (this is possible only in a simulation) and use datasets as large as possible to prevent singular fits

3. Try different non-linear optimizers, some of them are slow but very effective, some others are fast but less reliable

Once fitted the model we will adopt the same approach used in the first part of the section, comparing each term of the estimated matrices to the quantities fixed at the beginning ($\boldsymbol{C}^E$, $\boldsymbol{C}^G$ and σ^2_{res}). In figures 4.5 and 4.6 we can see the three term of the genetic matrix (4.5), and the three elements of the environmental matrix with the residuals (4.5). The black dashed line indicates the true value fixed at the beginning. Notice that the IQR of the genetic coefficients is smaller than the one of the environmental part. This is helpful since, as we said before, our quantity of interest is the estimate of the genetic term. Some of the outliers are consistently far from the true value, this could be caused by the underlying routine of `lme4`, can happen indeed that `lme4` wrongly estimates a parameter as zero, and this causes a much higher estimation for one of the other elements.

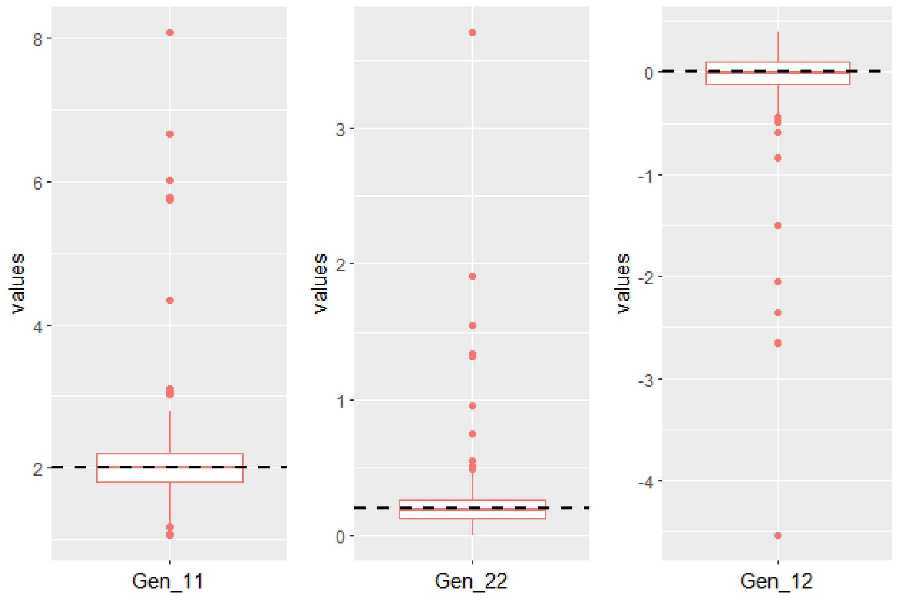

FIGURE 4.5: Box-plots of the elements of the genetic matrix

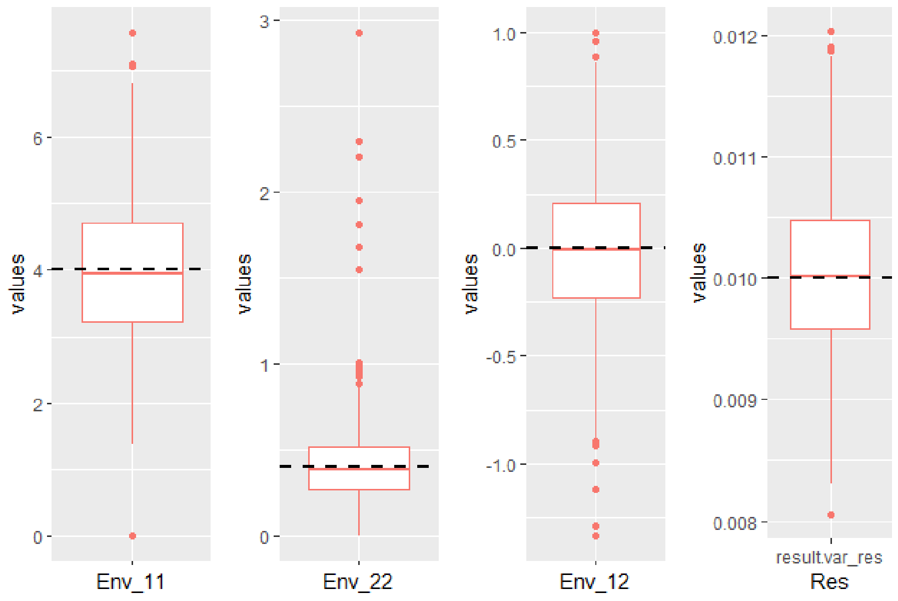

FIGURE 4.6: Box-plots of the elements of the environmental matrix

Following the same strategy of the first part we display the ratio between the RMSE and the true value for each of the elements on the diagonals of $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$ and for the residuals. For the terms of the off diagonals, whose true value is zero, we will just show the RMSE. $R - RMSE$ stands for Ratio-RMSE:

$$R - RMSE_G = \begin{bmatrix} 0.56 & * \\ * & 1.1 \end{bmatrix}$$

$$R - RMSE_E = \begin{bmatrix} 0.28 & * \\ * & 0.7 \end{bmatrix}$$

$$R - RMSE_{res} = 6.74e - 02$$

The RMSEs for the samples of the off-diagonal elements are respectively 0.34, 0.36. Some of the indexes have worsened w.r.t. the one of the first simulation, especially the ones relative to the random slope which however are of secondary importance. However, as said before we withhold our judgement waiting for the result of the final functional inference criteria.

4.2 Functional Model Simulation

In this section we finally introduce the functional approach to our analysis. For this last part of the work we will not use the dataset `milk` and adopt the one described in the introduction on the flour beetle *Triboleum Castaneum*. This insect has been richly studied in the past and it showed a significant heritability in body mass at several stages of development (Okada & Hardin, 1967 [27]; Conner & Via, 1992 [11]; Campo & Rodriguez, 1986 [9]). The trait reported is the body mass measured in $10^{-5}g$, while the independent variable is time, more specifically days post-hatch, of each of the individuals. This dataset is called `TRFUN25PUP4` and contains 873 individuals and 6860 measurements, with an average of approximately 8 measurements per individual. Sampling points are not taken at fixed times as they vary in number and location, the range of days measured is 1-25 days.

Here there are the first rows of the dataset:

Trait	x	id
2.2	1	10001
8.6	7	10001
31.4	11	10001
3.1	2	10002
9.1	8	10002
..	..	..

A whole breeding experiment for the collection of these data has been conducted, we do not report it; for a detailed explanation of the design we refer to Irwin and Carter, 2013 [15]. In the following we will use only some of the information present in the dataset: the `x` values, the response `Y`, the `id` and the pedigree. The dataset and the pedigree are also available with Irwin and Carter, 2013 [15].
In this section we will build two simulations using `TRFUN25PUP4`'s covariates to test the validity of `fme` and the inference strategy.
The steps will be the following:

- *Functional Simulation*, in this part some true parameters will be fixed, a number of responses will be simulated and the parameters fitted through `fme`. These fittings will enable us to compute a sample of estimated covariance functions and eigenfunctions to be compared with the true elements

- *Functional Bootstrap simulation*, in this section we will fix a number of parameters as before, and this time simulate a single response. This response will be fitted generating a single estimate, then a bootstrap sample is produced using as center the fitted values. At this point we will construct pointwise and simultaneous confidence bands for the true covariance and eigenfunctions. Lastly we will compare the confidence bands with the true initial parameters

4.2.1 Functional Simulation

It is necessary now to explain in detail the steps of this simulation. Let us start with the introduction of the model that will be used:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{Z}^G\boldsymbol{\alpha} + \boldsymbol{Z}^E\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

with the following distribution of the random vectors:

$$\boldsymbol{\alpha} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{A} \otimes \boldsymbol{C}^G)$$

$$\boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_N \otimes \boldsymbol{C}^E)$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_{res} * \boldsymbol{I}_n)$$

We refer to chapter 2 for the detailed definition of the matrices $\boldsymbol{Z}^G$ and $\boldsymbol{Z}^E$. First thing to do is to fix a basis for our functional space, it is indeed necessary to build the matrices $\boldsymbol{Z}^G$ and $\boldsymbol{Z}^E$ which contain the elements of the basis evaluated in some points of the domain. For our simulations we will set a Fourier basis, however, in general for a fitting it is appropriate to choose the suitable one based on the characteristics of the dataset.
Once decided the basis to use, we set the distributions of the random effect vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$. This is equivalent to fix the matrices $\boldsymbol{C}^G$, $\boldsymbol{C}^E$ and a value for the variance of the residuals σ^2_{res}.
More specifically we do the following:

- Reparametrizing the t axis from $[1, 25] \to [-1, 1]$
- Considering a Fourier vector basis of degree 3 so:

$$\boldsymbol{\phi}(t) = [\ \frac{1}{\sqrt{T}}\ ,\ \sqrt{\frac{2}{T}}cos(\frac{2\pi t}{T})\ ,\ \sqrt{\frac{2}{T}}sin(\frac{2\pi t}{T})\ ]^T$$

- Building an $\boldsymbol{X}$ matrix using the same covariates used for the $\boldsymbol{Z}$ matrix

We can now fully define the model. Indeed, under these conditions:

- $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$ are 3×3 matrices of correlations indices between the elements of each $\boldsymbol{\alpha}^*_i$, $\boldsymbol{\gamma}^*_i$

- n number of total observations is 6860
- N number of individuals is 873

Our goal is now to fix a couple of true covariance matrices. Let us set the following values:

$$\boldsymbol{C}^E = \begin{bmatrix} 400 & 40 & 0 \\ 40 & 200 & 40 \\ 0 & 40 & 200 \end{bmatrix} (10^{-5}kg)^2$$

$$\boldsymbol{C}^E = \begin{bmatrix} 400 & 40 & 0 \\ 40 & 200 & 40 \\ 0 & 40 & 200 \end{bmatrix} (10^{-5}kg)^2$$

Using formula 2.6 in matrix notation it is possible to write down the expression of the genetic and environmental covariance functions as:

$$G(t,s) = \boldsymbol{\phi}(t)^T * \boldsymbol{C}^G * \boldsymbol{\phi}(s) = \boldsymbol{\phi}(t)^T * \begin{bmatrix} 400 & 40 & 0 \\ 40 & 200 & 40 \\ 0 & 40 & 200 \end{bmatrix} * \boldsymbol{\phi}(s)$$

$$E(t,s) = \boldsymbol{\phi}(t)^T * \boldsymbol{C}^E * \boldsymbol{\phi}(s) = \boldsymbol{\phi}(t)^T * \begin{bmatrix} 400 & 40 & 0 \\ 40 & 200 & 40 \\ 0 & 40 & 200 \end{bmatrix} * \boldsymbol{\phi}(s)$$

Finally the variance of the residuals σ^2_{res} will be set to a value of 50. These are the true covariance functions that will be estimated. In figures 4.7 and 4.8 we can respectively see a colorplot and the graph of the genetic function (which is the same of the environmental one).

Using these parameters we can easily generate a sample of responses, simulating from the distributions of $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$. 150 different responses have been generated, which led to the estimation of 150 couples of covariance matrices, estimates of $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$.

