May 2016

Longitudinal Data Models with Nonparametric Random Effect Distributions

Hartmut Jakob Stenz
University of Wisconsin-Milwaukee

Longitudinal data models with nonparametric random effect distributions

by

Hartmut Jakob Stenz

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Mathematics

at

The University of Wisconsin-Milwaukee

May 2016

Abstract

Longitudinal data models with nonparametric random effect distributions

by

Hartmut Jakob Stenz

The University of Wisconsin-Milwaukee, 2016
Under the Supervision of Professor Daniel Gervini

There is the saying which says you cannot see the woods for the trees. This thesis aims to circumvent this unfortunate situation: Longitudinal data on tree growth, as an example of multiple observations of similar individuals pooled together in one data set, are modeled simultaneously rather than each individual separately. This is done under the assumption that one model is suitable for all individuals but its parameters vary following unknown nonparametric random effect distributions. The goal is a maximum likelihood estimation of these distributions considering all provided data and using basis-spline-approximations for the densities of each distribution function over the same spline-base. The implementation of all procedures is carried out in `R` and attached to this thesis.

Table of contents

List of figures

List of tables

Acknowledgments

First, I would like to express my sincere gratitude to my advisor Professor Daniel Gervini for his guidance and assistance throughout this thesis project.

I also thank my friend and fellow student Martin G. Vieten, not only for his advice and academic inspiration, but especially for the time we could spend together.

Finally, I am very grateful for the support of my friends and roommates here in the United States.

Für meinen Bruder,

Johannes Mariano,

der mich nie im Stich lässt.

In dubio pro deo.

1 Introduction

1.1 Data

Our data, provided in the `R`-package "`spuRs`" belonging to [1], contains $N = 106$ trees, coded by its location, site and number. For each tree T_i, m_i observations of the bole volumes v_{ij} in cubic-decimeters at ages z_{ij}, when the tree was measured, are reported. So, for each tree T_i we have a time-series $(v_{ij})_{1 \leq j \leq m_i}$ over the set $(z_{ij})_{1 \leq j \leq m_i}$ of points in time. Neither are for a given tree $(z_{ij})_{1 \leq j \leq m_i}$ equidistant, nor are the m_i's equal, so we have a different number of observations for each tree.

1.2 General model

To describe the relationship between the age of a tree and its bole volume we will use a model containing the *Richards' curve*, which computes v_{ij} for a given z_{ij} by using parameters $\theta_i = (a_i, b_i, c_i)^T$:

$$v_{ij} = g(\theta_i, z_{ij}) + \varepsilon = a_i \left(1 - e^{-b_i z_{ij}}\right)^{c_i} + \varepsilon_{ij}. \tag{1}$$

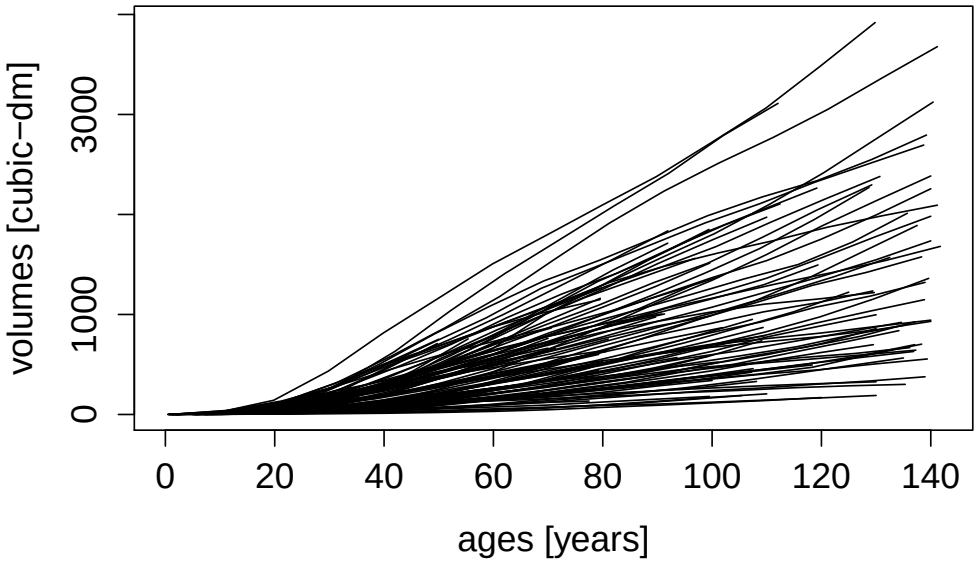

Figure 1: Plot of all tree data. Each line is an interpolations of the data of one tree.

a_i stands for the maximum size of the tree T_i, b_i describes the speed of its growing and c_i as a compensatory parameter should be near to 3 for each tree. For more information see page 238 in [1]. Our goal is to estimate the distribution functions of these parameters by using basis splines, not like in [2], where function g describing a shape invariant model is approximated with b-splines. The quantities $\varepsilon_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$ are the noises of the giving time series with a constant variance σ^2 for all trees. We must also estimate σ^2.

2 Maximum likelihood estimation

2.1 Conditional independence

Before an estimation is done questions of independence in a given model have to be clarified: The noises ε_{ij} are assumed independent and identically normally distributed. Therefore, if we know θ_i, a good estimator for the variance σ^2 would be given by the *maximum likelihood estimator*:

$$\sigma^2 \approx \frac{1}{m_i} \sum_{j=1}^{m_i} \left(v_{ij} - g\left(\theta_i, z_{ij}\right)\right)^2 .$$

Assuming now we have a θ for all trees, a reasonable estimator of σ^2 can be the mean of all MLE's with $\theta_i = \theta$ for all i:

$$\sigma^2 \approx s^2 := \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{m_i} \sum_{j=1}^{m_i} \left(v_{ij} - g\left(\theta, z_{ij}\right)\right)^2 \right). \tag{2}$$

However, s^2 in (2) is not longer a MLE for σ^2. Although observations from different individuals are independent, the data from a single individual are not independent, but rather *conditionally independent*: Given a parameter

vector $\theta_i = (a_i, b_i, c_i)^T$ of tree T_i,

$$v_{ij}|\theta_i \sim \mathcal{N}\left(g\left(\theta_i, z_{ij}\right), \sigma^2\right) \tag{3}$$

for all $j \in \{1, \ldots, m_i\}$ and $i \in \{1, \ldots, N\}$. Equation (3) follows from (1) by solving for ε_{ij} and using the fact that ε_{ij} are normally distributed for every observation.

2.2 Likelihood function and gradient

Based on the idea of conditional independence we can now compute a likelihood function $L(\eta)$ to estimate the parameter vector η of a density function $f_\eta(\theta)$. Notice that η is not directly a parameter vector for the model itself but it describes the density function of the model random effect θ. Given

such a $f_\eta(\theta)$, the joint pdf of the observations for each tree is:

$$\begin{aligned} f_i\left(v_{i,j=1,\ldots,m_i}\right) &= \int f_\eta(\theta)\,\Pi_{j=1}^{m_i} f\left(v_{ij}|\theta\right)d\theta \\ &\underset{(3)}{=} \int f_\eta(\theta)\,\Pi_{j=1}^{m_i}\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{v_{ij}-g\left(z_{ij},\theta\right)}{2\sigma^2}\right)\right)d\theta \\ &\underset{(*)}{=} \int f_\eta(\theta)\left(\frac{1}{\sqrt{2\pi s^2}}\right)^{m_i}\exp\left(-\frac{\sum_{j=1}^{m_i}\left(v_{ij}-g\left(z_{ij},\theta\right)\right)^2}{2s^2}\right)d\theta \\ &\underset{(**)}{=} \int f_\eta(\theta)\left(\frac{1}{\sqrt{2\pi s^2}}\right)^{m_i}\exp\left(-\frac{h_i}{2s^2}\right)d\theta \end{aligned} \tag{4}$$

with

$$(**) \quad h_i := \sum_{j=1}^{m_i}\left(v_{ij}-g\left(\theta,z_{ij}\right)\right)^2 \tag{5}$$

$$(*) \quad s^2 \underset{(2)}{:=} \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\left(v_{ij}-g\left(\theta,z_{ij}\right)\right)^2\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{h_i}{m_i}. \tag{6}$$

In (6) we plug in the estimator for σ^2 from (2). Notice that the value of h_i and s^2 just depend on the data and a given θ, but not on η. To go a step

further, we take a look at the joint density f of all observations:

$$\begin{aligned} f\left(v_{\substack{i=1,\ldots,N \\ j=1,\ldots,m_i}}\right) &\underset{T_i \text{ independent}}{=} \Pi_{i=1}^{N} f_i\left(v_{i,j=1,\ldots,m_i}\right) \\ &\underset{(4)}{=} \Pi_{i=1}^{N} \int f_\eta(\theta) \left(\frac{1}{\sqrt{2\pi s^2}}\right)^{m_i} \exp\left(-\frac{h_i}{2s^2}\right) d\theta \\ &= \Pi_{i=1}^{N} \int f_\eta(\theta)\, I_i d\theta \end{aligned} \tag{7}$$

with

$$I_i := \left(\frac{1}{\sqrt{2\pi s^2}}\right)^{m_i} \exp\left(-\frac{h_i}{2s^2}\right). \tag{8}$$

The maximum likelihood estimator $\hat{\eta}$ maximizes $L(\eta)$. Taking the logarithm in (7), we get the log-likelihood function $l(\eta)$:

$$l(\eta) = \log\left(\Pi_{i=1}^{N} \int f_\eta(\theta)\, I_i d\theta\right) = \sum_{i=1}^{N} \log\left(\int f_\eta(\theta)\, I_i d\theta\right). \tag{9}$$

Note that $\hat{\eta}$ also maximizes $l(\eta)$, which is numerically easier to work with. Now we compute the entries of the gradient,

$$\frac{d}{d\eta_x} l(\eta) = \frac{d}{d\eta_x}\left(\sum_{i=1}^{N} \log\left(\int f_\eta(\theta)\, I_i d\theta\right)\right) = \sum_{i=1}^{N} \frac{\int \frac{d}{d\eta_x} f_\eta(\theta)\, I_i d\theta}{\int f_\eta(\theta)\, I_i d\theta}, \tag{10}$$

which are needed to numerically compute $\hat{\eta}$. These functions can be written more explicitly, but to do so we have to specify the density $f_\eta(\theta)$ of the nonparametric random effect distribution.

