Functional Linear Regression on Namibian and South African Data

Nosipho Mzimela

Supervisor:
A/Prof Sugnet Lubbe

A thesis submitted for the degree of
Master of Commerce

Department of Statistical Sciences
University of Cape Town
2016

Acknowledgements

I would like to express boundless gratitude to my supervisor A/Prof Sugnet Lubbe for all her support and commitment throughout my dissertation. I would also like to thank Mr Sam Jack and Prof Timm Hoffman from the Plant Conservation Unit at the University of Cape Town, for providing me with all the Aloe Tree datasets. Lastly, I would like to extend my appreciation to the researchers at the Global Historical Climatology Network (GHCN) for the data sets containing the climate records used in the analysis.

Abstract

Indigenous to Southern Africa, the Aloe Dichotoma, most commonly known as the Quiver tree, are species of Aloe found mostly in the southern parts of Namibia and the Northern Cape Province in South Africa. Researchers noticed that Quiver trees assumed very different shapes depending on their geographical location. This project aims to model the observed differences in structural form of the trees between geographically spate populations with functional regression analysis using climate variables at each location. A number of statistical challenges present themselves such as the multivariate nature of the data. Functional data analysis was used in this project to display the data so as to highlight various characteristics while allowing us to study important sources of pattern and variation among the data. Functional data analysis can be best summarised as approximating discrete data with a function by assuming the existence of a function giving rise to the observed data. The underlying function is assumed to be smooth such that a pair of adjacent data values are necessarily linked together and unlikely to be too different from each other. There are a number of smoothing methods used to fit a function to the discrete data. In this project we use Roughness Penalty Smoothing methods which are based on optimising a fitting criterion that defines what a smooth of the data is trying to achieve. The meaning of smooth is explicitly expressed at the level of the criterion being optimised, rather than implicitly in terms of the number of basis functions used. Once the continuous functions for the climate variables have been fitted, these are used as predictors in a functional regression model with the structural variables as responses. This allows for the estimation of regression coefficients to describe the effect of the climate variables on each structural variable.The functional models suggest that maximum temperature has an effect on the structural form of Aloe Dichotoma. Further, the structural form of Aloe Dichotoma does differ in geographically spate locations. Trees found in the warmer Northern regions are more likely to have taller trees. The results did not necessarily prove the hypothesis that the number of branches found on trees in the North is fewer than those in the South, but these trees are more likely to have more dichotomous branches which may be translated to more branches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter discusses some history on Aloe Dichotoma and past weather trends in the areas in which you would find these trees. Researchers have speculated on the relationship between climate and Aloe Dichotoma and the intention of this dissertation is to test the hypothesis these researchers have. It is believed that trees in the northern regions are taller with fewer branches. A relationship between the weather in these regions and the structural form of the trees is implied by their speculation. We then give a brief roadmap of the dissertation.

The Quiver tree that grows in the arid parts of Namibia and South Africa is a kind of aloe. Similar to the rest of the aloe family, it has thick succulent leaves growing in a rosette, each at the end of a stumpy branch. The succulent leaves allow the Quiver trees to escape the worst of the devastating heat and reduce the amount of moisture inevitably lost by evaporation from the surface of their leaves. Quiver trees have branches that are thickly covered in a fine white powder to reflect the sun's heat rather than to absorb it. This white powder helps in keeping the Quiver tree cool. The fiber filled branches and trunk are the defining characteristic of the Quiver tree. The fiber in the branches and trunk are the reason Quiver trees have great storage capacity (Attenborough, 1995).

Weather stations report that the average temperatures in the Quiver tree regions have increased in recent decades and the increase is predicted to continue into the future and rainfall will decrease (IUCN, 2009). Large die offs were occurring amongst the Quiver tree population by 2001. Scientists found most of these die offs to be in the hotter equator-ward areas of the range of Quiver trees where the Quiver trees are subjected to drought stress. The cooler regions farther from the equator and on high mountaintops had populations that were growing and reproducing well. The climate changes as a result of global warming have resulted in the Quiver trees shifting their range to higher latitudes and higher

altitudes where conditions are generally cooler and moister (IUCN, 2009).

Previously work was done as a first step into exploring the relationship between Quiver trees in geographically spate locations. The relationship between the structure and health of the Quiver trees with underlying climate conditions was also explored. Four statistical methods were applied to this analysis. Fisher's Linear Discriminant analysis was used to discriminate by location, with the aim of allocating each observation into its respective group, in this case, location. Variable selection was used to select which variables best indicate the differences in the populations. Multiple linear regressions were applied to a number of structural and health variables to assess the relationship between these variables and longitude-latitude coordinates as well as aspect and slope. The relationship analysis was taken a step further by the use of Generalized Additive Models (GAM). GAM provided improved insight on the relationship between the variables in question and the longitude-latitude coordinates, aspect and slope. GAM is a more flexible method that does not assume a linear relationship. A Biplot was used to represent the samples with alpha bags used to show the groups. To relate the differences in the structure and health of the Quiver trees, Partial least squares regression was used due to the high correlation of the climate variables observed (Mzimela, 2013).