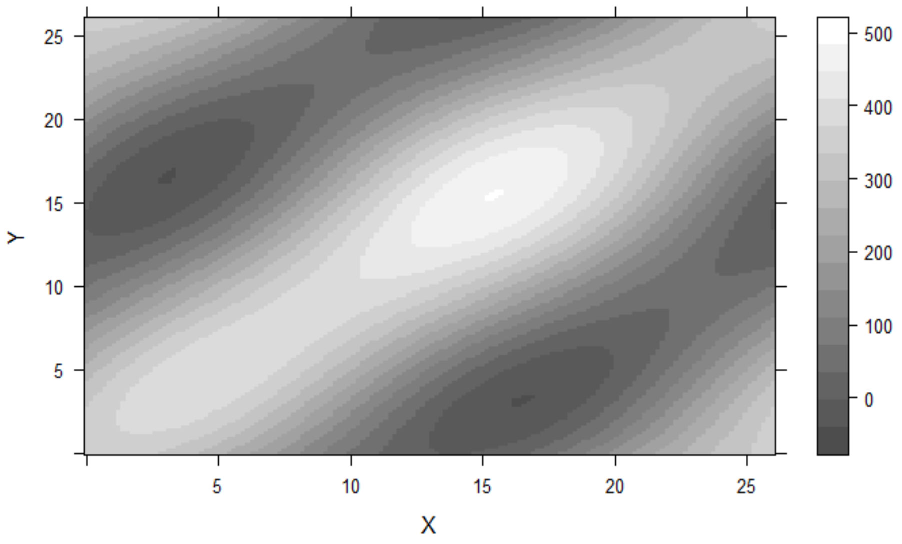

FIGURE 4.7: Genetic covariance matrix (colorplot)

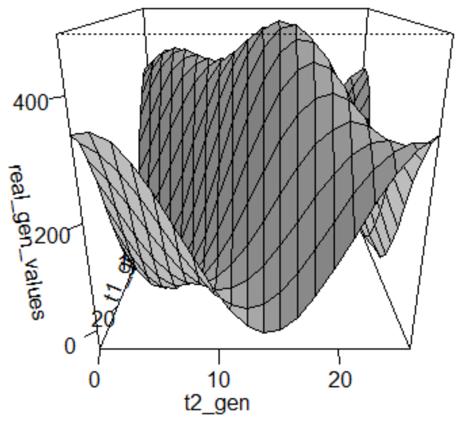

FIGURE 4.8: Genetic covariance matrix (axonometry)

In figure 4.9 we show the histograms of each of the elements of the genetic covariance matrix:

It is important to point out that in this subsection and the following we will concentrate more on the genetic covariance function, as was stated in previous sections, the estimation of G is the core of our analysis.

Notice now that since we are working with functions, histograms like the ones in figure 4.9 cannot tell much about the distribution of the functional sample. Some of the

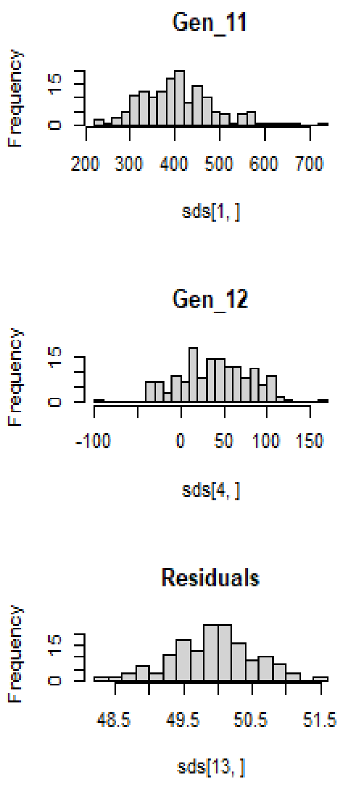

FIGURE 4.9: Histograms of the single elements of the sample of matrices

alternatives for our visualizations can be the functional box-plot of the first two eigenfunction.

Let us analyze the sample of first eigenfunctions $\{e_1^{(1)}, e_2^{(1)}, .., e_S^{(1)}\}$. In figure 4.11 is displayed a functional box-plot where the black central line is the median of the sample and the red one is the true first eigenfunction. The grey band is the functional IQR, and also some outliers are plotted. In functional box-plots the IQ curves are defined as the envelope of the 50% most central functions. The concept of centrality is determined after the definition of a functional distance in a suitable space (this will not be explained in detail). Notice how the true eigenvalue is very close to the median curve and falls inside the IQR.

The functional boxplot of the second eigenfunction is very similar to the latter, it is displayed in figure 4.11

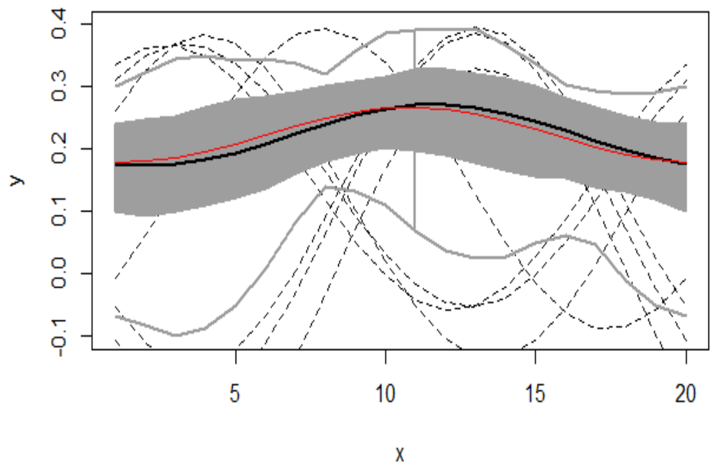

FIGURE 4.10: Functional boxplot of the sample, the true first eigenfunction is displayed in red, the median curve is the black one and the grey band corresponds to the IQR. Finally in dashed you can see the outliers

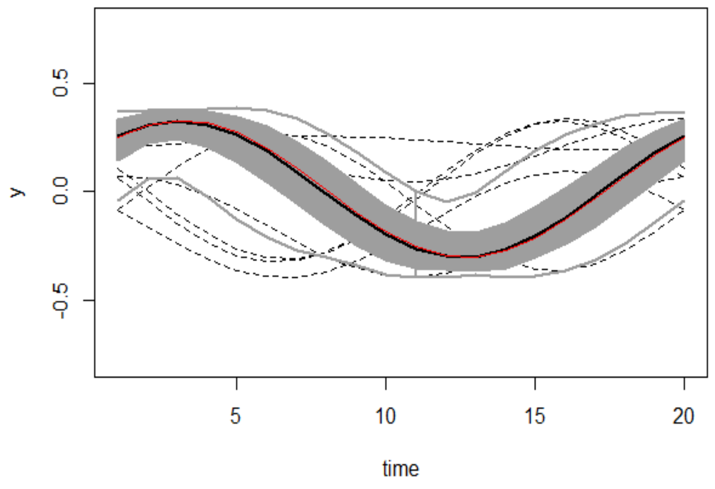

FIGURE 4.11: Functional boxplot of the sample, the true second eigenfunction is displayed in red, the median curve is the black one and the grey band corresponds to the IQR. Finally in dashed you can see the outliers

4.2.2 Functional Bootstrap Simulation

As anticipated in the introduction, in this part we will test our parametric bootstrapping strategy for making inference on the covariance function G and the most important eigenfunctions. In this section we will not analyze in detail the genetic implications of

our estimates since they are artificial, in the last chapter we will instead thoroughly discuss the genetic setting.
This section's simulation will be different from the previous one and will consist of the following steps:

1. Set the true parameters for the model (we will use the same model and parameters of *Functional Simulation*)
2. Simulate from these parameters just one response vector $\boldsymbol{y}$
3. Fit this response vector and extract the estimates for $\boldsymbol{C}^G$, $\boldsymbol{C}^E$ and σ^2_{res}, then compute the estimated covariance function $\hat{G}$
4. Use the estimated matrices and the residual variance to simulate a number S of responses $\{\boldsymbol{Y}_1, .., \boldsymbol{Y}_S\}$. From these responses then estimate again the covariance matrices and so the genetic covariance functions $\{G_1, .., G_S\}$

In the end of the simulation we have a sample of S covariance functions. From this we can extract the sample of i-th eigenvalues $\{e_1^{(i)}, .., e_S^{(i)}\}$.
Given $\{G_1, .., G_S\}$ and $\{e_1^{(i)}, .., e_S^{(i)}\}$ we will compute the point-wise and simultaneous confidence bands for the two, and underline the differences between the two approaches.

Pointwise confidence bands

Starting with the pointwise approach we show how to get to a pointwise bootstrap confidence band of confidence α. Let us compute a pointwise band for the genetic covariance function, fix $\bar{\boldsymbol{t}} \in [0, T]^2$. The goal is to find $c_{\alpha/2}$ and $c_{1-\alpha/2}$ s.t.:

$$P(c_{1-\alpha/2} \leq \hat{G}(\omega, \bar{\boldsymbol{t}}) - G(\bar{\boldsymbol{t}}) \leq c_{\alpha/2}) = 1 - \alpha$$

Since the distribution of the r.v. $\hat{G}(\omega, \bar{\boldsymbol{t}}) - G(\bar{\boldsymbol{t}})$ is not known we need to approximate this probability using the bootstrap sample $\{G_1, .., G_S\}$ performing the following approximation:

$$1-\alpha = P(c_{1-\alpha/2} \leq \hat{G}(\omega,\bar{\boldsymbol{t}}) - G(\bar{\boldsymbol{t}}) \leq c_{\alpha/2}) \cong P(c^b_{1-\alpha/2} \leq G_i(\omega,\bar{\boldsymbol{t}}) - \hat{G}(\omega,\bar{\boldsymbol{t}}) \leq c^b_{\alpha/2}) \tag{4.1}$$

Where $c^b_{\alpha/2}$ is the empirical quantile of order $\alpha/2$ of the distribution of $G_i(\omega,\bar{\boldsymbol{t}})-\hat{G}(\omega,\bar{\boldsymbol{t}})$. We omitted that the quantile $c^b_{\alpha/2}$ depends on the fixed point $\bar{\boldsymbol{t}}$ In the end our pointwise band will be in the form:

$$P\{\ \hat{G}(\omega,\bar{\boldsymbol{t}}) - c^b_{\alpha/2}(\bar{\boldsymbol{t}}) \leq G(\bar{\boldsymbol{t}}) \leq \hat{G}(\omega,\bar{\boldsymbol{t}}) - c^b_{1-\alpha/2}(\bar{\boldsymbol{t}})\ \} = 1-\alpha \tag{4.2}$$

In the following plots we show some axonometries and projections of the 3 surfaces of interest: the lower bound, the upper bound and the real surface (true covariance function). The first two surfaces are in grey while the true covariance function is in red. The goal is to detect in which points the true surface is not contained in the pointwise intervals.
It is challenging to visualize the entities we have just described. From figure 4.12 to 4.15 different views of the surfaces, the first two are upper and lower side projections, while the last two are axonometries from two different views. The projections turn out to be useful to visualize the extension of the area in which the confidence band does not contain the true function.