2.3 Nonparametric random effect distribution

Now we assume that $a \in [\alpha, \alpha + d_\alpha]$, $b \in [\beta, \beta + d_\beta]$ and $c \in [\gamma, \gamma + d_\gamma]$. Furthermore, a, b and c are independent, f_α, f_β and f_γ are their density functions on their original ranges and f_{η^α}, f_{η^β} and f_{η^γ} density functions of $\frac{a-\alpha}{d_\alpha}$, $\frac{b-\beta}{d_\beta}$ and $\frac{c-\gamma}{d_\gamma}$ approximated by b-splines given by the basis functions $\{\delta_1, \ldots, \delta_p\}$ on $[0,1]$. Then with $\eta = \left\{\eta_1^\alpha, \ldots, \eta_p^\alpha, \eta_1^\beta, \ldots, \eta_p^\beta, \eta_1^\gamma, \ldots, \eta_p^\gamma\right\} \in \mathbb{R}_+^{3p}$, $a = \alpha + d_\alpha u$, $b = \beta + d_\beta v$, $c = \gamma + d_\gamma w$ and $u, v, w \in [0,1]$ we get

$$\begin{aligned} f_\eta(\theta) &= f_\alpha(a) f_\beta(b) f_\gamma(c) = \frac{1}{d_\alpha} f_{\eta^\alpha}\left(\frac{a-\alpha}{d_\alpha}\right) \frac{1}{d_\beta} f_{\eta^\beta}\left(\frac{b-\beta}{d_\beta}\right) \frac{1}{d_\gamma} f_{\eta^\gamma}\left(\frac{c-\gamma}{d_\gamma}\right) \\ &= \frac{1}{d_\alpha d_\beta d_\gamma} \left(\sum_{r=1}^{p} \eta_r^\alpha \delta_r(u)\right) \left(\sum_{s=1}^{p} \eta_s^\beta \delta_s(v)\right) \left(\sum_{t=1}^{p} \eta_t^\gamma \delta_t(w)\right). \end{aligned} \tag{11}$$

Because (11) has to be a density function, $\eta \in \mathbb{R}_+^{3p}$ and it should hold

$$1 = \int_0^1 f_{\eta^\alpha}(u)du = \sum_{r=1}^{p} \eta_r^\alpha \int_0^1 \delta_r(u)du = \sum_{r=1}^{p} \eta_r^\alpha e_r$$

with $e_r = \int_0^1 \delta_r(u)du$. We can reach this by dividing $f_{\eta^\alpha}(a)$ by $\sum_{r=1}^p \eta_r^\alpha e_r$. So let us define

$$f_a(u) := \frac{f_{\eta^\alpha}(u)}{\sum_{r=1}^p \eta_r^\alpha e_r} = \frac{\sum_{r=1}^p \eta_r^\alpha \delta_r(u)}{\sum_{r=1}^p \eta_r^\alpha e_r}. \tag{12}$$

Doing the same for $f_{\eta^\beta}(v)$ and $f_{\eta^\gamma}(w)$ we finally get:

$$\begin{aligned} f_\eta(\theta) &= \frac{\left(\sum_{r=1}^p \eta_r^\alpha \delta_r(u)\right)\left(\sum_{s=1}^p \eta_s^\beta \delta_s(v)\right)\left(\sum_{t=1}^p \eta_t^\gamma \delta_t(w)\right)}{d_\alpha d_\beta d_\gamma \left(\sum_{r=1}^p \eta_r^\alpha e_r\right)\left(\sum_{s=1}^p \eta_s^\beta e_s\right)\left(\sum_{t=1}^p \eta_t^\gamma e_t\right)} \\ &= \frac{f_a(u) f_b(v) f_c(w)}{d_\alpha d_\beta d_\gamma}. \end{aligned} \tag{13}$$

Equation (13) leads to an interesting fact: Let $\eta' = k\eta$ for a $k > 0$. Then $f_{\eta'} = f_\eta$, because for each parameter a, b and c, from (12) we have

$$\frac{\sum_{r=1}^p \eta_r^{\alpha'} \delta_r(u)}{\sum_{r=1}^p \eta_r^{\alpha'} e_r} = \frac{\sum_{r=1}^p k\eta_r^\alpha \delta_r(u)}{\sum_{r=1}^p k\eta_r^\alpha e_r} = \frac{\sum_{r=1}^p \eta_r^\alpha \delta_r(u)}{\sum_{r=1}^p \eta_r^\alpha e_r} = f_a(u).$$

So, if there exists an optimal solution for η, there would actually be infinite optimal solutions, or, more precisely, for an optimal solution η^* the ray $\{\eta | \eta = k\eta^*, k > 0\} \subset \mathbb{R}_+^{3p}$ contains only optimal solutions too.

Now we calculate $\frac{d}{d\eta_x^\alpha} f_\eta(\theta)$. Therefore we see that:

$$\frac{d}{d\eta_x^\alpha} f_\eta(\theta) = \frac{f_b(v) f_c(w)}{d_\alpha d_\beta d_\gamma} \frac{d}{d\eta_x^\alpha} f_a(u). \tag{14}$$

So it is enough to compute $\frac{d}{d\eta_x^\alpha} f_a(u)$. This can be done by using the *quotient rule* in (12):

$$\begin{aligned} \frac{d}{d\eta_x^\alpha} f_a(u) &= \frac{\delta_x(u)\left(\sum_{r=1}^p \eta_r^\alpha e_r\right) - e_x \sum_{r=1}^p \eta_r^\alpha \delta_r(u)}{\left(\sum_{r=1}^p \eta_r^\alpha e_r\right)^2} \\ &= \frac{\delta_x(u) - e_x f_a(u)}{\sum_{r=1}^p \eta_r^\alpha e_r}. \end{aligned} \tag{15}$$

Plugging (15) into (14) we get:

$$\begin{aligned} \frac{d}{d\eta_x^\alpha} f_\eta(\theta) &= \frac{f_b(v) f_c(w)\left(\delta_x(u) - e_x f_a(u)\right)}{d_\alpha d_\beta d_\gamma \sum_{r=1}^p \eta_r^\alpha e_r} \\ &= \frac{1}{\sum_{r=1}^p \eta_r^\alpha e_r}\left(\frac{\delta_x(u) f_b(v) f_c(w)}{d_\alpha d_\beta d_\gamma} - e_x f_\eta(\theta)\right). \end{aligned} \tag{16}$$

Under our assumptions about the ranges of a,b and c, we can transform g, a function of vector $\theta = (a, b, c)^T$, to $\tilde{g}$, a function of $\theta' = (u, v, w)^T \in [0, 1]^3$:

$$\tilde{g}(\theta', z_{ij}) = (\alpha + d_\alpha u)\left(1 - e^{-\left(\beta + d_\beta v\right) z_{ij}}\right)^{\gamma + d_\gamma w}. \tag{17}$$

Same for h_i, s and I by plugging in (17) into (5), (6) and (8), so they become $\tilde{h}_i$, $\tilde{s}$ and $\tilde{I}_i$. Also by plugging in (13) into (9) and doing *integration by substitution* we finally get:

$$\begin{aligned} l(\eta) &\underset{(9)}{=} \sum_{i=1}^{N} \log\left(\int f_\eta(\theta) I_i d\theta\right) \underset{\theta\to\theta'}{=} \sum_{i=1}^{N} \log\left(\int_{[0,1]^3} f_\eta(\theta') \tilde{I}_i d_\alpha d_\beta d_\gamma d\theta'\right) \\ &\underset{(13)}{=} \sum_{i=1}^{N} \log\left(\int_{[0,1]^3} f_a(u) f_b(v) f_c(w) \tilde{I}_i d\theta'\right). \end{aligned} \tag{18}$$