The results of this analysis were in line with the researchers findings showing a significant relationship between locations with taller trees in the north and shorter trees with fewer branches in the south. The structural variables discussed in both analyses were total height, height of first branch, canopy diameter, basal circumference, circumference at first branch, branches off main stem and number of dichotomous events. All these variables were significantly different by location in terms of longitude and latitude. The slope was found to only have effects on height at first branch and number of branches off the main stem, while aspect has a significant effect on total height, canopy diameter, basal circumference, circumference at first branch and number of dichotomous events (Mzimela, 2013). In this thesis the analysis is taken a step further using functional data analysis

In this analysis we used basis functions to transform the discrete data to functions, discussed in chapter 4. 1. We look at two different basis function systems, namely, Fourier series and B-spline series. Fourier basis systems are used on data where the times of observations of the data are equally spaced. Temperature data would usually fall under this category but in our case the temperature data was recorded intermittently over several years and the focus is not specifically on periodic fluctuations. B-splines are polynomial segments that are joined at the ends and do not require the data to be equally spaced. The objective when approximating discrete data is that the continuous function should exhibit similar features to the data it is approximating. Controlling smoothness by limiting the number of basis functions is discontinuous. Using Roughness penalty smoothing with basis functions allows for continuous control over smoothness. This is discussed in Section 4.

2. The Roughness penalty smoothing method fits the data using a differential equation and filters out noise whilst keeping the bias in the model low. This is the reason it is our method of choice as a more powerful method for approximating discrete data using a function.

We then explore methods to choose a smoothing parameter λ, which is the key input used to smooth the data, discussed in section 4. 3. The methods explored in this analysis are Generalized Cross-Validation (GCV) and Generalized Information Criterion (GIC). GIC is the extension of the AIC and is a generalized version. The general information criterion relaxes the assumptions that (i) estimation is by maximum likelihood, and that (ii) this is carried out in a parametric family of distributions including the true model (Konishi and Kitagawa, 2008). In this analysis we found that GCV produced better fitting λ values, which we used in the remainder of the analysis. The GCV aims to estimate a well fitting function that minimizes the error sum of squares without under-smoothing by double discounting the mean squared error. Once we have our smooth functions with similar features to the underlying discrete data we perform functional regression analysis to see the effects of the functional predictor variables on our scalar responses. The analysis is presented in Chapter 5 and the results discussed in Chapter 6.

1.1 Objectives

This dissertation looks to test the hypotheses researchers have, regarding the structural form of the Quiver trees in geographically separated populations. It is believed that taller trunks with fewer branches characterize the Quiver tree populations found in the hot dry northern regions. Shorter trunks that are rounder for better seed production characterize the Quiver trees found in the Southern regions. Thus, the question we try to answer is whether or not there are significant differences between Quiver trees in geographically separated locations. In particular, the objectives are divided into:

- Provide thorough analysis of methods used to estimate discrete data using a function. The emphasis here is on using functions that fit the data well and keep bias low;
- Provide comparisons of different methods used to estimate the functions and choose the smoothing parameter;
- Provide estimated predictor functions for each of the predictors used in the analysis, on each of the response variables;

- Provide functional regression models that approximate the relationship between the predictors and each response;
- Identify patterns across the different locations where temperature and rainfall had similar effects on the structural variables(predictors) against the similarities in the geographic aspects of each location which could explain this relationship.

1.2 Notation

The following notation was used throughout:

n	number of samples / observations
y	n $\times$ 1 vector of observed regression response values for n sample points (tree structure variable)
x	n $\times$ 1 vector of observed predictor variable (climate variable)
t_j	time point j where an observation was made
x_j	observed value at time point t_j
$x(t_j)$	function value of x at time t_j
$x(\mathbf{t})$	function values along all observed time points $t_1, \ldots, t_p$
$\beta(t)$	regression coefficient function value at time t
K	number of basis functions
$\phi_k(t)$	value of k-th basis function at time t
$\mathbf{\Phi}(t)$	$K \times 1$ vector of values of the set of K basis functions at time t
$\mathbf{\Phi}$	$n \times K$ matrix of values of the set of K basis functions at time t in row i for the ith observation
$\mathbf{c}$	$K \times 1$ vector of coefficients

This chapter served to set the scene and outline the topics discussed in this dissertation. In answering the research question, the dataset used is very important in verifying the credibility of the analysis and quality of results produced. Chapter 2 provides details on the data used, particularly how and where it was collected as well as other details that made the analysis possible.

Chapter 2

Data

The first set of data consists of 3368 observations on Aloe Dichotoma in parts of Nambia and South Africa and will be referred to as tree data. The tree data reports on 37 variables as well as the location of each observation and the date observed. In this analysis we consider seven of these variables. This data was collected by our researchers intermittently over approximately 1 year over the period June 2008 to September 2009,where each tree was visited once. The total set of variables can be found in Table 2.1. The data is separated by location,where there were 14 Locations observed in the analysis. These locations are shown in Table 2.2 .