The same can be done for the eigenfunctions, the suitable approximation is the following:

$$1-\alpha = P(f_{1-\alpha/2} \leq \hat{e}(\omega,\bar{t}) - e(\bar{t}) \leq f_{\alpha/2}) \cong P(f^b_{\alpha/2} \leq e^{(i)}(\omega,\bar{t}) - \hat{e}(\omega,\bar{t}) \leq f^b_{\alpha/2}) \tag{4.3}$$

Where $f^b_{\alpha/2}$ is the empirical quantile of the bootstrap sample and $\bar{t}$ is a fixed element of $[0,T]$.

The pointwise confidence band for the first eigenfunction is displayed in figure 4.16, the true eigenfunction is in red while in dark grey the upper and lower limits. Notice how this kind of band does not include the true function.

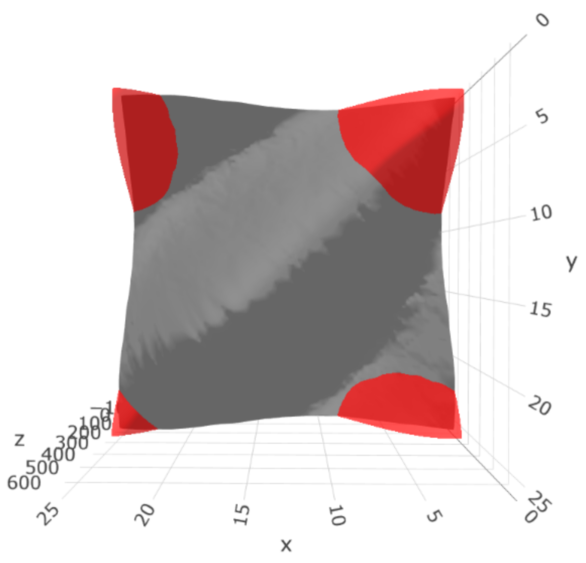

FIGURE 4.12: View from the top of the true covariance function in red and upper bound of the pointwise confidence band in grey

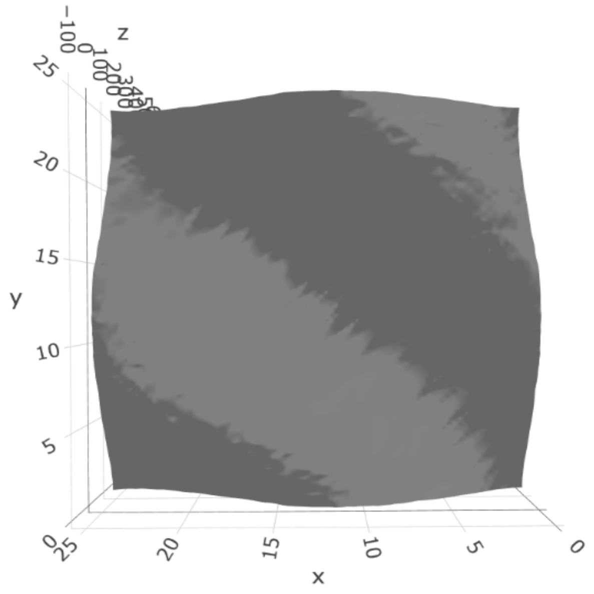

FIGURE 4.13: View from below of the true covariance function in red and of the lower bound of the pointwise band in grey

Simultaneous confidence bands

At this point we change approach and compute the simultaneous confidence bands. Starting from the covariance function, the initial intent is that of finding a band which will contain the entire covariance function with confidence α. Then, contrarily to the pointwise band, in this section we will find bootstrap coefficients which are independent of $\bar{\boldsymbol{t}}$ and $\bar{t}$. Let us start with the computation of the covariance band as usual, the goal

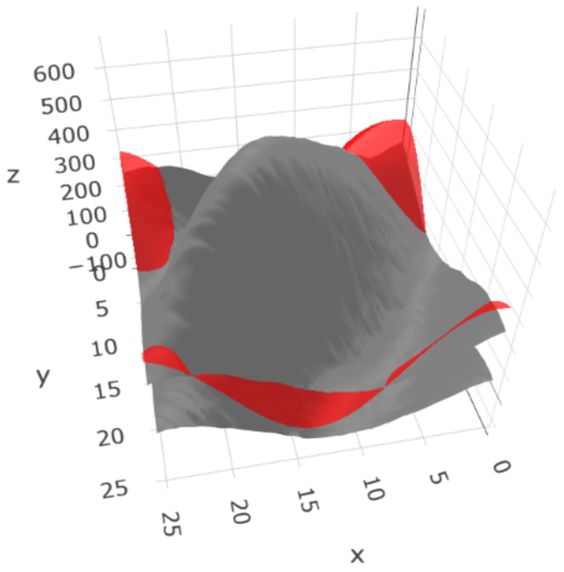

FIGURE 4.14: Axonometry of the upper and lower bound of the pointwise confidence band in grey and true covariance surfaces in red

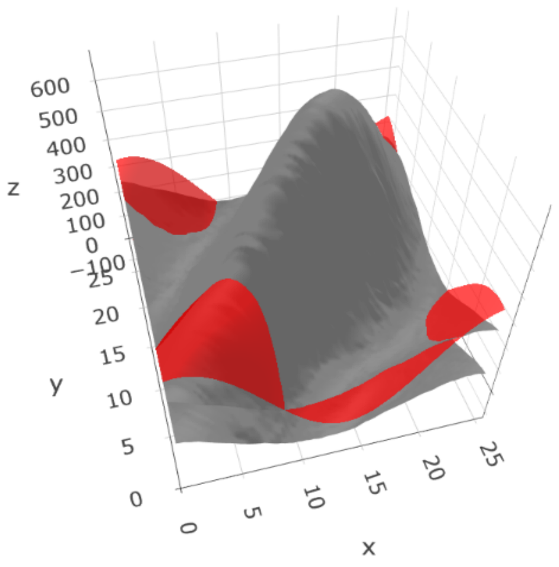

FIGURE 4.15: Axonometry of the upper and lower bound of the pointwise confidence band in grey and true covariance surfaces in red

is to find an approximation for the coefficient d_α in the expression:

$$P(\sup_{t\in[0,T]^2} |\hat{G}(\omega,\boldsymbol{t}) - G(\boldsymbol{t})| \leq d_\alpha) = 1 - \alpha$$

We can do this using the bootstrap sample, approximating the law of $\sup_{t\in[0,T]^2} |\hat{G}(\omega,\boldsymbol{t}) - G(\boldsymbol{t})|$ with the law of the random variable $\sup_{\boldsymbol{t}\in[0,T]^2} |G_i(\omega,\boldsymbol{t}) - \hat{G}(\omega,\boldsymbol{t})|$, implying:

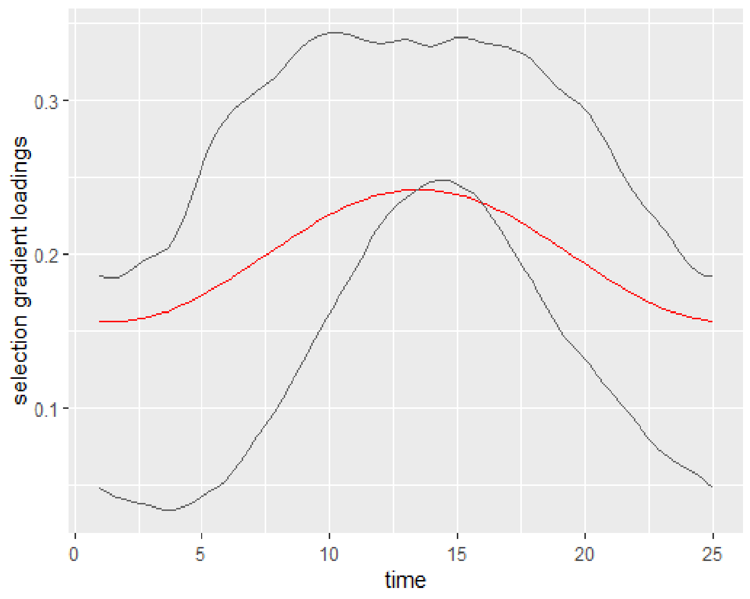

FIGURE 4.16: In grey the upper and lower bound of the pointwise band for the first eigenfunction, in red the true first eigenfunction

$$\begin{aligned} 1-\alpha &= P\big(\sup_{\boldsymbol{t}\in[0,T]^2} |\hat{G}(\omega,\boldsymbol{t}) - G(\boldsymbol{t})| \leq d_\alpha \big) \\ &\cong P\big(\sup_{\boldsymbol{t}\in[0,T]^2} |G_i(\omega,\boldsymbol{t}) - \hat{G}(\omega,\boldsymbol{t})| \leq d_\alpha^b \big) \end{aligned} \tag{4.4}$$

Through a couple of steps the simultaneous band comes out:

$$P\{ \, (\hat{G}(\omega,\boldsymbol{t}) - d_\alpha^b \leq G(\boldsymbol{t}) \leq \hat{G}(\omega,\boldsymbol{t}) + d_\alpha^b) \quad \forall \boldsymbol{t} \in [0,T]^2\} = 1-\alpha \tag{4.5}$$

In figure 4.16 we can see the true surface in red and the simultaneous band bounded by the grey surfaces. In this case the covariance is fully contained inside the band. Notice that the estimate $\hat{G}$ is not displayed to avoid having a frought representation. It is worth to mention that, since the coefficient d_α now doesn't depend on $\boldsymbol{t}$ the grey surfaces are equidistant from the estimated covariance graph.