In a similar way we can plug in (16) into (10) for each parameter and use the same substitutions to get the final gradient:

$$\frac{d}{d\eta} l(\eta) = \left(\frac{d}{d\eta_1^\alpha} l(\eta), \ldots, \frac{d}{d\eta_p^\alpha} l(\eta), \frac{d}{d\eta_1^\beta} l(\eta), \ldots, \frac{d}{d\eta_p^\alpha} l(\eta), \frac{d}{d\eta_1^\gamma} l(\eta), \ldots, \frac{d}{d\eta_p^\gamma} l(\eta)\right)$$

with

$$\frac{d}{d\eta_x^\alpha} l(\eta) = \frac{\left(\sum_{i=1}^{N} \frac{\int_{[0,1]^3} \delta_x(u) f_b(v) f_c(w) \tilde{I}_i d\theta'}{\int_{[0,1]^3} f_\eta(\theta') \tilde{I}_i d\theta'}\right) - N e_x}{\sum_{r=1}^{p} \eta_r^\alpha e_r} \tag{19}$$

$$\frac{d}{d\eta_x^\beta} l(\eta) = \frac{\left(\sum_{i=1}^{N} \frac{\int_{[0,1]^3} f_a(u) \delta_x(v) f_c(w) \tilde{I}_i d\theta'}{\int_{[0,1]^3} f_\eta(\theta') \tilde{I}_i d\theta'}\right) - N e_x}{\sum_{s=1}^{p} \eta_s^\beta e_s} \tag{20}$$

$$\frac{d}{d\eta_x^\gamma} l(\eta) = \frac{\left(\sum_{i=1}^{N} \frac{\int_{[0,1]^3} f_a(u) f_b(v) \delta_x(w) \tilde{I}_i d\theta'}{\int_{[0,1]^3} f_\eta(\theta') \tilde{I}_i d\theta'}\right) - N e_x}{\sum_{t=1}^{p} \eta_t^\gamma e_t}. \tag{21}$$

2.4 Spline basis

Now we want to specify the basis $\{\delta_1, \ldots, \delta_p\}$ on $[0,1]$ under the assumption that the splines have degree $q < p$ and we divide $[0,1]$ in $p-q$ equidistant pieces. We define our knots with $\omega_k = \frac{k}{p-q}$ for all k as follows:

$$\omega_{-q} < \cdots < \omega_{-1} < 0 = \omega_0 < \omega_1 < \cdots < \omega_{p-q-1} < \omega_{p-q} = 1 < \omega_{p-q+1} < \cdots < \omega_p.$$

Let us denote with $\delta_t^q(w)$ the t^{th} b-spline at w of degree q. In this case we have the following recursive definition of a b-spline:

$$\delta_t^0(w) = \begin{cases} 1 & \omega_{t-1} \leq w < \omega_t \\ 0 & \text{otherwise} \end{cases} = \mathcal{I}_{[\omega_{t-1}, \omega_t)}(w) \tag{22}$$

$$\begin{aligned} \delta_t^q(w) &= \frac{w - \omega_{t-q-1}}{\omega_{t-1} - \omega_{t-q-1}} \delta_{t-1}^{q-1}(w) + \frac{\omega_t - w}{\omega_t - \omega_{t-q}} \delta_t^{q-1}(w) \\ &= \frac{p-q}{q} \left((w - \omega_{t-q-1}) \delta_{t-1}^{q-1}(w) + (\omega_t - w) \delta_t^{q-1}(w) \right). \end{aligned} \tag{23}$$

For $\delta_t^q(w)$ we can also use the following explicit formula:

$$\delta_t^q(w) = \frac{(p-q)^q}{q!} \sum_{k=0}^{q+1} \binom{q+1}{k} (-1)^k (w - \omega_{k+t-q-1})_+^q. \tag{24}$$

We will prove (24) by induction, with $(a-b)_+^c = (a-b)_+^{c-1}(a-b)$ and the convention $0^0 = 1$:

$\boxed{q=0}$

$$\frac{(p-0)^0}{0!}\sum_{k=0}^{1}\binom{1}{k}(-1)^k(w-\omega_{k+t-0-1})_+^0$$

$$=\binom{1}{0}(w-\omega_{t-1})_+^0-\binom{1}{1}(w-\omega_t)_+^0=\mathcal{I}_{[\omega_{t-1},\omega_t)}(w)=\delta_t^0(w).$$

$\boxed{q-1\to q}$

$$\frac{(p-q)^q}{q!}\sum_{k=0}^{q+1}\binom{q+1}{k}(-1)^k(w-\omega_{k+t-q-1})_+^q$$

$$=\frac{(p-q)^q}{q!}\left((w-\omega_{t-q-1})_+^q+\sum_{k=1}^{q}\binom{q+1}{k}(-1)^k(w-\omega_{k+t-q-1})_+^q\right)$$

$$+\frac{(p-q)^q}{q!}(-1)^{q+1}(w-\omega_t)_+^q$$

$$=\frac{(p-q)^q}{q!}\left((w-\omega_{t-q-1})_+^q+\sum_{k=1}^{q}\left(\binom{q}{k}+\binom{q}{k-1}\right)(-1)^k(w-\omega_{k+t-q-1})_+^q\right)$$

$$+\frac{(p-q)^q}{q!}(-1)^{q+1}(w-\omega_t)_+^q$$

$$=\frac{(p-q)^q}{q!}\left(\sum_{k=0}^{q}\binom{q}{k}(-1)^k(w-\omega_{k+t-q-1})_+^q+\sum_{k=1}^{q}\binom{q}{k-1}(-1)^k(w-\omega_{k+t-q-1})_+^q\right)$$

$$+\frac{(p-q)^q}{q!}(-1)^{q+1}(w-\omega_t)_+^q$$

$$
\begin{aligned}
&= \frac{(p-q)^q}{q!}\left(\sum_{k=0}^{q}\binom{q}{k}(-1)^k\left(w-\omega_{k+t-q-1}\right)_+^q + \sum_{k=0}^{q}\binom{q}{k}(-1)^{k+1}\left(w-\omega_{k+t-q}\right)_+^q\right)\\
&= \frac{(p-q)^q}{q!}\sum_{k=0}^{q}\binom{q}{k}(-1)^k\left(w-\omega_{k+t-q-1}\right)_+^{q-1}\left(w-\omega_{t-q-1}-\frac{k}{p-q}\right)\\
&\quad + \frac{(p-q)^q}{q!}\sum_{k=0}^{q}\binom{q}{k}(-1)^k\left(w-\omega_{k+t-q}\right)_+^{q-1}\left(\omega_t-w-\frac{q-k}{p-q}\right)\\
&= \frac{(p-q)}{q}\left(w-\omega_{t-q-1}\right)\frac{(p-q)^{q-1}}{(q-1)!}\sum_{k=0}^{q}\binom{q}{k}(-1)^k\left(w-\omega_{k+(t-1)-(q-1)-1}\right)_+^{q-1}\\
&\quad + \frac{(p-q)}{q}\left(\omega_t-w\right)\frac{(p-q)^{q-1}}{(q-1)!}\sum_{k=0}^{q}\binom{q}{k}(-1)^k\left(w-\omega_{k+t-(q-1)-1}\right)_+^{q-1}\\
&\underbrace{-\sum_{k=1}^{q}\frac{(p-q)^{q-1}}{(q-k)!\,(k-1)!}(-1)^k\left(w-\omega_{k+t-q-1}\right)_+^{q-1} + \sum_{k=0}^{q-1}\frac{(p-q)^{q-1}}{(q-k-1)!k!}(-1)^{k+1}\left(w-\omega_{k+t-q}\right)_+^{q-1}}_{=0}\\
&= \frac{p-q}{q}\left(\left(w-\omega_{t-q-1}\right)\delta_{t-1}^{q-1}(w)+\left(\omega_t-w\right)\delta_t^{q-1}(w)\right).
\end{aligned}
$$

q.e.d.

Equation (23) implies that $\delta_t^q(w) > 0$ only for $\omega_{t-q-1} \leq w \leq \omega_t$, which can be simply proved by induction over q: $\delta_t^0(w) = \mathcal{I}_{[\omega_{t-1},\omega_t)}(w)$ and therefore holds for $q > 1$ that $\delta_t^q(w)$ as linear-combination of $\delta_{t-1}^{q-1}(w)$, by induction not 0 between $\omega_{(t-1)-(q-1)-1} = \omega_{t-q-1}$ and ω_{t-1}, and $\delta_t^{q-1}(w)$, also by induction not 0 between $\omega_{t-(q-1)-1}$ and ω_t, is in fact not 0 between ω_{t-q-1} and ω_t. This fact will become important in Section 3.3 when we want to find good initial values for the optimization.