Table 2.1: Table Showing Structural Variables Observed

Variable Name
Total Height
Height of First Branch
Canopy Diameter
Basal Circumference
Circumference at First Branch
Branches Off Main Stem
Number of Dichotomous Events

Table 2.2: Table Showing All Locations Considered

Tree Location
Brandberg
Omaruru River
Spitzkoppe
Tinkas River
Remhoogte
Hauchabfontein
Gorab
Namtib
Carolinahof
Kliphoek
Grunau
Bulletrap
Rooifontein
Gannabos

The second set of data consists of three climate variables, namely minimum and maximum temperature as well as rainfall, taken from six weather stations. We use the climate variables to relate the differences in the structure of the trees between the different locations to the underlying rainfall and temperature measures. This data set consists of weather data over a 19 year period from January 1996 to December 2014, obtained from the Global Historical Climatology Network Daily(GHCND). GHCND is a database that contains historical daily temperature, rainfall and snow records over global land areas. Numerous sources are used to collect these measurements which then undergo comprehensive quality checks. The weather stations are generally identified by city or airport name in this dataset. Latitude, longitude and elevation above mean sea level are used to describe the geographic location. The temperature is recorded in celsius degrees to tenths and the rainfall in tenths of mm. The full set of weather stations considered in this dissertation are shown in Table 2.3

Table 2.3: Table Showing Full Set of Weather Stations

Weather Station	Latitude	Longitude	Elevation
Omaruru	-21.42	15.93	1 217
JGH Van der Wath	-26.53	18.12	1 077
Vioolsdrif	-28.70	17.60	168
Niewouldville	-31.37	19.12	719
Gobabeb	-23.57	15.05	400
Springbok	-29.67	17.90	1 007

The longitude and latitude of each tree was noted. These are plotted on a scatter diagram with different colours used for different locations. The R function 'PlotOnStaticMap' in the R package *RgoogleMaps*(Loecher.M, Ropkins.K.2015) allows one to import a Google map as a background. The resulting plot is found in Figure 2.1.

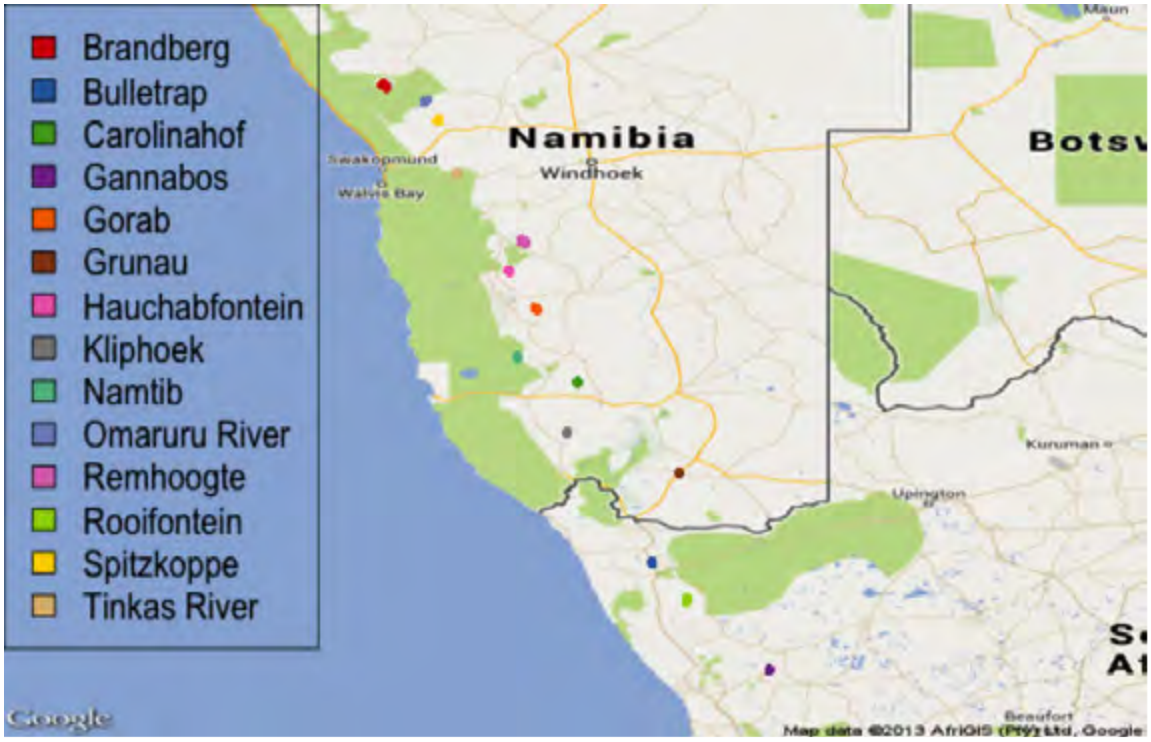


Figure 2.1: Graphical Representation of Tree Locations

Now that we understand the data , in Chapter 3 we review the literature around the methods we used for estimating functional predictor variables used in our functional regression

models. Here we discuss literature around regularization,in particular, in functional regression modelling as well as some of its other uses.