It is now time to check if the simultaneous approach works also for the principal eigenfunction (the first one). In this case it is necessary to approximate l_α:

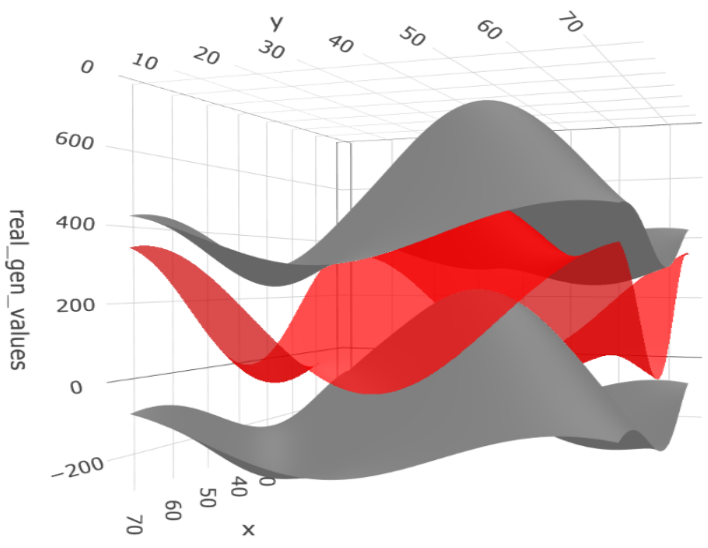

FIGURE 4.17: Upper, lower and true covariance surfaces in axonometry

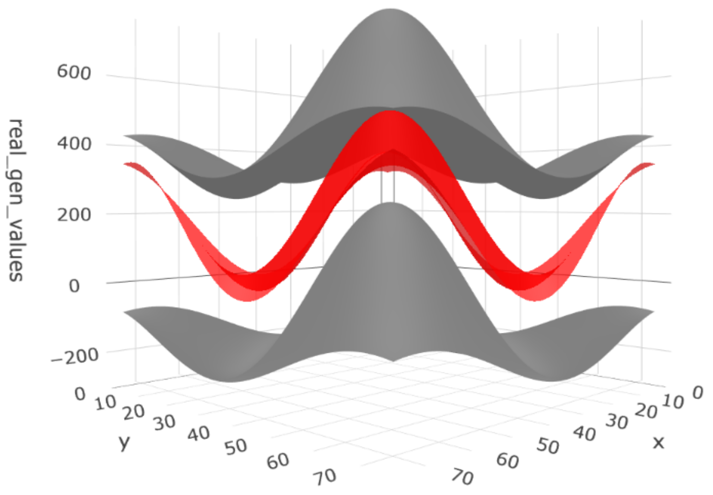

FIGURE 4.18: Upper, lower and true covariance surfaces in axonometry

$$P(\sup_{t\in[0,T]} |\hat{e}(\omega,\boldsymbol{t}) - e(\boldsymbol{t})| \leq l_\alpha) = 1-\alpha$$

Using the bootstrap sample it is possible to perform the following approximation:

$$\begin{aligned} 1-\alpha &= P(\sup_{t\in[0,T]} |\hat{e}(\omega,t)-e(t)| \leq l_\alpha) \\ &\cong P(\sup_{t\in[0,T]} |e_i(\omega,t)-\hat{e}(\omega,t)| \leq l_\alpha^b) \end{aligned} \tag{4.6}$$

And with a couple of steps we derive the following simultaneous band:

$$P\{ (\hat{e}^{(i)}(\omega,t)-l_\alpha^b \leq e(t) \leq \hat{e}^{(i)}(\omega,t)+l_\alpha^b) \quad \forall t\in[0,T] \} = 1-\alpha \tag{4.7}$$

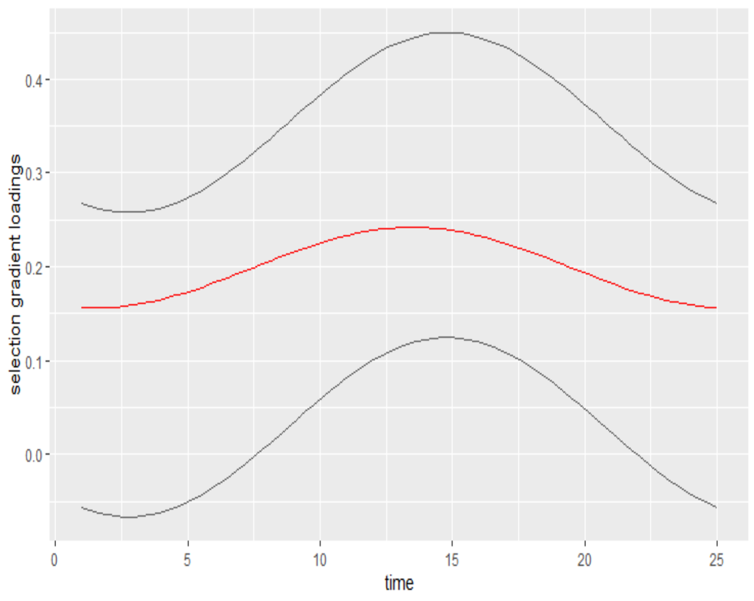

FIGURE 4.19: True first eigenvalue in red and the simultaneous band in grey

Figure 4.18 shows again the true eigenfunction in red and the borders of the band in black. Also in this case l_α^b doesn't depend on t so the band is symmetric w.r.t. the estimated eigenfunction $\hat{e}^{(1)}$.

Notice how the simultaneous approach turns out to be effective and realize the initial expectations on the inference band, while the pointwise interval does not account for the functional nature of the data. Given these positive inference results for our bootstrap strategy we can confidently apply this method to the case of *Triboleum Castaneum* described in the introduction.

Chapter 5

An application to *Triboleum Castaneum*

In this chapter we will apply the strategy described in the previous sections to the already introduced dataset `TRFUN25PUP4`. The reason why we used the structure of an existing dataset for the previous simulations is the additive relationship matrix $\boldsymbol{A}$. It was necessary to use real data to reproduce a plausible genetic correlation structure, all the other elements of the dataset could have been simulated (despite our use of some original covariates). Now instead we fit the real response and apply our bootstrap inference strategy to this problem. The result will then be compared to the analysis in Irwin and Carter, 2013 [15]; in this paper they just provide estimates for the genetic covariance and eigenfunction, we will confront them with our confidence bands and assess how reliable are these estimated values.
This chapter is organized in the following way:

- Section 5.1: In this section we describe the preprocessing steps, and the problem of choosing the right model complexity
- Section 5.2: Here we will fit the model with the one chosen in the previous section, display the estimated functions ($\hat{G}$ and eigenfunctions), and comment the fit from a genetic perspective. Lastly we compute the bootstrap confidence bands and assess whether our estimates are reliable or not

5.1 Model selection

In the simulation of chapter 4 we chose a given model in the form:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{Z}^G\boldsymbol{\alpha} + \boldsymbol{Z}^E\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

with the following distribution of the random vectors:

$$\boldsymbol{\alpha} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{A} \otimes \boldsymbol{C}^G)$$

$$\boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_N \otimes \boldsymbol{C}^E)$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_{res} * \boldsymbol{I}_n)$$

and arbitrarily choose a basis for the functional space to use in the simulation. The order of the chosen basis influences the dimension of the matrices $\boldsymbol{C}^G$ and $\boldsymbol{C}^E$, which in turn have en effect on the dimension of the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$.
In the case of fitting real data the problem of choosing the right model complexity is a debated issue and it is strictly related with the problems of singularity and convergence of the model. Let us first introduce the dataset plotting some of the growth curves in figure 5.1.

The kind of basis can be chosen qualitatively from the plot of the raw curves. A more complicated choice is the one of the complexity of the model, i.e. the order of the basis to use. The are two main (discordant) approaches to the identification of a suitable choice:

- The first approach is proposed by Barr et al. (2013) [3], which suggests to "keep it maximal", which means starting with the maximal model and going gradually diminuishing complexity until convergence problems and singularity issues stop to occur. The maximal model is a model with the highest possible order which is theoretically identifiable (i.e. highest possible number of random effect components).

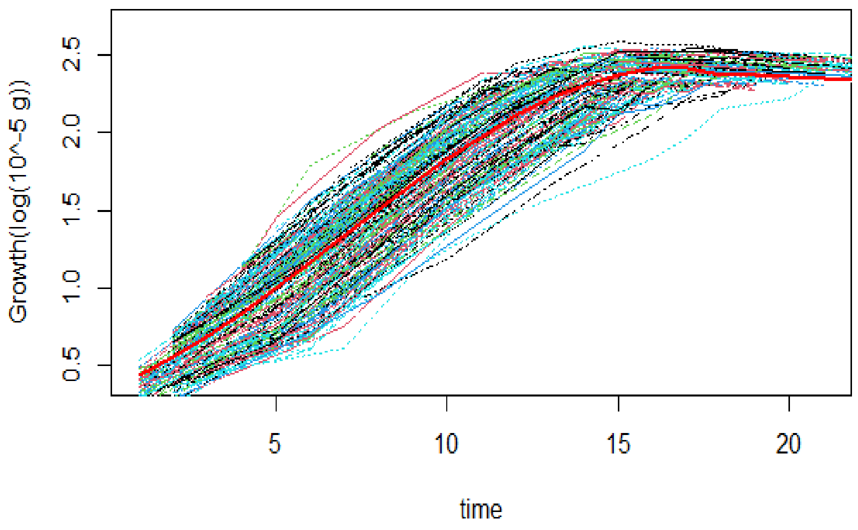

FIGURE 5.1: Growth curves and mean curve of *Triboleum Castaneum*

- The second approach goes in the opposite direction and suggests to start from the simplest possible model, increasing gradually its complexity. This strategy is supported by Matuschek et al., 2017 [22] and D. Bates, Kliegl et al., 2015 [5], which explain how the model should be maintained simple *a priori*.

When the model is overly complicated the resulting fitting is singular. A fitting is singular when the estimated covariance matrix is singular, this happens if the *template matrix* $\boldsymbol{T}$ (defined in the introduction of section 2.3) is degenerate, which means that the elements on the diagonal of the parameter $\boldsymbol{\theta}$ are zero or very close to zero. Singularity can happen also if the estimates of correlation between two random effect terms are +1 or -1.

For our fitting we have chosen a Legendre basis made by the first orthonormal elements, and the order of the model was decided following the procedure suggested by Barr et al., 2013 [3] using a modified BIC index to pick a certain model complexity. This index is very similar to the BIC except for the fact that the likelihood is substituted by the deviance, i.e the quantity minimized by the non-linear optimization step in `lme4`.

The choice fell to a 3 elements basis for the genetic variance where the constant is included, the basis is in the form:

$$\boldsymbol{\phi}(t) = \begin{bmatrix} 1/\sqrt{T} \\ \sqrt{\frac{2}{T}}cos(\frac{2\pi t}{T}) \\ \sqrt{\frac{2}{T}}sin(\frac{2\pi t}{T}) \end{bmatrix}$$

Where T is the period of the basis functions.

5.2 Parameter Estimation

The goal now is to display and comment the estimated quantities, which are the genetic covariance function $\hat{G}$, and the most important eigenfunctions. Needs however to be pointed out that the maximum number of eigenfunctions is limited by the order of fit chosen in the model selection step (in our case 3).

Let us start from the estimated matrix $\hat{\boldsymbol{C}}^G$ which is the following:

$$\hat{\boldsymbol{C}}^G = \begin{bmatrix} 0.007 & -0.014 & -0.013 \\ -0.014 & 0.044 & 0.041 \\ -0.013 & 0.041 & 0.039 \end{bmatrix}$$

So the estimated covariance function $\hat{G}$ will be:

$$\hat{G}(t,s) = \boldsymbol{\phi}(t)^T * \hat{\boldsymbol{C}}^G * \boldsymbol{\phi}(s) = \boldsymbol{\phi}(t)^T * \begin{bmatrix} 0.007 & -0.014 & -0.013 \\ -0.014 & 0.044 & 0.041 \\ -0.013 & 0.041 & 0.039 \end{bmatrix} * \boldsymbol{\phi}(s)$$

In figure 5.2 and 5.3 are displayed a colorplot and an axonometry to give the idea of the shape of the estimated function.

As you can see from the colorplot we have a pretty high correlation between early ages, and also a high correlation between higher ages. The covariance can be negative instead between high and low ages. G gets higher approaching upper-right part of the colorplot, this is probably given by estimation uncertainties at the border caused by the use of random regression models (Meyer & Kirkpatrick, 2005 [25]).