3 Real-data examples

3.1 Richards' curve implementation and curve fitting

In `R`, all trees are written in a list `B0`, where `B0`$[[i]][1,]$ is the time series of volumes v_{ij} for tree T_i to the corresponding points in time z_{ij} saved in `B0`$[[i]][2,]$. The function `L(B0)` in the appendix returns a vector containing the number of observations m_i for each tree. Model (1) is implemented as function $\mathtt{g}(\theta_i, z_{ij})$ in the appendix in a way that it takes not only a single parameter vector θ_i, but also a whole parameter matrix $(\theta_{i_1} \ldots, \theta_{i_n})^T$ and a given array $(z_{i1}, \ldots, z_{im_i})^T$ of points in time and calculates the matrix $(v_{i_k j})_{\substack{k=1,\ldots,n \\ j=1,\ldots,m_i}}$ of estimated bole volumes. The reason for this is (18); for the likelihood function we have to compute integrals over $[0, 1]$. We will do that by generating different combinations of θ_{i_k} and evaluating $\mathtt{g}(\theta_{i_k}, z_{i_k j})$ at these combinations. The sum of these evaluations will then be divided by the total number of different combinations to get an estimation of the integral. To generate these different combinations we use function `Int` in the appendix, which returns a $(n^3) \times 3$-matrix, where each line is a different combination of parameters chosen to be equal to $\frac{k+0.5}{n}$ for $k = 0, \ldots n-1$. An example of an output of this function is given in the appendix. Before implementation of h_i

and s, the ranges $[\alpha, \alpha + d_\alpha]$, $[\beta, \beta + d_\beta]$ and $[\gamma, \gamma + d_\gamma]$ have to be estimated comparing Section 2.3. This can be done by using the method of *curve fitting* and the `R`-function `Optim`. For more information see Section 12.7 in [1]. The sum of squares will be set as loss-function `fkt` in the appendix. We also have to choose an initial value θ_0: This choice leads to different results. This will be discussed in chapter 4. We will set $\theta_0 = (1000, 0.1, 3)^T$ as it is done in [1]. The results of the curve fitting for the whole data-set is written in a list `P0` by the function `P`, function 5 in the appendix: The first three list members are arrays consisting of all curve fitting results for parameters a, b and c each; the last member is a 3×2-matrix reporting in its first column the lowest curve fitting results α, β and γ of all parameters and in the second column the difference between the lowest and highest results and therefore the lengths d_α, d_β and d_γ of the ranges for each parameter. The values in `P0`[[1]], `P0`[[2]] and `P0`[[3]] are represented in histograms with equal sized categories over the normalized ranges of each parameter, so that each category represents a percentage of the whole range. This is done by function `Hst` in the appendix, which gives back the number of results in each category. This procedure will be important for finding initial values in the section 3.3. Of course, different numbers of categories lead to different results, not only for the histograms,

but also for the initial values; concerning this last question, the number of categories is related to the number of b-splines. For more information see Section 3.3.

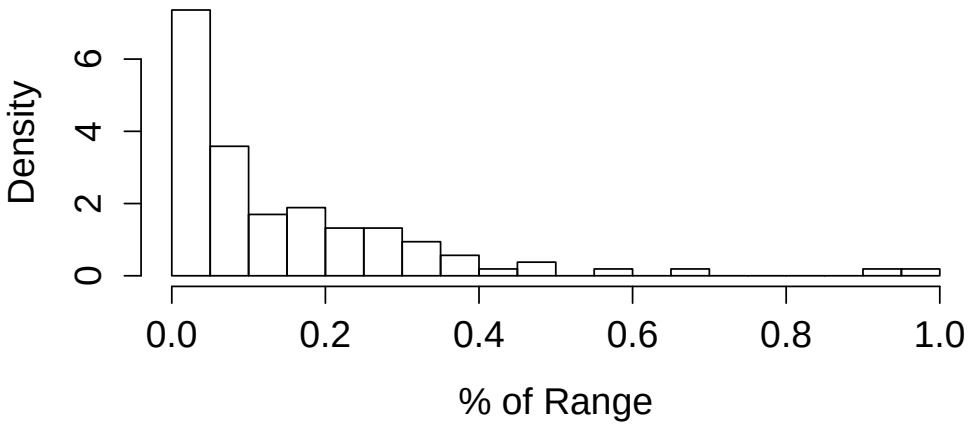

Figure 2: Histogram of a with $n = 20$ categories.

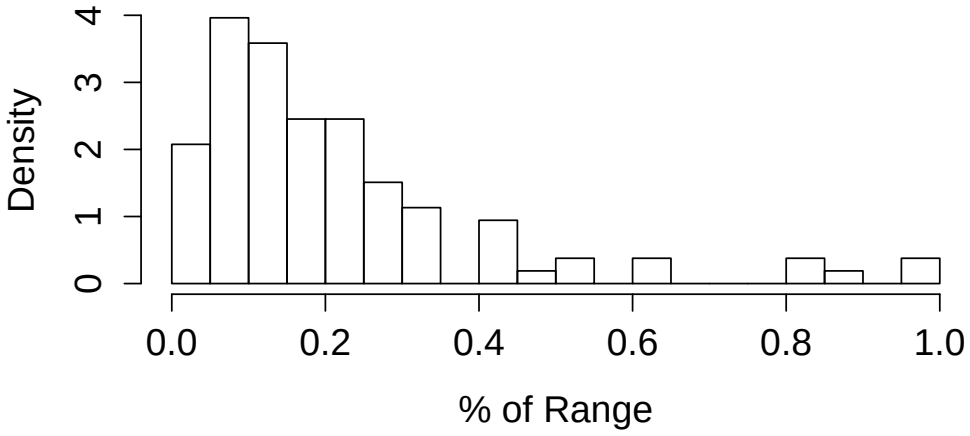

Figure 3: Histogram of b with $n = 20$ categories.

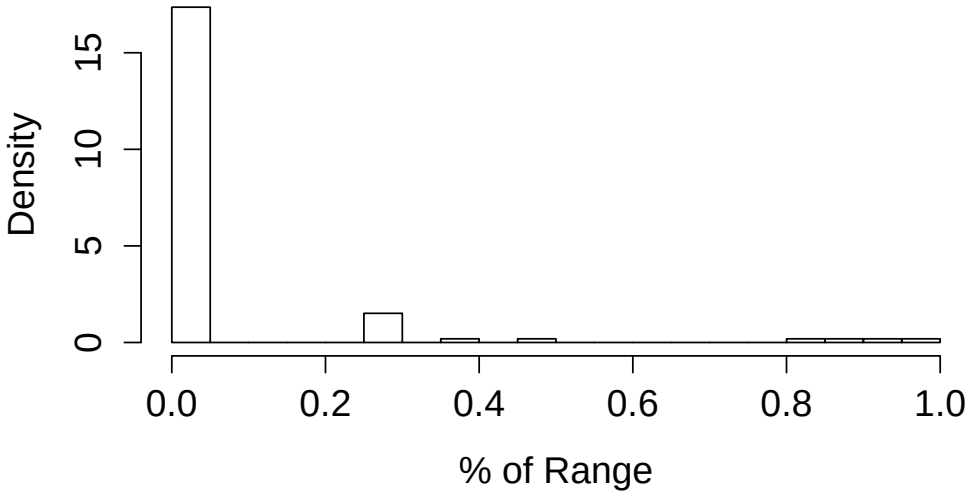

Figure 4: Histogram of c with $n = 20$ categories.

P00[[4]]	min	max-min
a	557.8971	1097.766
b	0.003185994	0.09697149
c	1.297667	359.4736

Table 1: Parameter matrix concerning all curve-fitting results

3.2 Parametrization and rest of implementation

The matrix P0[[4]] is reported in Table 1 and will be used in upcoming calculations frequently because of the following relationship; let θ be a vector in $[0,1]^3$, then θ' with

$$\theta_i' := \texttt{P0}[[4]][i,1] + \theta_i \cdot \texttt{P0}[[4]][i,2] \tag{25}$$

for $i = 1,2,3$ is a vector in $[\alpha, \alpha + d_\alpha] \times [\beta, \beta + d_\beta] \times [\gamma, \gamma + d_\gamma]$. With (25), h_i, s and I_i can be transformed to $\tilde{h}_i$, $\tilde{s}$ and $\tilde{I}_i$, which is done in the beginning of function `h` in the appendix: The input, a $n^3 \times 3$-matrix `M=Int(n)` with entries in $[0,1]$, is changed to matrix `A` with entries in the ranges of the corresponding parameters before it is used to calculate h_i following equation (5). The output is then a vector of length n^3 containing $\tilde{h}_i$ of tree T_i for n^3 different parameter

combinations. Function `H` in the appendix creates matrix $H = \left(\tilde{h}_1 \cdots \tilde{h}_N\right)$, so that the i^{th} column is $\tilde{h}_i$ of tree T_i, and function `s` in the appendix uses `H` and `L` to calculate $\tilde{s}$ following equation (6). Finally, function `I` in the appendix computes matrix $\left(\tilde{I}_{1_k}, \cdots, \tilde{I}_{N_k}\right)_{1 \leq k \leq n^3}$ following equation (8) and taking `P0`, `B0` and `M=Int(n)` as inputs: For each tree T_i and each parameter combination θ'_k, $\tilde{I}_{i_k}$ is independent from any η. Also independent from η are the spline bases: For a given p, q and a vector $x \in [0,1]^{n^3}$, function `bsplines` in the appendix calculates matrix $\left(\delta_1^q(x_k), \ldots, \delta_p^q(x_k)\right)_{1 \leq k \leq n^3}$ following (24). To apply definition (12) we also use function `bint` in the appendix, which computes for a given p, q and m vector $(e_1, \ldots, e_p)$ with $e_k = \int_0^1 \delta_r^q(u)du$ by taking for each δ_r^q the mean over $\delta_r^q(x)$ with $x = \left(\frac{1}{m}, \frac{2}{m}, \ldots, \frac{m}{m} = 1\right)$. Taking a parameter combination, given as matrix `M`$= \left(\theta_{i_1}, \ldots, \theta_{i_{n^3}}\right)$, and m as input, function `b` in the appendix calculates a list; the first three members of the list contain the b-splines for vectors $\left(a_{i_1}, \ldots, a_{i_{n^3}}\right)$, $\left(b_{i_1}, \ldots, b_{i_{n^3}}\right)$ and $\left(c_{i_1}, \ldots, c_{i_{n^3}}\right)$, and the last member is `bint`(p, q, m). So far, every calculation can be done without a specific η. Now function `dstb` in appendix with inputs k, a vector $v \in \mathbb{R}_+^p$ and a list `b` of b-splines and their integrals, applies definition (12) by using matrix multiplication and returns the density f_a of a for $k = 1$,f_b of b for $k = 2$ and f_c of c for $k = 3$. Finally, the joint