Chapter 3

Literature Review

In this chapter we discuss literature around the methods used in this paper.We briefly discuss the origins of functional data analysis and the different approaches used to estimate the response variables from naive, less sophisticated methods such a discretization to more robust methods such as regularization. We then discuss different uses of regularization in regression models.

Functional linear regression with a scalar response is one of the more widely explored topics within the relatively new study of functional data analysis. To this end there are a number of different methods that have been established and documented. The term functional data analysis, from which functional linear regression is derived, was first coined by Ramsey and Dalzell(1991) describing the use of 'functional' data rather than discrete data to make statistical inferences.

Hastie and Mallows(1993) and Ramsey and Silverman(1997) discuss a naive approach to estimating a scalar response with functional approaches using discretization of the covariate function. The idea here is to evaluate each observation at each time point as a separate covariate before doing a regular regression. The problem with this, highlighted by these authors is that the resulting functions tend to have high correlation, consequently leading to uninterpretable models.There is an infinitely large number of solutions produced by this method, all giving reliable prediction of the observed data. Furthermore, this method uses a significant number of degrees of freedom.

Ramsey and Silverman(2002) further demonstrate that the β coefficient function is likely to be under-determined because the the infinite number of parameters created by discretization is met by a finite number of responses to approximate. The infinite dimensional nature of the resulting discretized functions means that minimizing the residual sum of squares alone wont produce meaningful interpretation of the the β coefficients. Ramsey and Silverman(2002) suggest that this essentially necessitates the use of roughness penalties where

functional covariates are involved to produce interpretable results.

Ramsey and Silverman(2002) noted that the use of basis function expansions is helpful in reducing the degrees of freedom in the model even further. Hastie and Mallows(1993) had similar views on the use of smooth basis expansions on the β functions to estimate interpretable responses.

Marx and Eilers(1999) introduce a difference penalty in a log-likelihood criterion in the context of smoothed generalized linear regression as well as the use of B-spline expansion for the β coefficient functions.Hastie and Mallows(1993) similarily introduce a smooth basis expansion in a least squares criterion with a roughness penalty. As suggested by Ramsey and Silverman(1997) introducing a roughness penalty in the least squares criterion allows a satisfactory level of smoothness in the estimated functions. The estimator proposed by Hastie and Mallows(1993) minimizes a penalized least squares criterion resulting in a cubic spline.

Marx and Eilers(1999) consider the case of a generalised linear regression where many of the regressors are highly correlated. An example used is that of digitized points of a curve on a temporal domain. The high correlation and the probability that the number of regressors is larger than that of observations suggests the use of regularization. They solve collinearity by forcing β to be smooth of the temporal domain. When the β coefficient vector is projected onto a moderate number of B spline, dimension reduction is achieved. Marx and Eilers(1999) introduce a P spline which is described as a difference penalty between B spline coefficients used to achieve further smoothness.

Choosing the number and position for B-spline knots is quite an involved task. Marx and Eilers(1996) propose the use of a large number of knots and a difference penalty. This method is similar to the one discussed in this paper using smoothing splines.

Araki et al (2007) considered the problem of constructing functional regression models for scalar responses and functional predictors using Gaussian basis functions along with the technique of regularization. Regularized functional basis expansions are believed to be advantageous to functional data analysis in that it creates a more flexible instrument for transforming each individual observations into functional form.

James (2002) shows how functional principal components can be used to gain insight into the relationship between the response and functional predictors. Muller and Stadtmuller (2005) propose a linear predictor obtained by forming the scalar product of the predictor function with a smooth parameter, and the expected value of the response is related to this linear predictor via a link function.

In the context of this dissertation regression analysis using functional data was applied to the prediction of the structural form of Aloe Dichotoma from the pattern of temperature variation over many years. It is apparent that some form of regularization is essential to get

functions which are interpretable. The uses of regularization are wide and varied making it a robust approach to functional analysis.

After reviewing methodology considered in literature we choose methods that analyse the data in a way that we determine to answer the research question most effectively. These methods discussed in chapter 4, are used to estimate the functional regression models which we use as our primary method of observing the results produced by this methodology.

Chapter 4

Methodology

This chapter discusses in detail the methodology and processes used to transform the data from discrete to functional and how we went about constructing the different inputs used in the functional regression models. Here we explored different methods in detail and motivated for the methods we used namely; basis functions and regularization to convert the data to functional parameter objects and then the process of smoothing and choosing a smoothing parameter. We discuss in detail the functional regression models used and how linear regression is modified to accommodate functional predictors.

The focus of this dissertation is on functional linear models. A classic linear regression is usually defined as follows:

$$y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i \quad i = 1, \ldots, N \tag{4.1}$$

A functional regression model implies that one or more independent variables that are functional as opposed to discrete are used to predict the values of response variables y_i. The response variables can be functional as well or scalar. The latter is true for this analysis.