It is also important to have a look at the differences between G and the whole phenotypical covariance function P (which is the sum of G and E in figure 5.4). The proportion

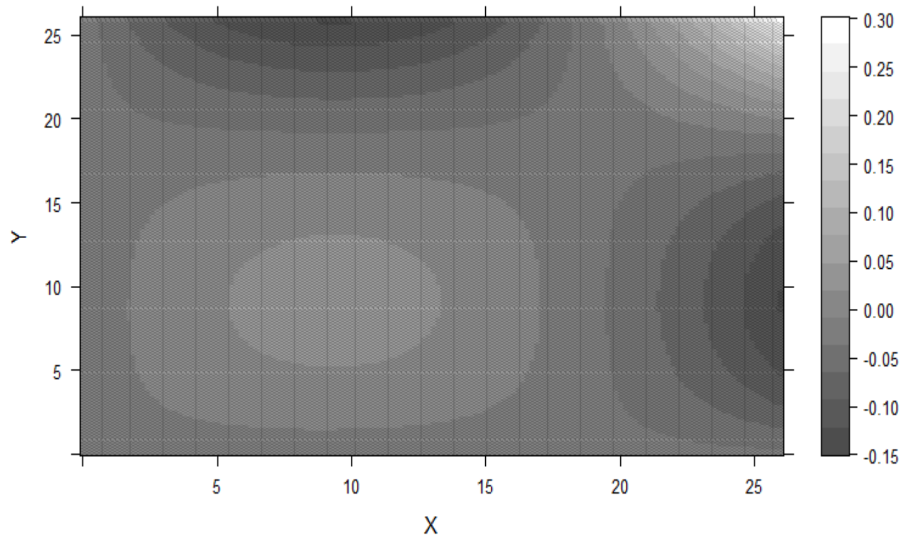

FIGURE 5.2: Estimated genetic covariance function

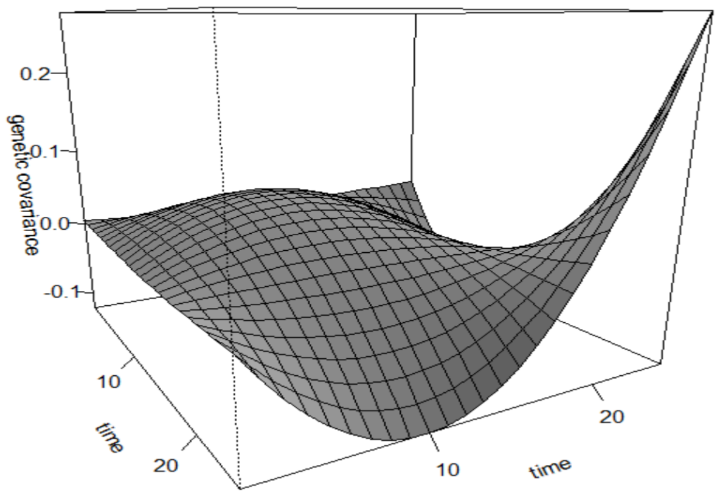

FIGURE 5.3: Estimated genetic covariance function

of P given by G is a clue over the heritability of this trait. From figure 5.3 and 5.4 it is possible to see that the portion of P constituted by G is large, concluding that heritability is consistently high.

To have a better idea of how small is the environmental contribution to the total variation we can show the estimated matrix $\hat{\boldsymbol{C}}^E$ and compare the coefficients with the one of $\hat{\boldsymbol{C}}^G$:

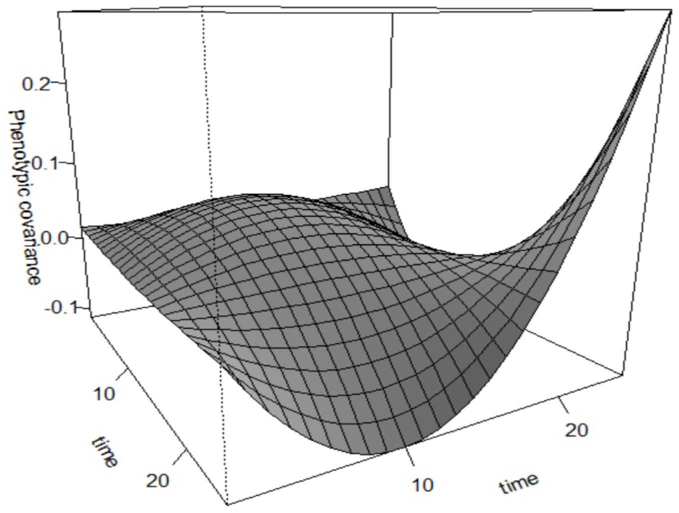

Figure 5.4: Estimated phenotypical covariance function

$$\hat{\boldsymbol{C}}^E = \begin{bmatrix} 1.31e-03 & -2.44e-03 & -1.37e-05 \\ -2.44e-03 & 5.94e-03 & -2.36e-05 \\ -1.38e-05 & -2.36e-05 & 1.86e-06 \end{bmatrix}$$

Notice for example how the element on the diagonal of $\hat{\boldsymbol{C}}^E$ are 10 or more times smaller then the correspondent elements in the genetic covariance matrix.
Let us now move to the analysis of the principal eigenfunctions of $\hat{G}$. The first three eigenfunctions account for the 99% of the total variance, we will analyze then these three functions. Evolutionary response is indeed commonly limited if one of the eigenfunctions accounts for a big part of the variability, in this case happens to have few significant evolutionary directions and many more with very small eigenvalues. In figure 5.5 there is a plot of the three main eigenfunctions, the first in solid, the second in dashed and the third in dot dashed.

The first eigenfunction (which accounts for the 96% of the variability) has positive loadings along the interval 1-17 days to become negative afterwards, this means that individuals with a high mass in the first period tend to lose mass more quickly in the last phase, before completing larval development, this last phase is called wandering phase. This information can be deducted also from the negative correlation of mass at early

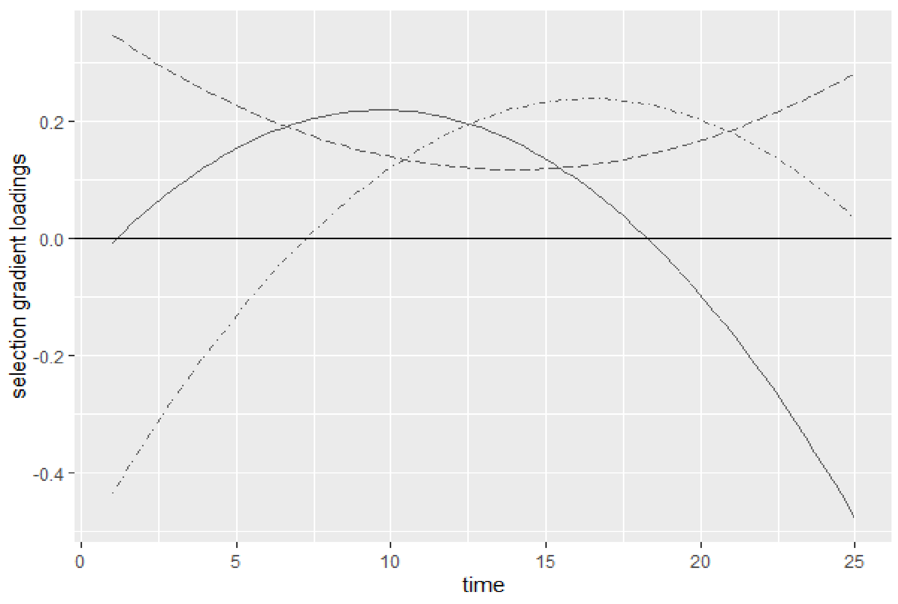

FIGURE 5.5: First three eigenfunctions:solid-firstEF, dashed-secondEF, dotdashed-third EF

stages with mass at later stages. The second eigenvalue is positive throughout all the period, it seems to reflect the isometric property of growth, individuals which have a high initial mass tend to maintain high body mass.

Notice that individuals that grow quickly decrease also their mass rapidly after the peak having reached it before the others; it could be interesting to consider curve registration techniques in our analysis to understand if high mass in the early period is negative correlated with the time before peaking or with mass in the later stages.

But let us grasp a deeper insight of the meaning of these first eigenfunctions plotting the mean of the dataset and its variation moving along the first and second main directions (figures 5.6 and 5.7).

As anticipated in the previous paragraph, the individuals with large components over the first eigenfunction tend to peak earlier in mass and the start their descent achieving a lower value than the mean at 25 days. This outcome is in accordance with Cheverud et al., 1983 [10], for which the maximal response curve alters only the curve height and not its shape. The second component which accounts for roughly the 3% of the variability seems to represent an overall mass component, individuals with a high score on EF2 have generally an higher value of body mass throughout the whole 25 days period. Finally, the third eigenfunction which accounts for very little of the variability, does not contribute

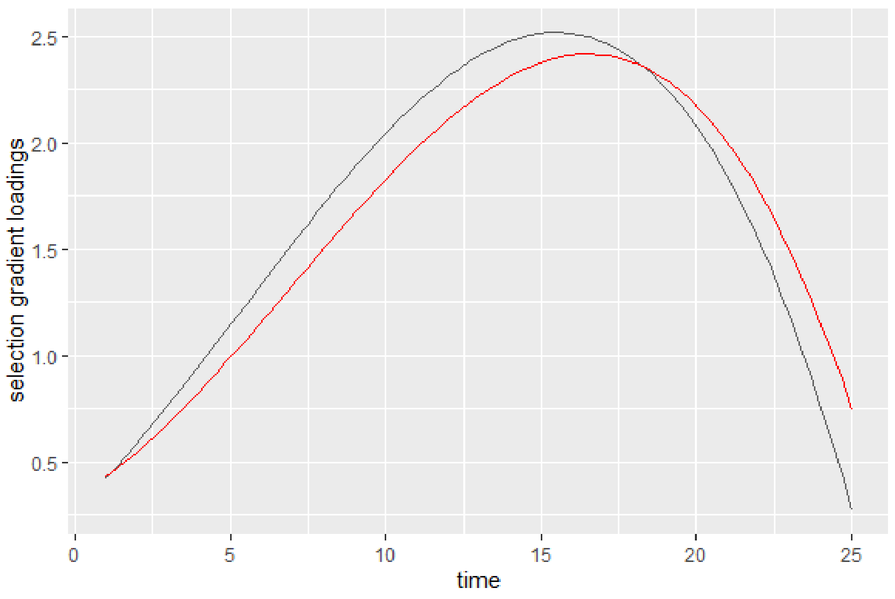

Figure 5.6: In solid is displayed the deviation from the mean moving along the first principal component

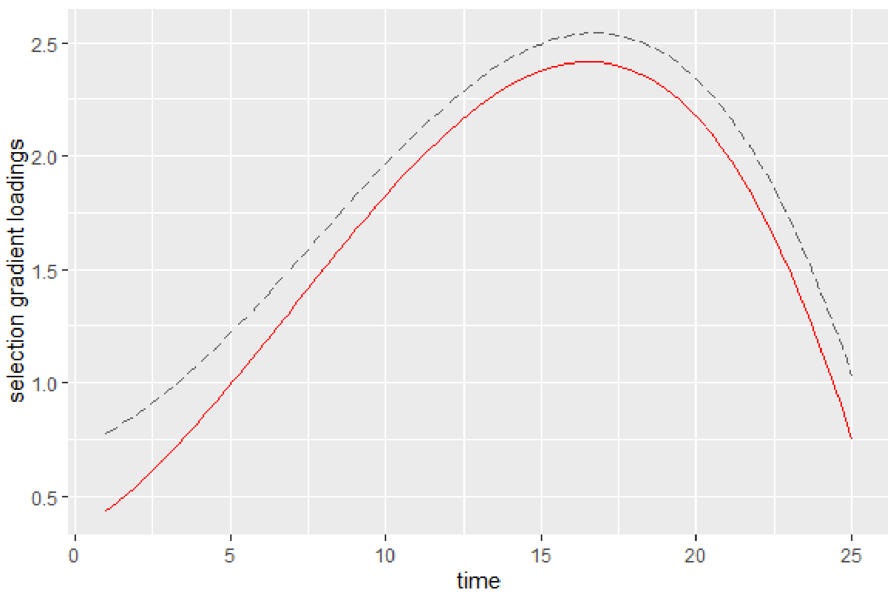

Figure 5.7: In dashed the deviation moving towards the second PC

on any change in the position of the peak, but allows individuals that have an earlier low mass raise their value at peak to a higher score than the mean one. The estimation of the principal eigenfunctions allows us to detect the evolutionary constraints. These correspond to directions which have small components on the principal eigenfunctions. Selection gradients along these directions cause a small variation in the mean phenotypic trait during selection.