density $f_a(u) f_b(v) f_c(w)$ over a given parameter combination $\left(\theta_{i_1}, \ldots, \theta_{i_{n^3}}\right)$ for vector $\eta \in \mathbb{R}_+^{3p}$ is computed by function `f` in the appendix, which returns vector $\left(f_\eta\left(\theta'_{i_1}\right), \ldots, f_\eta\left(\theta'_{i_{n^3}}\right)\right)$. As a consequence, this output corresponds to the output of `I` on a given parameter combination `M`: Vector `f`$\in [0,1]^{n^3}$ multiplied with matrix `I`$\in \mathbb{R}_+^{n^3 \times N}$ becomes a vector. The N entries of this vector, divided by n^3, are an estimation of the integrals in our likelihood function from (18). So we get $l(\eta)$ by taking the logarithm of each entry and sum them up. This does function `l` in the appendix and in a similar way, applying (19) – (21), function `dl` in the appendix computes $\frac{d}{d\eta} l(\eta)$. Both functions do not have any `for`-loops and for each input of η they have the same input `I0` and `b0`. Therefore, an evaluation of `l` and also `dl` is done very quickly once `I0` and `b0` are computed.

3.3 Initial values

Our optimization problem is constrained because $\eta_i \geq 0$ for all $1 \leq i \leq 3p$. If $\mathcal{I} \in \mathbb{R}^{3p \times 3p}$ is the identity matrix, we can formalize this condition:

$$\mathcal{I} \cdot \eta \geq c := \mathbf{0} \in \mathbb{R}^{3p}. \tag{26}$$

`R` provides function `constrOptim` taking as input our functions `l` and `dl`, values $\mathcal{I}$ and c from (26), our calculated values for `I0` and `b0` and an initial value for η. Therefore, we will first try to find such an initial value. We have already discussed that, if there is an optimal solution η^*, the ray $\{\eta|\eta = k\eta^*, k > 0\} \subset \mathbb{R}_+^{3p}$ actually contains infinite optimal solutions, more precisely, $l(\eta')$ is the same for all $\eta' \in \{\eta|\eta = k\eta^*, k > 0\}$. Our initial value should be near to that ray in order to increase the probability of reaching it faster. One idea for finding such a value would be to use a histogram with n categories: As a discrete estimation of the density, a histogram gives an idea what this density should look like. Therefore, if we take as initial value a η which computes a density near to the histogram, we would get a continuous estimation of the real density. To do so, we use the fact, that the b-spline δ_t^q is not 0 on the interval $(\omega_{t-q-1}, \omega_t)$. Let $(c_i)_{i=1,\dots,n}$ be the categories of the given histogram and $(h_i)_{i=1,\dots,n}$ the corresponding numbers of observations in these categories. We want to identify c_i with the interval (ω_{i-1}, ω_i) for $i = 1, \dots, n$ and add $c_0, \dots, c_{-q+1}$ with $h_0 = \dots = h_{-q+1} = h_1$ and also $c_{n+1}, \dots, c_{n+q}$ with $h_{n+1} = \dots = h_{n+q} = h_n$. Setting $\eta_t^k = \sum_{i=t-q}^{t} h_i$ for all $1 \leq t \leq p$ and $k = \alpha, \beta, \gamma$. Length p has to be equal to $n - q$, so the choice for the histogram is also a choice for the dimension of η which we

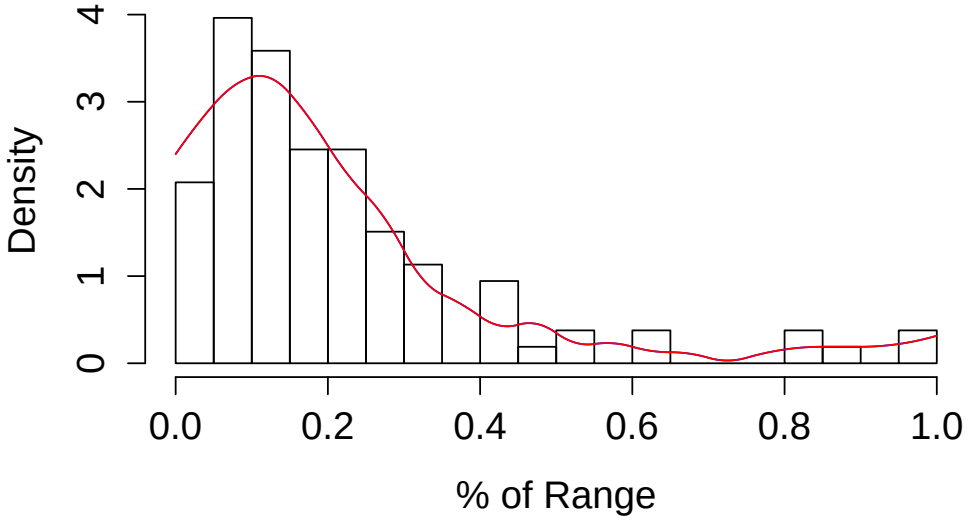

Figure 5: Density (red) of the initial value for parameter b with $n = 20$ and $q = 2$.

want to estimate. Function `initial` in the appendix computes, for a given p and q, the initial value we need by using `P0` and function `Hst`. In Table 2, $l\left(\eta = \texttt{initial}\left(\texttt{P0}, p, q\right), \texttt{I0}, \texttt{b0}\right)$ for different p and q are listed, there `I0` and `b0` were calculated with `M=Int(20)`. Some values can not be calculated because it has to be that $p > q$, so they are not defined (n.d.). Looking closer to the values we can find out that

$$\texttt{l(initial(P}, 1 + t \cdot q_1, q_1\texttt{)...)} \leq \texttt{l(initial(P}, 1 + t \cdot q_2, q_2\texttt{)...)}. \quad (27)$$

`initial`(p,q)	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$
$p=1$	n.d.	n.d.	n.d.	n.d.	n.d.
$p=2$	-9045.926	n.d.	n.d.	n.d.	n.d.
$p=3$	-8987.880	-9045.926	n.d.	n.d.	n.d.
$p=4$	-8960.650	-9003.681	-9045.926	n.d.	n.d.
$p=5$	-8946.797	-8976.492	-9012.699	-9045.926	n.d.
$p=6$	-8928.720	-8958.863	-8990.256	-9018.530	-9045.926
$p=7$	-8920.689	-8940.767	-8970.755	-9000.464	-9022.614
$p=8$	-8914.226	-8931.262	-8952.147	-8982.014	-9007.739
$p=9$	-8905.135	-8923.521	-8940.938	-8963.103	-8991.655
$p=10$	-8901.141	-8915.379	-8932.025	-8950.472	-8973.378

Table 2: `l` for different initial values.

for $q_1 < q_2$ but the same t. Inequality (27) also makes sense if we look again at the procedure how an initial value is calculated. Therefore, let $q_1 < q_2$, $p_k = 1+t\cdot q_k$ for $k = 1, 2$ and n_k be the number of categories of the histograms

used to calculate $\texttt{initial}(\texttt{P0}, p_k, q_k)$. It holds that

$$n_1 \underset{\text{Def.}}{=} p_1 - q_1 \underset{\text{Def.}}{=} 1 + (t-1)\,q_1 \underset{q_1 < q_2}{\leq} 1 + (t-1)\,q_2 \underset{\text{Def.}}{=} p_2 - q_2 \underset{\text{Def.}}{=} n_2.$$

So, the histogram used to calculate $\texttt{initial}(\texttt{P0}, p_2, q_2)$ has more categories and therefore leads to a better estimation. In other words, if we want to use basis splines with a higher degree, we also have to increase the number of basis splines to get still better solutions. For the rest of this thesis we will only compare solutions of different degree by taking the same t and $p_k = 1 + t \cdot q_k$ for $k = 1, 2, \ldots$, so our results will be reported not as pairs (p_k, q_k) but as pairs (t, q_k), as in Table 3.

$l\,(\texttt{initial}\,(p, q))$	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
$t = 1$	-9045.926	-9045.926	-9045.926	-9045.926	-9045.926
$t = 2$	-8987.880	-8976.492	-8970.755	-8963.103	-8959.772
$t = 3$	-8960.650	-8940.767	-8932.025	-8922.397	-8914.360
$t = 4$	-8946.797	-8923.521	-8906.264	-8899.908	-8889.419
$t = 5$	-8928.720	-8909.789	-8895.985	-8886.056	-8882.050

Table 3: `l` for different initial values ordered by (t, q_k).