When functional independent variables are used, the discrete independent observations are replaced by functions of the form $x_i(t)$ such that the equation becomes:

$$y_i = \alpha_0 + \sum_{j=1}^{q} x_i(t_j)\beta_j + \epsilon_i \tag{4.2}$$

A naive approach would be to discretize each of the N functional independent variables by treating each independent variable at each time point as a separate independent variable. The problem with this approach is the limitation on $q < N$. The smaller the differences in the time points become Equation 4.2 becomes:

$$y_i = \alpha_0 + \int x_i(t)\beta(t)dt + \epsilon_i \tag{4.3}$$

Herein lies the problem; there is a finite number of independent variables with which to determine $\beta(t)$ which is inherently infinite dimensional. The larger more important problem produced by this method are the uninterpretable betas, and subsequently underdetermined models, produced due to the fact that their are essentially many regression coefficient functions $\beta(t)$ that predict the exact same response variable value. To produce interpretable beta values one must use some form of regularization. The regularization method used in this analysis is the Roughness Penalty Smoothing method.

The solution calls for the redefinition of the problem using basis coefficient expansions of β. With this, β is now defined as follows:

$$\beta(t) = \sum_{k}^{K} c_k\phi_k(t) = \mathbf{c}'\phi(t) \tag{4.4}$$

In this analysis, we use the function 'fRegress' in the R package fda(Ramsay.J,2014). This function makes use of the following inputs; y_i is a vector of N scalar responses, x_i is a list of functional predictors and $\beta(t)$ is a list of functional regression coefficients.

As previously mentioned, the coefficients of our functional regression models were estimated using roughness penalty smoothing. Using roughness penalty smoothing gives more control over the smoothing process and allows for the dimension K in Equation 4.3 to be large relative to N. This relationship means that more of the important features of the discrete data are included which may have been missed by using a low dimension K. Low dimensional coefficient estimates may also include irrelevant features of the discrete data if the basis functions are too small. Using regularization results in more reliable and more interpretable regression coefficients β.

The following sections will discuss these methods in more detail.

4.1 Basis Functions

There are several methods used to transform discrete data to functions. Basis function expansion is one of the more popular techniques used to achieve this. In this paper two basis function systems will be discussed, Fourier bases and B-spline bases with a strong focus on B-splines as the system of choice in the analysis.

A basis function system is a set of known functions ϕ_k that are mathematically independent of each other, and can be extended to include any number K in the system. A function $x(t)$ is constructed as a linear combination of these basis functions and can be expressed as

$$x(t) = \sum_{k=1}^{K} c_k \phi_k(t) \tag{4.5}$$

in terms of K known basis functions ϕ_k, and where ϕ_k are the basis functions used in constructing the functions.

c_k are the coefficients associated with these basis functions for each corresponding function. The coefficients of a basis system are calculated by solving a system of linear functions, with one equation for each basis function.

We can also express Equation 4.5 in matrix notation as

$$x(t) = \mathbf{c}'\boldsymbol{\phi}(t) = \boldsymbol{\phi}'\mathbf{c} \tag{4.6}$$

Where $\mathbf{c}$ is a vector of length K of the coefficients c_k and ϕ is a functional vector whose elements are the basis functions ϕ_k. The $n \times k$ matrix containing the values $\phi_k(t_j)$ is $\mathbf{\Phi}$.

In effect, basis expansion methods represent the potentially infinite-dimensional world of functions within the finite-dimensional framework of vectors like $\mathbf{c}$. It is important to note that although the dimension of the expansion is K, this does not mean that functional data analysis can be simply reduced to multivariate data analysis. The way the basis system, ϕ, is chosen, plays a pivotal role.

The number of basis functions, K, determines the degree to which the data x_j are smoothed as opposed to interpolated. The coefficients c_k can be chosen to yield $x(t_j) = x_j$ for each j when $K = n$, the number of parameters estimated by each curve. To this end, when

defining a basis system, K itself is seen as a parameter, which is chosen based on the characteristics of the data as opposed to being a fixed number of parameters.

Determining the number of basis functions is a process. One of the most common techniques to do so is much like testing the variability of least squares regression by minimizing the square of the residuals or least squares approximation.

Basis functions should have similar features to that of the functions being estimated by them to ensure optimal approximation using comparatively fewer basis functions. When K is relatively small, and the basis functions reflect characteristics of the data relatively well, the following can be expected:

- more degrees of freedom to test hypotheses and compute accurate confidence intervals;
- less computation required; and
- the coefficients themselves to be interesting descriptors of the data from a substantive perspective. Ramsey and Silverman (2002)

The two types of basis functions discussed in this paper are chosen based on the structure of the data being analysed. Fourier basis functions consist of pairs of sine and cosine functions that increase in frequency. B-spline basis functions are piecewise polynomial functions that are defined over intervals that are joined end to end at values of t called knots. These joined segments are constrained to be smooth at the knots.