In the last step let us compute the simultaneous confidence bands and visualize the bands and the estimated quantities. In figures 5.8 and 5.9 we can see only one side of the band and the estimate for a light and understandable representation, needs to be said that the other boundary is simply symmetric w.r.t. the estimate in red.

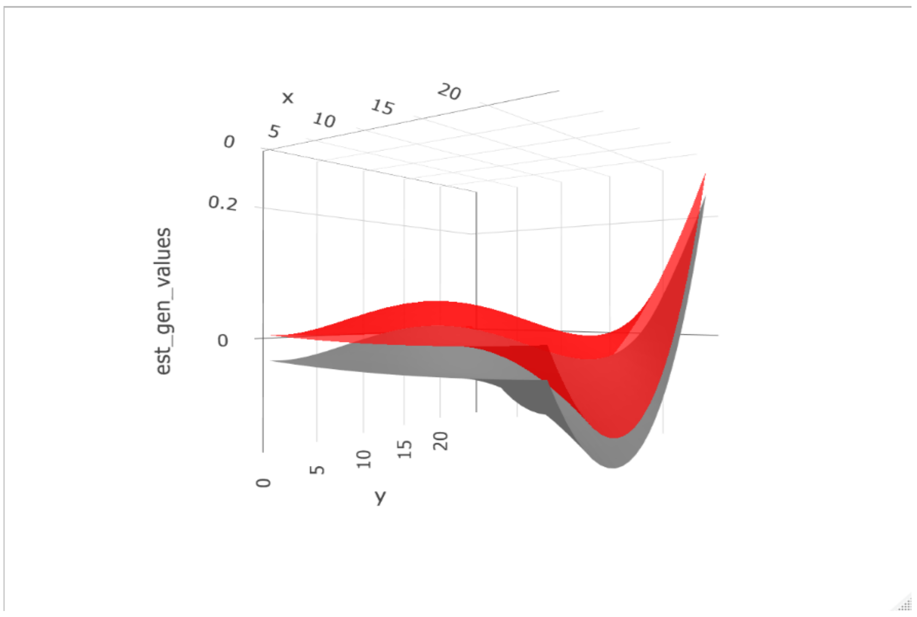

FIGURE 5.8: Covariance simultaneous confidence interval, one side for better displayment

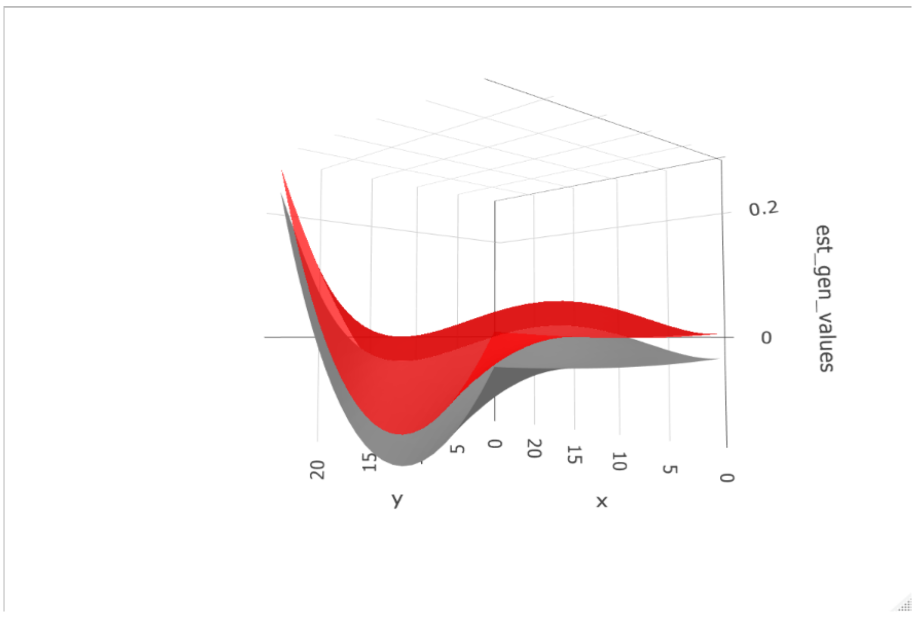

FIGURE 5.9: Covariance simultaneous confidence interval, one side for better displayment

In our last part of the work we analyze the first two eigenfunctions of $\hat{G}$, we do not analyze the third one since its associated percentage of variability it is small.

Notice figure 5.10 where the first eigenfunction is displayed, the confidence band is narrow along the whole function, this does not happen for the second eigenfunction in figure 5.11, where the band seems to be way larger (approximately 0.15, almost 3 times bigger than before). This result is somehow encouraging since the better precision is given for the eigenfunction corresponding to the higher variability.

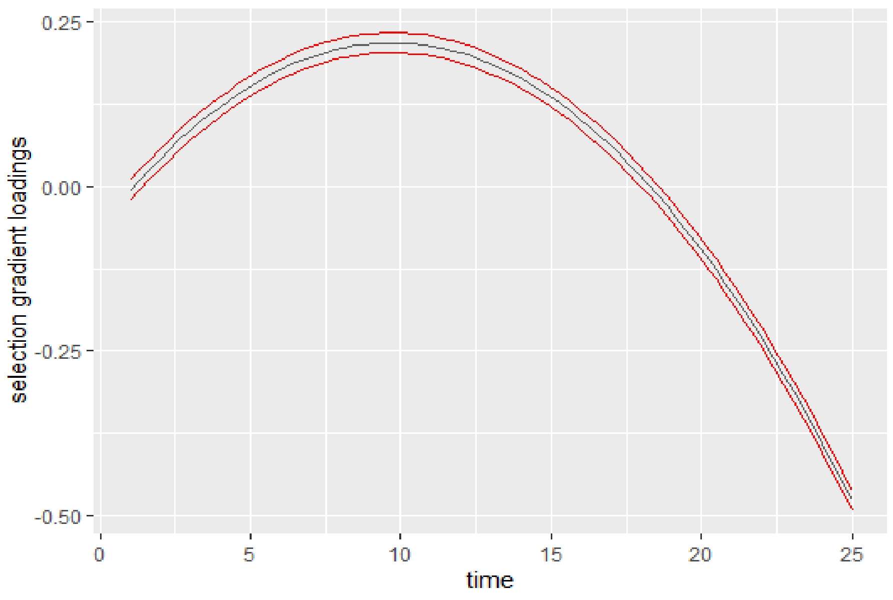

FIGURE 5.10: Simultaneous confidence interval for the first eigenfunction

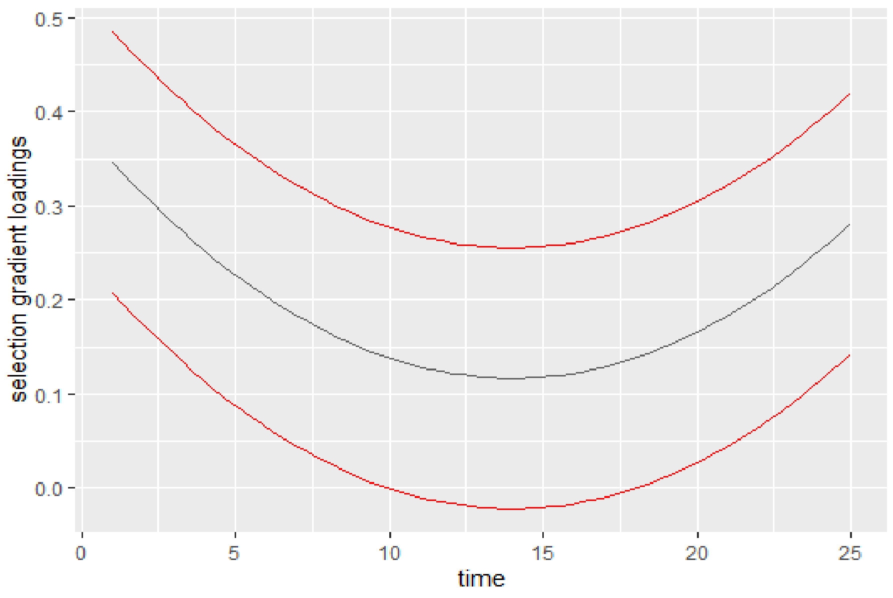

FIGURE 5.11: Simultaneous confidence interval for the second eigenfunction

Chapter 6

Conclusions and future directions

Let us now summarize the initial goals of this work and how they were tackled. The first one was the necessity to set up a routine to fit functional models, which corresponds to a fitting algorithm supporting a particular covariance structure of the random effect vector and an arbitrary number of covariates. This has been done through a reparameterization strategy and tested on multivariate models first with satisfactory results. The second goal regarded the construction of an inference strategy for covariance and eigenfunction estimation. Once set up the algorithm for the functional case we have performed a simulation to see if the proposed simultaneous band strategy was a valuable one and how it was different from a point-wise approach. The simulation results have demonstrated that `fme`'s estimates are consistent with the true parameters since the bootstrap simultaneous approach proposed correct outcomes for the inference problem. We applied the method to a real case study on *Triboleum Castaneum* discussing the insights coming from the estimation of G and of its main eigenfunctions. The results on the estimates have been consistent with other studies on the same dataset [15], while the inference strategy gave us a useful insight on the distribution of the functional estimates. The estimation of the principal eigenvectors shed also a light on the evolutionary constraints, it is indeed possible to detect these constraints, they correspond to selection gradients along directions which have small components on the principal eigenfunctions.

There are three main directions of improvement for this work: the first one regards the code of `fme` and its optimization, the second one is about a theoretical justification of the bootstrap approach and the last one is about a more detailed genetic analysis of *Triboleum Castaneum*. `fme` works with a high number of covariates and several random effect terms, however, it can be improved on handling factors and some testing can be made to ensure its easy usability. The work has confirmed how the bootstrap approach can be a valuable one for inference purposes. This result suggests some directions of improvements: a theoretical guarantee of convergence of the estimated genetic covariance function towards the true one would ensure a formal justification for the bootstrap method, maybe suggesting also a different statistic. The last direction is about a the use of registration [10] and warping [29] techniques for a clearer interpretation of covariance functions.