3.4 Genetic algorithm for optimization

Besides `constrOptim` we want to compute a genetic algorithm of optimization because these algorithms are often better to find a global maximum than algorithms using gradient methods. For more information see [3]. The algorithm here is based on the idea of *swarm intelligence*: First we set an initial value $\eta^* \in \mathbb{R}_+^{3p}$ and then generate $2N$ particles η_i randomly chosen from area $F = \mathbb{R}_+^{3p} \cap (1 \pm h)\,\eta^*$. After that we set $g = \eta^*$, $G = l\,(\eta^*, \dots)$ and vector D with $D_i = l\,(\eta_{N+i}, \dots)$. G will be interpreted as the global maximum which was reached by at least one particle so far, and D_i as the individual maximum which was reached by each particle itself so far. In each iteration we are going through the set $\{\eta_1, \dots, \eta_N\}$ and comparing $d_i = l\,(\eta_i, \dots)$ with G and D_i. If $d_i > D_i$ we set $\eta_{N+i} = \eta_i$ and $D_i = d_i$. If also $d_i > G$, we set $g = \eta_i$ and $G = d_i$. In the end we update each η_i by

$$\eta_i = w_1 \cdot \underbrace{(\eta_{N+i} - \eta_i)}_{:=v_1} + w_2 \cdot \underbrace{(g - \eta_i)}_{:=v_2} + w_3 \cdot r, \tag{28}$$

where r is randomly chosen from the set $\{v | v = a \cdot v_1 + b \cdot v_2, 0 \leq a, b \leq 1\}$. So, each particle moves in the direction of its best individual position (v_1), in the direction of the best global position (v_2) and in a random direction in

between these two. The weights $0 \leq w_j \leq 1$ are adjusted at the beginning. The algorithm returns g when all particles are close enough to each other. The whole procedure is implemented as Function `opt` in the appendix.

3.5 Results

First we look at the likelihood function values of the results generated by `constrOptim` using the initial values of Table 3:

l (`constrOptim` (p, q))	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
$t = 1$	-8958.145	-8947.838	-8985.513	-8966.426	-9003.047
$t = 2$	-8973.315	-8961.603	-8955.014	-8944.304	-8940.630
$t = 3$	-8940.264	-8926.225	-8916.200	-8903.978	-8898.286
$t = 4$	-8931.290	-8912.719	-8895.984	-8887.176	-8877.553
$t = 5$	-8922.828	-8896.184	-8883.133	-8872.131	-8876.420

Table 4: `l` for the results of `constrOptim` ordered by (t, q_k).

Sometimes it is necessary to add a small number (i.e. 10^{-16}) to an entry of an initial value, otherwise `constrOptim` would not accept the input because $0 \notin \mathbb{R}_+$. We also give $-l$ and $-dl$ as input because `constrOptim` always tries to minimize, and the maximization of a function f is equivalent to the

minimization of $-f$. Comparing values from Table 4 with Table 3 we see that all of them have been improved, so in fact a maximization-process is in fact happening. Now we try function `opt`; we set as input $N = 300$ particles, $h = 25\%$ and weights $\omega_j = \frac{1}{3}$ for all j.

$l\,(\texttt{opt}\,(p,q))$	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$
$t=1$	-9025.413	-9031.659	-9031.582	-9036.228	-9034.539
$t=2$	-8958.860	-8953.851	-8949.213	-8943.480	-8940.477
$t=3$	-8933.926	-8921.908	-8912.149	-8896.244	-8895.509
$t=4$	-8923.981	-8902.244	-8884.411	-8882.399	-8874.017
$t=5$	-8907.143	-8889.035	-8876.822	-8868.573	-8866.713

Table 5: `l` for the results of `opt` ordered by (t, q_k).

Comparing Table 5 with Table 4 and Table 3 it holds for all $p = 1 + t \cdot q$ with $t > 1$ and all q that

$$\texttt{l(initial}(p,q)\texttt{)} < \texttt{l(constrOpt}(p,q)\texttt{)} < \texttt{l(opt}(p,q)\texttt{)}. \tag{29}$$

To get an idea of what is behind these values we take a look at an example of estimated densities. Therefore, let $q = 2$. Then the best result is reported

for $t = 5$ and therefore for $p = 1 + 5 \cdot 2 = 11$. By taking these values and plotting the densities for a, b and c, we can see that the results calculated with `opt` (red line) differ more from the initial value (blue line) than the results calculated with `constrOptim` (red dashed).

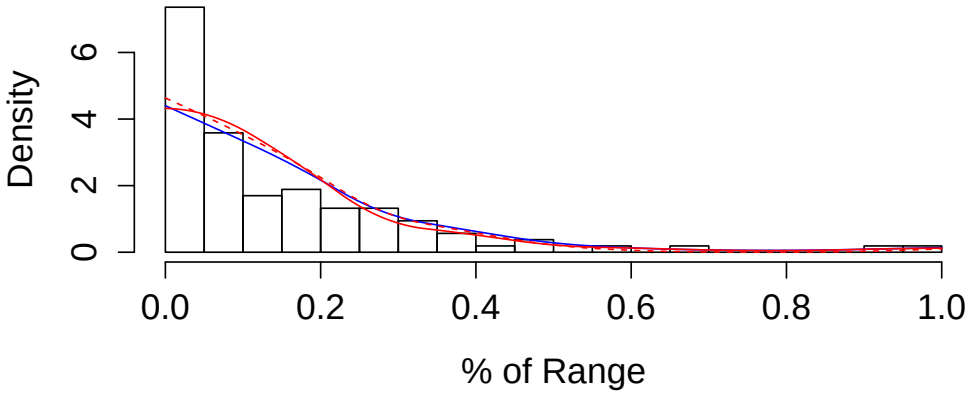

Figure 6: Densities of the initial value (blue), `constrOptim` (red dashed) and `opt` (red) for parameter a with $t = 5$ and $q = 2$.

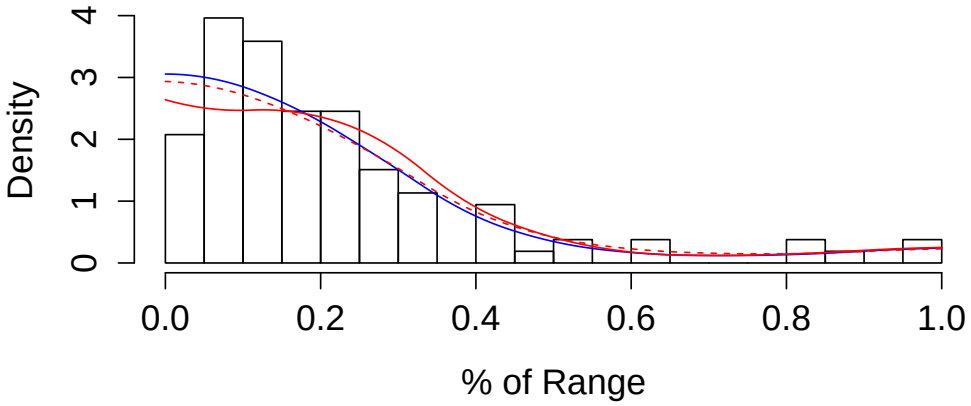

Figure 7: Densities of the initial value (blue), `constrOptim` (red dashed) and `opt` (red) for parameter b with $t = 5$ and $q = 2$.

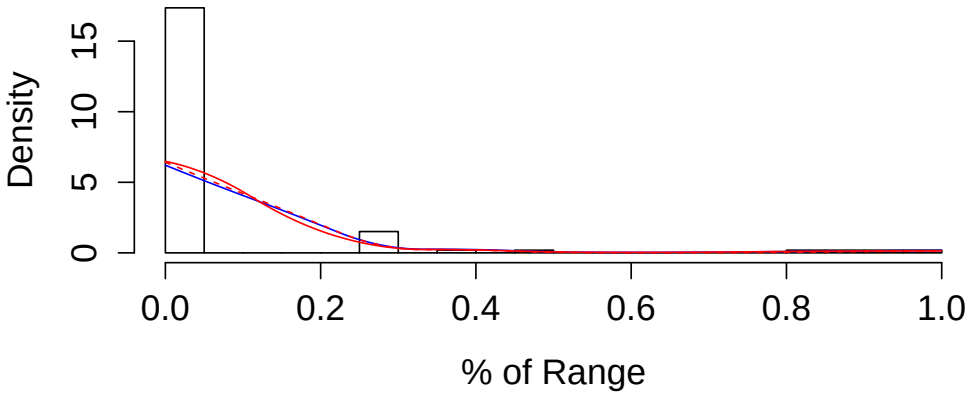

Figure 8: Densities of the initial value (blue), `constrOptim` (red dashed) and `opt` (red) for parameter c with $t = 5$ $q = 2$.