4.1.1 Fourier Basis System For Periodic Data

The Fourier series is the most widely known basis expansion and can be shown as follows:

$$\hat{x}(t) = c_0 + c_1 sin\omega t + c_2 cos\omega t + c_3 sin2\omega t + c_4 cos2\omega t + ... \tag{4.7}$$

defined by the basis $\phi_0(t) = 1$, $\phi_{2r-1}(t) = sinr\omega t$, and $\phi_{2r}(t) = cosr\omega t$. The chosen continuum over which the data was recorded is time, represented by t. t_j is the time at each point in the interval.

This basis is periodic, and the parameter ω determines the period $2\pi/\omega$. If the values of t_j are equally spaced on τ and the period is equal to the length of interval τ, then the basis is orthogonal in the sense that the cross product matrix $\mathbf{\Phi'\Phi}$ is diagonal, and can be made

equal to the identity by dividing the basis functions by suitable constants, $\sqrt{n}$ for $j = 0$ and $\sqrt{n/2}$ for all other j. Ramsey and Silverman (2002)

Fourier basis functions have advantageous computational properties if the times of observation of the data are equally spaced. A Fourier series is particularly useful for functions where there are no strong local features and where the curvature tends to be of the same order everywhere. Fourier series are generally suitable for periodic data such as temperatures viewed over a calendar year, provided the times of observation are equally spaced as previously mentioned. Six Fourier basis functions are shown in Figure 4.1.

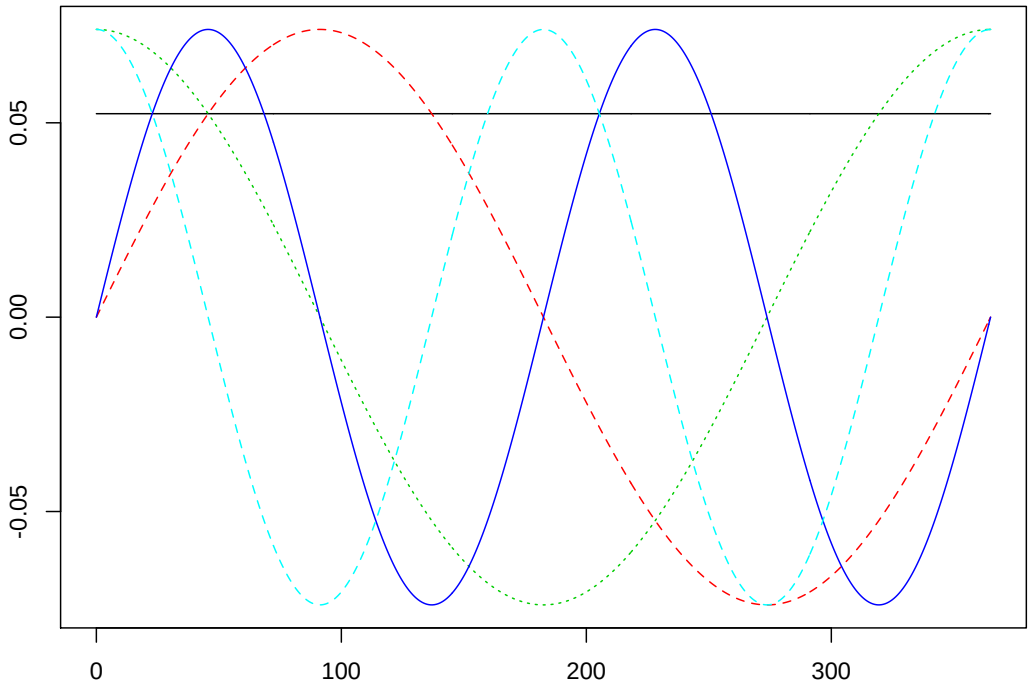


Figure 4.1: Fourier Basis Functions

4.1.2 B-Splines

Splines are polynomial segments that are joined at the ends. The points at which these segments join are called knots. A spline function is defined by the order of the polynomial segments and the location of the knots or knot sequence $\tau.\tau$ represents the interval over which the data was collected. When using B-splines one needs to choose how many knots to use, where to place the knots and how many basis functions. When defining a spline it is important to break the interval τ into subintervals separated by values $\tau_l, l = 1, \ldots, L-1$. L, is the number of subintervals. A spline is essentially a polynomial of order m over each interval. The number of constants required to define a polynomial is referred to as the *order* of the polynomial, m and it is greater than the *degree* of the polynomial by one. The number of parameters or basis functions required to define a spline function in the usual situation of one knot per breakpoint is the order plus the number of interior knots, $m + L - 1$.

The basis function system, $\phi_k(t)$, used in constructing a spline function has the following essential properties:

- Each basis function $\phi_k(t)$ is itself a spline function as defined by an order m and a knot sequence.
- Since a multiple of a spline function is still a spline function, and since sums and differences of splines are also splines, any linear combination of these basis functions is a spline function.
- Any spline function defined by m and τ can be expressed as a linear combination of these basis functions.