Appendix A

Code

In this appendix we show part of the code used for the implementation. This code is available in the following git-hub repository `quant-genetic-thesis` [6]. The first example is the `fit_me` function which is the main function for the fit of the model. Supports models in the form of the equation 2.3, i.e. with a general covariance structure for the random effect vector. The documentation for this function can be found below. Some external packages have been used to ease the computation, the package `Matrix` contains tools for operations between sparse matrices. The random effect matrix $\boldsymbol{Z}$, the matrix $\boldsymbol{A}$ have indeed a sparse structure. Also the package `pedigreemm` is used for some methods for the extraction of the suitable additive relationship matrix for our data.

```
library(Matrix)
library(lme4)
library(pedigreemm)
library(docstring)
library(numDeriv)

fit_me = function(formula,
                  data,
                  label,
                  pedigree = NULL,
                  n = 1,
                  optimizer = "nloptwrap")
    {
      #' Fit mixed effect model
      #'
      #' Fits a mixed effect model with a non block covariance matrix
```

```
  #' @param formula A language type formula for the model to fit
  #' @param data A dataframe with columns the labels included in the formula
  #' @param label Given that the dataset "data" is attached, label is the column
  #' relative to the pedigree's label
  #' @param pedigree A pedigree having as labels the column "label" of the dataframe
  #' @param n The number of terms inside the mixed effect term relative to label
  #' @param optimizer The wrapper to use in the non-linear optimization step
  #' @return Returns a fitted lme4 object

  f = lFormula(formula,
               data = data,
               control = lmerControl(optimizer = optimizer,
                                     restart_edge = TRUE,
                                     boundary.tol = 0.01))

  # Extract the relationship matrix A from the pedigree
  pos = unique(label)[order(unique(label))]
  A = if(is.null(pedigree)) diag(length(unique(label)))else getA(pedigree)[pos,pos]

  I = as(diag(n), "dgCMatrix" )
  nrest = dim(f$reTrms$Zt)[1]-dim(A)[1]*n

  Lt = chol(A)
  Mt = kronecker(Lt,I)

  # Modify the upper part of the matrix Zt
  f$reTrms$Zt[1:(n*(dim(A)[1])), ] = as(Mt%*%(f$reTrms$Zt[1:(n*(dim(A)[1])), ]),
                                        "dgCMatrix")

  #Optimization steps
  devfun <- do.call(mkLmerDevfun,f)
  opt <- optimizeLmer(devfun,
                      control = lmerControl(optimizer = optimizer,
                                            restart_edge = TRUE,
                                            boundary.tol = 0.01,
                                            calc.derivs = FALSE))

  fit <- mkMerMod(environment(devfun),
                  opt,
                  f$reTrms,fr = f$fr)

  return(fit)
}
```

In the following we show the complete code of the bootstrap simulation, also the code for plots and visualization is available below.

```
        library(MASS)
library(ggplot2)
library(ggpubr)
library(Metrics)
library(lme4)
library(pedigreemm)
library(mvnfast)
library(plotly)

TRFUN25PUP4 <- read.delim("TRFUN25PUP4.DAT", header=FALSE)
names(TRFUN25PUP4)<-c("id","sire","dam","trait","x")

df = TRFUN25PUP4

# set the number in each random effect vector
n_of_terms_gen = 3
n_of_terms_nongen = 3

iterations = 400

# remapping of x in [-1,1]
df$x = df$x*(1/13)-1
head(df)

# matrix of the fourier basis
Z = basis_matrix(df$x, r = 3, type = "legendre")
df = cbind(df,Z)
names(df)[6:8] = c("z1", "z2", "z3")
head(df)

# pedigree costruction from the dataset df
posFirstUniqueId = which((!duplicated(df$id))==TRUE)

sr = c( rep(NA, 100),
        rep(NA, 9900),
        rep(NA, 1500))
sr[unique(df$id)] = (df$sire)[posFirstUniqueId]

dm = c( rep(NA, 100),
        rep(NA, 9900),
        rep(NA, 1500))
dm[unique(df$id)] = (df$dam)[posFirstUniqueId]

lbl = c(1:11500)
pedigree = pedigree(sr, dm, lbl)

# get the matrix A
position = unique(df$id)[order(unique(df$id))]
A = getA(pedigree)[position,position]
dim(A)

df$id2 = df$id

# create the matrices from which to generate data

# small genetic covariance matrix
small_mat_gen = matrix(c(400, 40, 0,
                         40, 200, 40,
                         0, 40, 200), nrow = 3, byrow = T)
small_mat_gen

# small environmental covariance
small_mat_env = matrix(c(400, 40, 0,
                         40, 200, 40,
                         0, 40, 200), nrow = 3, byrow = T)
small_mat_env

# variance of the error sigma^2
s_2_residuals = 25



# big genetic covariance
c_gen = as(kronecker(A, small_mat_gen),"dgCMatrix")

# big env covariance
c_env = as(kronecker(diag(length(unique(df$id))), small_mat_env),"dgCMatrix")

# general covariance
c = as(bdiag(c_gen,c_env),"dgCMatrix")
dim(c)

# length of the vector u
ul = n_of_terms_gen*length(unique(df$id)) + n_of_terms_nongen*length(unique(df$id))
```

```
n = dim(df)[1]

# compute the fixed effect
g = lm(trait ~ -1+ df$z1+ df$z2+ df$z3, data = df)

length_error_1 = n
mu_u = rep(0, ul)
mu_e = rep(0, length_error_1)
c_e = as(s_2_residuals*diag(length_error_1), "dgCMatrix")

f = df$z1*g$coefficients[1] + df$z2*g$coefficients[2]+ df$z3*g$coefficients[3]

u = as.vector(rmvn(1, mu_u, c, ncores = 2))
e = as.vector(rmvn(1, mu_e, c_e, ncores = 2))
form = lFormula(df$trait ~ -1 + df$z1 +df$z2 + df$z3 +
(-1 + df$z1 +df$z2 + df$z3 |df$id) + (-1 + df$z1 +df$z2+ df$z3 |df$id),
                data = df)
df$trait = as.vector(f + t(form$reTrms$Zt)%*%u + e)

formula = df$trait ~ -1 + df$z1 + df$z2 + df$z3 +
( -1 + df$z1 + df$z2 + df$z3 |df$id) + ( -1 + df$z1 + df$z2 + df$z3  |df$id2)

fit <- fit_me( formula,
               data = df,
               label = df$id,
               pedigree = pedigree,
               n = n_of_terms_gen,
               optimizer = 'nloptwrap')

extract_variance = function(fit_object)
{
  result = as.matrix(as.data.frame(VarCorr(fit_object))["vcov"])
  return(result)
}

# extract the estimate genetic and env. variance
# genetic
data_gen = extract_variance(fit)
vec_gen = c(data_gen[1],
            data_gen[4],
            data_gen[5],
            data_gen[4],
            data_gen[2],
            data_gen[6],
            data_gen[5],
            data_gen[6],
            data_gen[3])
est_mat_gen = matrix(vec_gen, nrow = 3)
save(est_mat_gen, file = "est_mat_gen.Rdata")

# env
data_env = extract_variance(fit)
vec_env = c(data_env[1+6],
            data_env[4+6],
            data_env[5+6],
            data_env[4+6],
            data_env[2+6],
            data_env[6+6],
            data_env[5+6],
            data_env[6+6],
            data_env[3+6])
real_mat_env = matrix(vec_env, nrow = 3)

# get the matrix A
position = unique(df$id)[order(unique(df$id))]
A = getA(pedigree)[position,position]
dim(A)

# big genetic covariance
c_gen = as(kronecker(A, est_mat_gen),"dgCMatrix")

# big env covariance
c_env = as(kronecker(diag(length(unique(df$id))), real_mat_env),"dgCMatrix")

# general covariance
c = as(bdiag(c_gen,c_env),"dgCMatrix")
dim(c)

# length of the vector u
ul = n_of_terms_gen*length(unique(df$id)) + n_of_terms_nongen*length(unique(df$id))

iterations = iterations

n = dim(df)[1] # potrebbe essere modificato dopo

sds = matrix(0, nrow = 0.5*n_of_terms_gen*(n_of_terms_gen+1)+0.5*
(n_of_terms_nongen)*(n_of_terms_nongen+1)+1, ncol = iterations)
```

```
g = lm(trait ~ -1+df$z1+df$z2+df$z3, data = df)

n = dim(df)[1]
length_error_1 = n
mu_u = rep(0, ul)
mu_e = rep(0, length_error_1)
c_e = as(s_2_residuals*diag(length_error_1), "dgCMatrix")

f = df$z1*g$coefficients[1] + df$z2*g$coefficients[2] + df$z3*g$coefficients[3]

df$id2 = df$id

# progress bar
total <- iterations
pb <- txtProgressBar(min = 0, max = total, style = 3)

system.time(
  for( i in 1:iterations)
  {
    # generate the random effect vector composed of alpha and gamma
    u = as.vector(rmvn(1, mu_u, c, ncores = 2))

    # the error vector
    e = as.vector(rmvn(1, mu_e, c_e, ncores = 2))

    form = lFormula(df$trait ~ -1 + df$z1 +df$z2+df$z3+
    (-1 + df$z1 + df$z2 +df$z3|df$id) + (-1 + df$z1 + df$z2 +df$z3|df$id),
                    data = df)

    # 1st version: in which i have as many generated points as in the dataset
    df$trait = as.vector(f + t(form$reTrms$Zt)%*%u + e)

    # # store it to analyze it later
    # responses = cbind(responses, df$trait)

    # now we fit the model, the result should give out the same cov matrix as before
    fm1 <- fit_me( df$trait ~ -1 + df$z1 +df$z2+df$z3+
    (-1 + df$z1 + df$z2 +df$z3|df$id) + (-1 + df$z1 + df$z2 +df$z3|df$id),
                   data = df,
                   label = id,
                   pedigree = pedigree,
                   n = n_of_terms_gen,
                   optimizer = 'nloptwrap')# nloptwrap, Nelder_mead, bobyqa
    samp = as.matrix(as.data.frame(VarCorr(fm1))["vcov"])
    sds[,i] = samp

    Sys.sleep(0.1)
    setTxtProgressBar(pb, i)
  }
)

sample_27 = sds
save(sample_27, file = "sample_27.Rdata")

# Pointwise confidence bands for covariance function and eigenvectors
fine_grid = 25

# covariance function

# real covariance function
real_gen_cov_func = creator_gen_cov_function(small_mat_gen)
t1_gen <- t2_gen <- seq(-1, 1, length= fine_grid)
real_gen_values <- outer(t1_gen, t2_gen, real_gen_cov_func)

# estimated genetic covariance function
est_gen_cov_fun = creator_gen_cov_function(est_mat_gen)
# trial
est_gen_cov_fun(c(0,0.5, 0.2),c(0,0.5, 0.2))

est_gen_values <- outer(t1_gen, t2_gen, est_gen_cov_fun)

x11()
persp(t1_gen,
      t2_gen,
      est_gen_values,
      theta = 30,
      phi = 15,
      shade = 0.45,
      col = "aquamarine",
      ticktype = "detailed",
      nticks = 4)

# compute the lower and upper for each point

# ar will be the 3D matrix with all the values of the functions in the grid
zeros = rep(0,fine_grid*fine_grid*iterations)
```

```
ar_values_cov <- array(zeros, c(fine_grid, fine_grid, iterations))
ar_differences_cov <- array(zeros, c(fine_grid, fine_grid, iterations))

ar_values_eig <- array(zeros, c(fine_grid, iterations))
ar_differences_eig <- array(zeros, c(fine_grid, iterations))

# compute the functions
for ( i in 1:iterations)
{
  data_gen = sds[,i]
  vec_gen = c(data_gen[1],
              data_gen[4],
              data_gen[5],
              data_gen[4],
              data_gen[2],
              data_gen[6],
              data_gen[5],
              data_gen[6],
              data_gen[3])
  mat_gen = matrix(vec_gen, nrow = 3)

  gen_cov_func = creator_gen_cov_function(mat_gen)

  t1 <- t2 <- seq(-1, 1, length = fine_grid)
  ar_values_cov[,,i] =  outer(t1, t2, gen_cov_func)
  ar_differences_cov[,,i] = ar_values_cov[,,i] - est_gen_values

  # eigenfunctions
  est_eigenfunction = eigen(est_gen_values)$vectors[,1]

  ar_values_eig[, i] = eigen(ar_values_cov[,,i])$vectors[,1]
  ar_differences_eig[, i] = ar_values_eig[, i]-est_eigenfunction
}

contenuto_cov = rep(0,fine_grid*fine_grid*2)
contenuto_eig = rep(0,fine_grid*2)
ul_bound_cov <- array(contenuto_cov, c(fine_grid, fine_grid, 2))
ul_bound_eig <- array(contenuto_eig, c(fine_grid, 2))

for ( i in 1:fine_grid)
{
  for (j in 1:fine_grid)
  {
    upper_lower_cov = quantile(ar_differences_cov[i, j,],
                               probs = c(0.025, 0.975))
    ul_bound_cov[i, j,] = c(est_gen_values[i, j]-upper_lower_cov[2],
                            est_gen_values[i, j]-upper_lower_cov[1])
  }
  upper_lower_eig = quantile(ar_differences_eig[i,],
                             probs = c(0.025, 0.975))
  ul_bound_eig[i,] = c(est_eigenfunction[i] -upper_lower_eig[2],
                       est_eigenfunction[i] -upper_lower_eig[1])
}

# new plotting
x11()
fig <- plot_ly(showscale = FALSE)
fig <- fig %>% add_surface(z = ~real_gen_values)
fig <- fig %>% add_surface(z = ~ul_bound_cov[,,1],
opacity = 0.98, colorscale = list(c(0, 0), c("tan", "blue")))
fig <- fig %>% add_surface(z = ~ul_bound_cov[,,2],
opacity = 0.98, colorscale = list(c(0, 0), c("tan", "blue")))
fig

real_eigenfunction = eigen(real_gen_values)$vectors[,1]
plot(1:20, -real_eigenfunction, type = 'l', ylim = c(-0.5,.2))
points(1:20, est_eigenfunction, col = "green", type = 'l')
points(1:20, ul_bound_eig[,1], col = "red", type = 'l')
points(1:20, ul_bound_eig[,2], col = "red", type = 'l')

# simultaneous confidence interval

vec_max_diff_cov = rep(0, iterations)
vec_max_diff_eig = rep(0, iterations)

for(k in 1:iterations)
{
  vec_max_diff_cov[k] = max(abs(ar_differences_cov[,,k]))
  vec_max_diff_eig[k] = max(abs(ar_differences_eig[,k]))
}

length(vec_max_diff_eig)
hist(vec_max_diff_cov, breaks = 100)
hist(vec_max_diff_eig, breaks = 100)


quantile_cov = quantile(vec_max_diff_cov, probs = 0.975)
```

```
quantile_eig_sim = quantile(vec_max_diff_eig, probs = 0.975)

ul_bound_cov_sim <- array(contenuto_cov, c(fine_grid, fine_grid, 2))
ul_bound_eig_sim <- array(contenuto_eig, c(fine_grid, 2))

ul_bound_cov_sim[,,1] = est_gen_values-quantile_cov
ul_bound_cov_sim[,,2] = est_gen_values+quantile_cov

ul_bound_eig_sim[, 1] = est_eigenfunction-quantile_eig_sim
ul_bound_eig_sim[, 2] = est_eigenfunction+quantile_eig_sim

# plotting
x11()
fig <- plot_ly(showscale = FALSE)
fig <- fig %>% add_surface(z = ~real_gen_values)
fig <- fig %>% add_surface(z = ~ul_bound_cov_sim[,,1],
opacity = 0.98, colorscale = list(c(0, 0), c("tan", "blue")))
fig <- fig %>% add_surface(z = ~ul_bound_cov_sim[,,2],
opacity = 0.98, colorscale = list(c(0, 0), c("tan", "blue")))
fig

real_eigenfunction = eigen(real_gen_values)$vectors[,1]
plot(1:20,- real_eigenfunction, type = 'l', ylim = c(-0.5,.5))
points(1:20, est_eigenfunction, col = "green", type = 'l')
points(1:20, ul_bound_eig_sim[,1], col = "red", type = 'l')
points(1:20, ul_bound_eig_sim[,2], col = "red", type = 'l')
```