3.6 Simulations

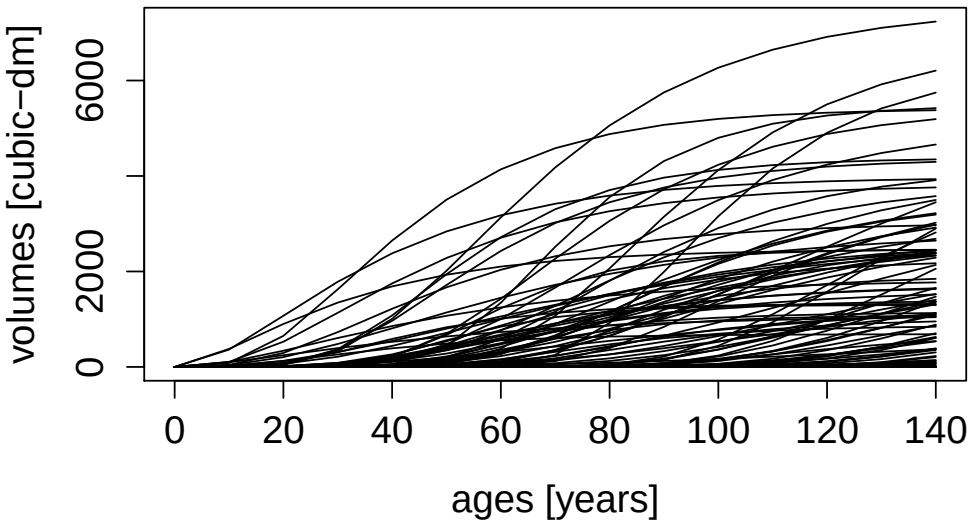

Figure 9: Simulation of tree data.

We will now run a simulation to verify that if our algorithm reaches the real densities. To do so, we use the same ranges from `P0` but choose $u \sim B(2,8)$, $v \sim B(3,7)$ and $w \sim B(1,9)$ with $a = \alpha + u \cdot d_\alpha$, $b = \beta + v \cdot d_\beta$ and $c = \gamma + w \cdot d_\gamma$. We can do so because Beta-distributed random variables only take values in $[0,1]$. The generating function of this simulation is implemented as function `simulation` in the appendix and takes as input `P0`, N for the number of trees being simulated, m for the number of observations per tree, t for the

time between two observations and all B-parameters. As output we get each parameter per tree and the original ranges in the same order, as it is in the output of function `P`. We can also simulate the tree data themselves from this output easily. With both outputs we can proceed like we did before with our original data but now we can compare our solutions with the real densities. As an example for this we look at the results for $t = 5$ and $q = 2$ again. Here the color of the real density is purple and the histograms in the backgrounds are also changed to new histogram showing the empirical density of the simulated u, v and w.

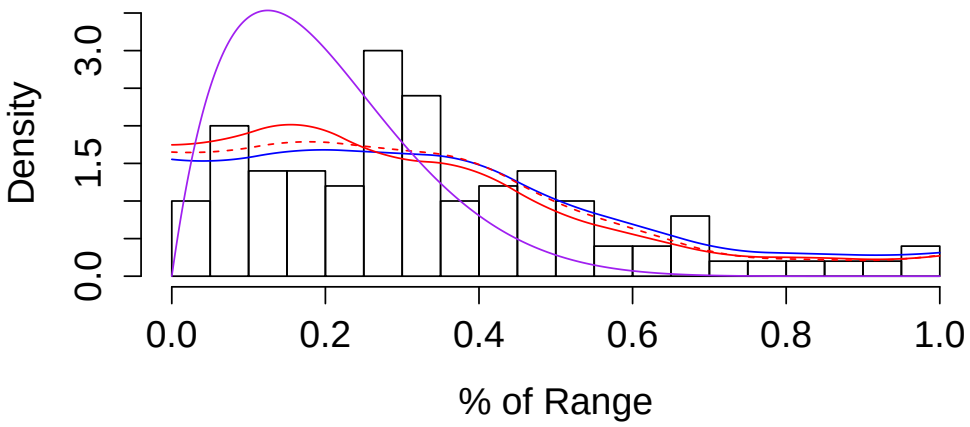

Figure 10: calculated densities of the initial value (blue), `constrOptim` (red dashed) and `opt` (red) and real density (purple) for parameter a with $t = 5$ and $q = 2$.

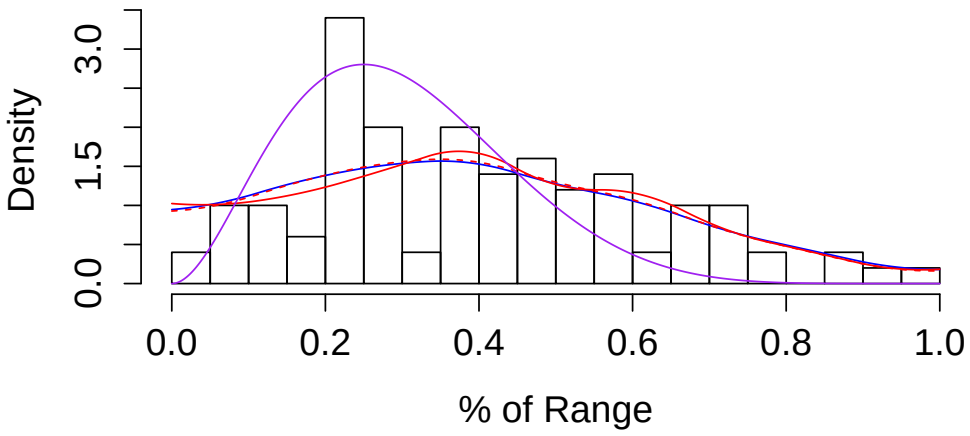

Figure 11: Densities of the initial value (blue), `constrOptim` (red dashed) and `opt` (red) and real density (purple) for parameter b with $t = 5$ and $q = 2$.

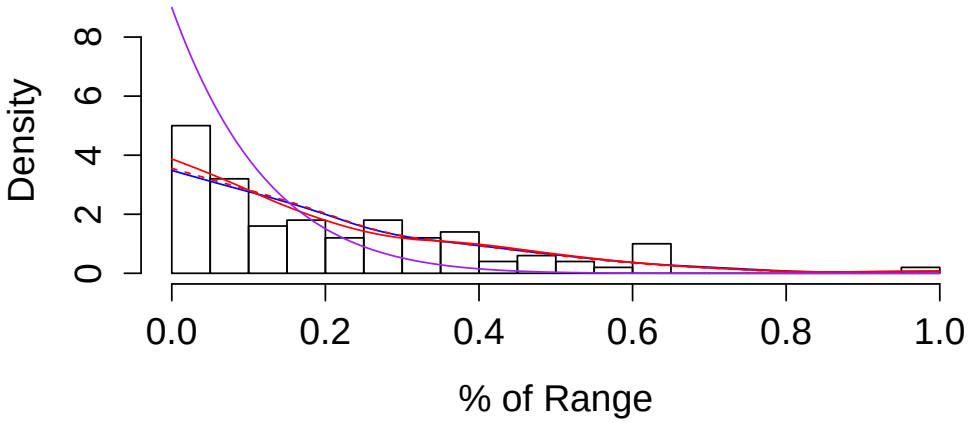

Figure 12: Densities of the initial value (blue), `constrOptim` (red dashed) and `opt` (red) and real density (purple) for parameter c with $t = 5$ $q = 2$.

4 Outlook

4.1 Conclusions

Our goal was to find distributions of our model parameters a, b and c. We accomplished this goal by estimating the densities of these distributions with our algorithms and our calculated log-likelihood function. However, considering the last section about simulation and also looking at our exemplary plots of our results, the estimated densities are closer to the histograms than the real density. The reason therefore can be found in the calculation of the initial value: η is first set to be as close as possible to the given histograms. But in the case that the histograms are not a good estimation of the real density this is not efficient. In fact, the initial value can be interpreted as an estimation of an estimation of the real density because the histogram is the discrete estimation of the continuous density. On the other hand, updating the initial value could be the key for solving that problem; we can be sure that the real optimum has been estimated to the degree that the initial value has been updated. Considering this, the generic algorithm `opt` provides better solutions than `constrOpt`.

4.2 open problems

One suggestion for future research would be to investigate how the input values of `opt` influence the output. It would be reasonable that more particles lead to a better estimation. If we increase h and therefore the area from which the first particles are chosen we may also get better results because more different points would become part of our optimization process. Anyway we should keep in mind that all optimal solutions are on a ray and therefore we are not only interested in how big this area is, but if it has a good position in $\mathbb{R}_+^{3p}$ to reach that ray. Also, by changing weights w_j, we may get better results. To answer these questions, lots of different combinations of (t, q) have to be investigated, and their results should be compared based on different inputs for `opt`. Also better results would maybe generated if we use another estimator of σ^2 or tread σ^2 as an parametrr like η; in this case $l(\eta, \sigma^2)$ has to be maximized. The choice of θ_0 for the curve fitting in the parametrization also provides opportunities for further research; all calculations of the histograms and the computations of the initial values are based on the estimation of the ranges for the model parameters. Therefore, the effects of the inputs for the curve fitting are also topics for further research.