Consider the model based on B-spline basis functions

$$x_j = \sum_{k=1}^{K} c_k \phi_k(t_j) + \varepsilon_j \tag{4.8}$$

where $\phi(t) = (\phi_1(t), \phi_2(t), \ldots, \phi_k(t))^T$ is an $K - dimensional$ vector of B- spline basis functions and $c = (c_1, c_2, \ldots, c_k)'$ is an $K - dimensional$ vector of unknown parameters. The B-spline basis function $\phi_k(t)$ is composed of known piecewise polynomials that are smoothly connected at points t_i, called knots. In this paper B-splines of degree 3, constructed from polynomial functions are considered. Equation 4.2 is essentially Equation 4.5 with fitted splines to the predictors over a time interval.

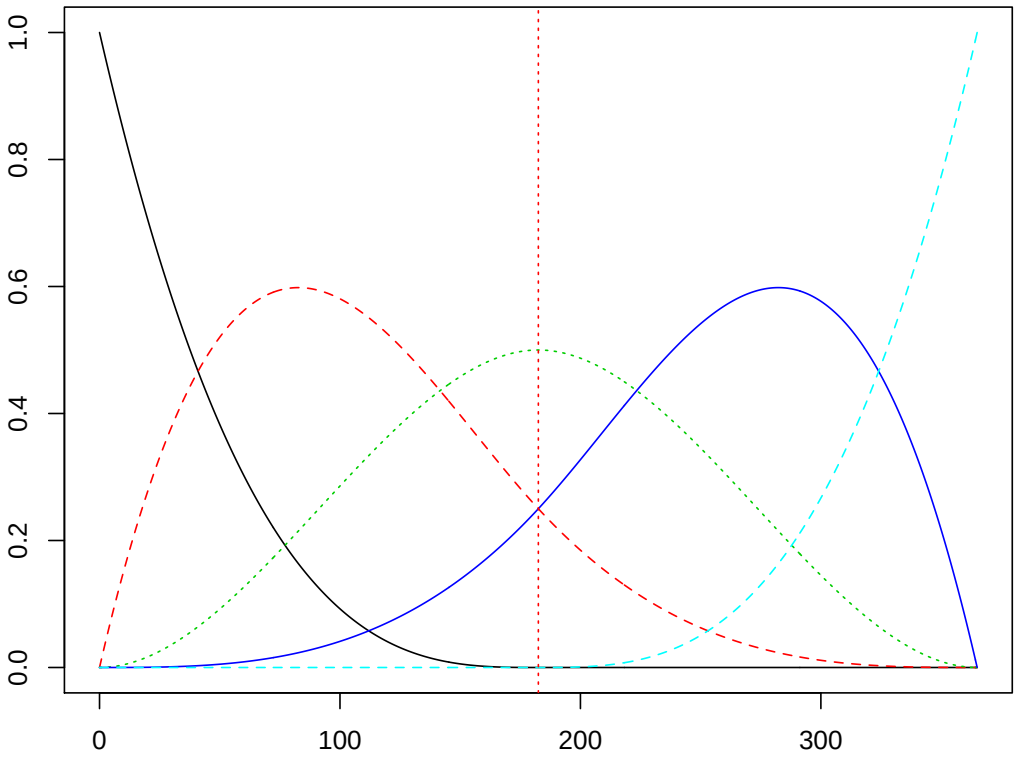


Figure 4.2: B-Spline Basis Functions

The knots required to construct m basis functions $\{b_1(x), b_2(x), ..., b_m(x)\}$ are set up as follows:

$$\tau_1 < \tau_2 < \tau_3 < \tau_4 = x_1 < \cdots < \tau_{m+1} = x_n < \ldots \tau_{m+4} \tag{4.9}$$

By setting the knots in this way, the n observations are partitioned into $m - 3$ intervals $[\tau_4, \tau_5], [\tau_5, \tau_6], \ldots, [\tau_m, \tau_{m+1}]$. Furthermore, each interval $[\tau_i, \tau_{i+1}](i = 4, \ldots, m)$ is covered by four B-spline basis functions.

There are a number of ways to choose the knots but also some important notes to consider. Knots are usually spaced equally, although this is not always necessary. With regards to each interval/segment, it is important that at least one data point exists within the interval. Furthermore, it is often quite effective to place more knots where there is identified strong

curvature and less where the function changes slowly. As mentioned above, the spline segments are constrained to be smooth at the knots which implies that the more knots you have the smoother the function, thereby suggesting that smoothness can be controlled by the number of knots chosen. In this analysis we only look at equidistant knots. Another way to determine the amount of smoothing is to choose many knots and add a roughness penalty, λ to the integrated squared second derivative.

4.2 Roughness Penalty Smoothing

Controlling smoothness by limiting the number of basis functions is discontinuous. A smoothing spline is a basis method that avoids the knot selection problem completely by using a maximal set of knots. The complexity of the fit is controlled by regularization such as roughness penalty smoothing. Roughness penalties allow continuous control over smoothness. These penalties entail fitting data using a differential equation. Basis expansions can provide good approximations to functional data provided that the basis functions have the same essential characteristics as the process generating the data. The roughness penalty or regularization approach is a more powerful option for approximating discrete data by a function. It retains the advantages of the basis function and local expansion smoothing techniques, but circumvents some of their limitations. Furthermore, it often produces better results, especially in the estimation of derivatives.