References

[1] LG de Albuquerque and Karin Meyer. Estimates of covariance functions for growth from birth to 630 days of age in nelore cattle. *Journal of animal science*, 79(11): 2776–2789, 2001.

[2] AV Badyaev and TE Martin. Individual variation in growth trajectories: phenotypic and genetic correlations in ontogeny of the house finch (carpodacus mexicanus). *Journal of Evolutionary Biology*, 13(2):290–301, 2000.

[3] Dale J Barr, Roger Levy, Christoph Scheepers, and Harry J Tily. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of memory and language*, 68(3):255–278, 2013.

[4] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *arXiv preprint arXiv:1406.5823*, 2014.

[5] Douglas Bates, Reinhold Kliegl, Shravan Vasishth, and Harald Baayen. Parsimonious mixed models. *arXiv preprint arXiv:1506.04967*, 2015.

[6] Marco Biazzo. quant-genetic-thesis, 8 2021. URL `https://github.com/marcobiazzo/quant-genetic-thesis`.

[7] Mats Björklund. Variation in growth in the blue tit (parus caeruleus). *Journal of Evolutionary Biology*, 10(2):139–155, 1997.

[8] Terrance P Callanan and David A Harville. *Some new algorithms for computing maximum likelihood estimates of variance components*. Iowa State University. Department of Statistics. Statistical Laboratory, 1989.

[9] JL Campo and Carmen Rodriguez. Experimental comparison of methods for simultaneous selection of two correlated traits in tribolium. 2. index selection and independent culling levels: a replicated single generation test. *Génétique sélection évolution*, 18(4):437–446, 1986.

[10] James M Cheverud, JJ Rutledge, and William R Atchley. Quantitative genetics of development: genetic correlations among age-specific trait values and the evolution of ontogeny. *Evolution*, pages 895–905, 1983.

[11] Jeffrey Conner and Sara Via. Natural selection on body size in tribolium: possible genetic constraints on adaptive evolution. *Heredity*, 69(1):73–83, 1992.

[12] Antonio Cuevas. A partial overview of the theory of statistics with functional data. *Journal of Statistical Planning and Inference*, 147:1–23, 2014. ISSN 0378-3758. doi: https://doi.org/10.1016/j.jspi.2013.04.002. URL https://www.sciencedirect.com/science/article/pii/S0378375813000748.

[13] Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*. Number 1. Cambridge university press, 1997.

[14] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[15] KK Irwin and PA Carter. Constraints on the evolution of function-valued traits: A study of growth in tribolium castaneum. *Journal of evolutionary biology*, 26(12): 2633–2643, 2013.

[16] CG Khatri and C Radhakrishna Rao. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhyā: the Indian journal of statistics, series A*, pages 167–180, 1968.

[17] Joel G Kingsolver, Richard Gomulkiewicz, and Patrick A Carter. Variation, selection and evolution of function-valued traits. *Microevolution rate, pattern, process*, pages 87–104, 2001.

[18] Mark Kirkpatrick and Nancy Heckman. A quantitative genetic model for growth, shape, reaction norms, and other infinite-dimensional characters. *Journal of mathematical biology*, 27(4):429–450, 1989.

[19] Mark Kirkpatrick and David Lofsvold. Measuring selection and constraint in the evolution of growth. *Evolution*, 46(4):954–971, 1992.

[20] Mark Kirkpatrick, David Lofsvold, and Michael Bulmer. Analysis of the inheritance, selection and evolution of growth trajectories. *Genetics*, 124(4):979–993, 1990.

[21] Mark Kirkpatrick, William G Hill, and Robin Thompson. Estimating the covariance structure of traits during growth and ageing, illustrated with lactation in dairy cattle. *Genetics Research*, 64(1):57–69, 1994.

[22] Hannes Matuschek, Reinhold Kliegl, Shravan Vasishth, Harald Baayen, and Douglas Bates. Balancing type i error and power in linear mixed models. *Journal of memory and language*, 94:305–315, 2017.

[23] Karin Meyer. Estimating covariance functions for longitudinal data using a random regression model. *Genetics Selection Evolution*, 30(3):221–240, 1998.

[24] Karin Meyer and William G Hill. Estimation of genetic and phenotypic covariance functions for longitudinal or 'repeated'records by restricted maximum likelihood. *Livestock Production Science*, 47(3):185–200, 1997.

[25] Karin Meyer and Mark Kirkpatrick. Up hill, down dale: quantitative genetics of curvaceous traits. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1459):1443–1455, 2005.

[26] Timothy A Mousseau and Derek A Roff. Natural selection and the heritability of fitness components. *Heredity*, 59(2):181–197, 1987.

[27] Ikuo Okada and RT Hardin. An experimental examination of restricted selection index, using tribolium castaneum. i. the results of two-way selection. *Genetics*, 57 (2):227, 1967.

[28] GJ Ragland and PA Carter. Genetic covariance structure of growth in the salamander ambystoma macrodactylum. *Heredity*, 92(6):569–578, 2004.

[29] JO Ramsay and BW Silverman. Principal components analysis for functional data. *Functional data analysis*, pages 147–172, 2005.

[30] Mauro Santos, Alfredo Ruiz, Jorge E Quezada-Díaz, Antonio Barbadilla, and Antonio Fontdevila. The evolutionary history of drosophila buzzatii. xx. positive phenotypic covariance between field adult fitness components and body size. *Journal of Evolutionary Biology*, 5(3):403–422, 1992.

[31] Natalia Sokolovska, Locke Rowe, and Frank Johansson. Fitness and body size in mature odonates. *Ecological entomology*, 25(2):239–248, 2000.

[32] AI Vazquez, DM Bates, GJM Rosa, D Gianola, and KA Weigel. an r package for fitting generalized linear mixed models in animal breeding. *Journal of animal science*, 88(2):497–504, 2010.