R-Code

```
### functions for calculations ###


#1 numbers of observations per tree

L=function(B){
  l=rep(0,length(B))
  for(k in 1:length(B)){
    l[k]=length(B[[k]][1,])
  }
  return(l)
}


#2 richards-curve

g=function(M,z){
  x=rep(z,length(z),each=length(M[,1]))
  a=rep(M[,1],length(z))
  b=rep(M[,2],length(z))
  c=rep(M[,3],length(z))
  return(matrix(a*(1-exp(-b*x))^c,length(M[,1]),length(z)))
}


#3 integrationmatrix

Int=function(N){
  return(matrix(c(rep(seq(0.5/N,1-0.5/N,1/N),N^2),
                  rep(sort(rep(seq(0.5/N,1-0.5/N,1/N),N)),N),
                  sort(rep(seq(0.5/N,1-0.5/N,1/N),N^2))),N^3,3))
}
```

```
#4 loss-function

fkt=function(p,A){
  M=matrix(p,1,3)
   return(sum((A[1,]-g(M,A[2,]))^2))
}


#5 curve-fitting

P=function(p0,B){
   a=rep(0,length(B))
   b=rep(0,length(B))
   c=rep(0,length(B))
   for(k in 1:length(B)){
      p=optim(p0,fkt,A=B[[k]])[[1]]
      a[k]=p[1]
      b[k]=p[2]
      c[k]=p[3]
   }
   return(list(a,b,c,matrix(c(min(a),max(a)-min(a),
                              min(b),max(b)-min(b),
                              min(c),max(c)-min(c)),3,2,TRUE)))
}


#6 histogram

Hst=function(P,k,N){
   return(hist((P[[k]]-min(P[[k]]))/(max(P[[k]])-min(P[[k]])),
               seq(0,1,1/N),
               freq=FALSE,xlab="% of Range",main="")$counts)
}
```

```
#7 R-version of h_i

h=function(M,k,P,B){
   A=M
   A[,1]=P[1,1]+M[,1]*P[1,2]
   A[,2]=P[2,1]+M[,2]*P[2,2]
   A[,3]=P[3,1]+M[,3]*P[3,2]
   H=(matrix(B[[k]][1,],length(A[,1]),length(B[[k]][1,]),TRUE)
       -g(A,B[[k]][2,]))^2
   return(rowSums(H))
}




#8 matrix H=(h_1,...,h_N)

H=function(M,P,B){
   K=matrix(0,length(M[,1]),length(B))
   for(k in 1:length(B)){
     K[,k]=h(M,k,P,B)
   }
   return(K)
}




#9 R-version of s

s=function(M,P,B){
   l=1/L(B)
   return(as.vector(H(M,P,B)%*%l)/length(B))
}
```

```
#10 R-version of I

I=function(M,P,B){
  l=L(B)
  s=s(M,P,B)
  H=H(M,P,B)
  A=matrix(0,length(M[,1]),length(B))
  for(k in 1:length(B)){
    A[,k]=(1/sqrt(2*pi*s))^l[k]*exp(-(H[,k]/s))
  }
  return(A)
}



#11 basis-spline function

bsplines=function(x,p,q){
  n=p-q
  A=matrix(rep(n*x,p)-rep(c(1:p),each=length(x)),length(x),p)
  B=matrix(0,length(x),p)
  for(k in 0:(q+1)){
    B=B+ifelse(A-k+q+1>0,choose(q+1,k)*(-1)^k*(A-k+q+1)^q,0)
  }
  return(B/factorial(q))
}



#12 basis-spline integrals

bint=function(p,q,m){
  x=seq(1/m,1,1/m)
  return(colMeans(bsplines(x,p,q)))
}
```

```
#13 basis-splines of a combination

b=function(M,p,q,m){
  b0=list()
  b0[[1]]=bsplines(M[,1],p,q)
  b0[[2]]=bsplines(M[,2],p,q)
  b0[[3]]=bsplines(M[,3],p,q)
  b0[[4]]=bint(p,q,m)
  return(b0)
}


#14 density of a single parameter

dstb=function(k,v,b){
  return(as.vector(b[[k]]%*%v)/sum(v*b[[4]]))
}


#15 joint density of all parameters

f=function(v,b){
  m=length(v)/3
  return(dstb(1,v[1:m],b)
         *dstb(2,v[(m+1):(2*m)],b)
         *dstb(3,v[(2*m+1):(3*m)],b))
}


#16 likelihood function

l=function(p,I,b){
  f=f(p,b)
  return(sum(log(as.vector(f%*%I)/length(I[,1]))))
}
```

```
#17 gradient of the likelihood function

dl=function(p,I,b){
  n=length(b[[4]])
  f=f(p,b)
  d=1/(as.vector(f%*%I)/length(I[,1]))
  d1=(t(b[[1]]*dstb(2,p[(n+1):(2*n)],b)*dstb(3,p[(2*n+1):(3*n)],b))
      %*%I)/length(I[,1])
  D1=(d1%*%d-(length(I[1,])*b[[4]]))/sum(p[1:n]*b[[4]])
  d2=(t(dstb(1,p[1:n],b)*b[[2]]*dstb(3,p[(2*n+1):(3*n)],b))
      %*%I)/length(I[,1])
  D2=(d2%*%d-(length(I[1,])*b[[4]]))/sum(p[(n+1):(2*n)]*b[[4]])
  d3=(t(dstb(1,p[1:n],b)*dstb(2,p[(n+1):(2*n)],b)*b[[3]])
      %*%I)/length(I[,1])
  D3=(d3%*%d-(length(I[1,])*b[[4]]))/sum(p[(2*n+1):(3*n)]*b[[4]])
  return(c(D1,D2,D3))
}


#18 initial value

initial=function(P,p,q){
  n=p-q
  V1=matrix(c(rep(c(rep(Hst(P,1,n)[1],q),
                    Hst(P,1,n),rep(Hst(P,1,n)[n],q)),n+q),
                    rep(0,n+q)),n+2*q+1,n+q)
  v1=colSums(V1[1:(q+1),])
  V2=matrix(c(rep(c(rep(Hst(P,2,n)[1],q),
                    Hst(P,2,n),rep(Hst(P,2,n)[n],q)),n+q),
                    rep(0,n+q)),n+2*q+1,n+q)
  v2=colSums(V2[1:(q+1),])
  V3=matrix(c(rep(c(rep(Hst(P,3,n)[1],q),
                    Hst(P,3,n),rep(Hst(P,3,n)[n],q)),n+q),
                    rep(0,n+q)),n+2*q+1,n+q)
  v3=colSums(V3[1:(q+1),])
  v=c(v1,v2,v3)
  return(v/max(v))
}
```

```
#19 genetic algorithm

Opt=function(P,q,b,I,N,h,u,v,w,O,tol){
  p=initial(P,length(b[[4]]),q)
  n=3*length(b[[4]])
  A=matrix(runif(2*n*N,pmax(p*(1-h),0),p*(1+h)),N,2*n,TRUE)
  G=l0(p,I,b)
  g=p
  D=rep(0,N)
  for(k in 1:N){
    D[k]=l0(A[k,(n+1):(2*n)],I,b)
  }
  Z=0
  while(max(abs(D-G))>abs(tol*G)){
    if(Z>O){return(list(p,A,D,G,g,Z))}
    for(k in 1:N){
      d=l0(A[k,1:n],I,b)
      if(d>D[k]){
        D[k]=d
        A[k,(n+1):(2*n)]=A[k,1:n]
      }
      if(d>G){
        G=d
        g=A[k,1:n]
      }
      A[k,1:n]=abs(A[k,1:n]+(u/(u+v+w))*(g-A[k,1:n])
                   +(v/(u+v+w))*(A[k,(n+1):(2*n)]-A[k,1:n])
                   +(w/(u+v+w))*runif(n,pmin(g-A[k,1:n],
                                             A[k,(n+1):(2*n)]-A[k,1:n]),
                                             pmax(g-A[k,1:n],
                                             A[k,(n+1):(2*n)]-A[k,1:n])))
    }
    Z=Z+1
  }
  return(list(p,A,D,G,g,Z))
}
```

```
#20 simulation of a data-set

simulation=function(P,N,m,t,c1,c2,c3){
  S=list()
  u=rep(0,N)
  v=rep(0,N)
  w=rep(0,N)
  for(k in 1:N){
    u[k]=rbeta(1,c1[1],c1[2])
    v[k]=rbeta(1,c2[1],c2[2])
    w[k]=rbeta(1,c3[1],c3[2])
    S[[k]]=matrix(0,2,m)
    a=P[1,1]+P[1,2]*u[k]
    b=P[2,1]+P[2,2]*v[k]
    c=P[3,1]+P[3,2]*w[k]
    S[[k]][2,]=seq(0,t*(m-1),t)
    S[[k]][1,]=a*(1-exp(-S[[k]][2,]*b))^c
  }
  datplot(S)
  return(list(u,v,w,P))
}



### OUTPUT EXAMPLE OF INT(2) ###

> Int(2)
     [,1] [,2] [,3]
[1,] 0.25 0.25 0.25
[2,] 0.75 0.25 0.25
[3,] 0.25 0.75 0.25
[4,] 0.75 0.75 0.25
[5,] 0.25 0.25 0.75
[6,] 0.75 0.25 0.75
[7,] 0.25 0.75 0.75
[8,] 0.75 0.75 0.75
```

References

[1] Owen Jones, Robert Maillardet, and Andrew Robinson. *Introduction to Scientific Programming and Simulation Using R, Second Edition* -. CRC Press, Boca Raton, Fla, 2 rev ed. edition, 2014.

[2] Mary J. Lindstrom. Self-modelling with random shift and scale parameters and a free-knot spline shape function. *Statistics in Medicine, Vol. 14, 2009-2021*, 1995.

[3] Shi Yuhui. *Emerging Research on Swarm Intelligence and Algorithm Optimization* -. IGI Global, Hershey, 2014.