The measure of "roughness" and consequently "smoothness", is the integrated squared second derivative (curvature at time t),

$$PEN_2(x) = \int [D^2 x(t)]^2 dt \tag{4.10}$$

To allow the roughness penalty to play a role in defining $x(t)$, the penalised residual sum of squares is defined as

$$PENSSE_\lambda(x|\mathbf{y}) = [\mathbf{x} - x(\mathbf{t})]'\mathbf{W}[\mathbf{x} - x(\mathbf{t})] + \lambda PEN_2(x) \tag{4.11}$$

Where $\mathbf{y}$ is the vector of data, y_j to be smoothed, $\mathbf{t}$ is the vector of values of time, t_j and W is a symmetric positive definite weight matrix. $x(t)$ is the vector of fitted values with the basis expansion defined in Equation 4.5.

The first term is the residual sum of squares error. Minimizing only the sum of squares can result in over-smoothing if there are too many knots. Therefore a roughness penalty is added to the equation. The estimate of the function is obtained by finding the function x that minimises $PENSSE_\lambda(x)$ over the space of functions x for which the roughness penalty is defined. Whilst the objective of roughness penalty smoothing is to smooth data

such that noise is filtered out and derivatives are better estimated, it is important that the smooth function fits the data well whilst keeping bias low.

The roughness penalty may be observed in matrix notation as follows

$$\begin{aligned} PEN_2(x) &= \int [D^2\mathbf{c}'\phi(t)]^2 dt \\ &= \mathbf{c}' \int [D^2\phi(t)][D^2\phi'(t)]dt\mathbf{c} \\ &= \mathbf{c}'\mathbf{R}\mathbf{c} \end{aligned} \tag{4.12}$$

The penalized least squares criterion can now be written as

$$PENSSE(x|c) = (\mathbf{x} - \mathbf{\Phi c})'\mathbf{W}(\mathbf{x} - \mathbf{\Phi c}) + \lambda \mathbf{c}'\mathbf{R}\mathbf{c} \tag{4.13}$$

Equation 4.13 is minimized by the following

$$\hat{\mathbf{x}} = (\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}'\mathbf{W}\mathbf{x} \tag{4.14}$$

The vector containing the fitted values can now be represented as

$$\hat{\mathbf{x}} = \mathbf{\Phi}(\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}'\mathbf{W}\mathbf{x} \tag{4.15}$$

The smoothing parameter matrix can therefore be written as

$$\mathbf{S}_{\mathbf{\Phi},\lambda} = \mathbf{\Phi}(\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}'\mathbf{W} \tag{4.16}$$

In roughness penalty smoothing the degrees of freedom is given by

$$df(\lambda) = trace(\mathbf{S}_{\mathbf{\Phi},\lambda}) \tag{4.17}$$

This is used to compare the fit between roughness penalty smoothing and a fixed number of basis functions.

4.3 Choosing the Smoothing Parameter

4.3.1 Generalized Cross Validation

The basic idea behind cross-validation is to set part of the data to one side, calling it a validation sample, and fit the model to the balance of the data, called the training sample.

In that way, we see how well the model fits data that were not used to estimate the model, thus avoiding the somewhat incestuous procedure of using the data to both fit the model and assess fit.

A versatile technique for choosing a smoothing parameter involves taking this notion to the extreme situation where we leave only one observation out as the validation sample, fitting the data to the rest, and then estimating the fitted value for the left out data value. This is sometimes called leave-one-out CV. If this procedure is repeated for each observation in turn, and the resulting error sum of squares summed over all values, the result is the cross-validated error sum of squares. We compute this criterion over a range of values of , and choose that value that yields its minimum. In this dissertation we only use leave-one-out CV and will refer to this as CV throughout.

Cross-validation can be used in a wide range of situations, and in effect rests only on the assumption that observations are relatively independent of one another. However, the method has two problems. First, it is usually computationally intensive, and not the sort of thing that would be feasible for sample sizes in the thousands. The second problem is that minimizing CV can lead to under-smoothing the data because the method tends too often to favor fitting noisy or high-frequency types of variation that we would prefer to ignore.

GCV was originally developed as a simpler version of the cross-validation procedure that avoided the need to re-smooth n times. It also has been found to be rather more reliable than cross-validation in the sense of having less of a tendency to under-smooth.

GCV can be expressed as

$$GCV(\lambda) = (\frac{n}{n - df(\lambda)})(\frac{SSE}{n - df(\lambda)}) \tag{4.18}$$

where df is degrees of freedom value for a spline smooth defined as 4.17 above and $\mathbf{S}_{\phi,\lambda}$ is the smoothing operator defined as

$$\mathbf{S}_{\phi,\lambda} = \mathbf{\Phi}(\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\Phi'\mathbf{W} \tag{4.19}$$

$GCV(\lambda)$ is essentially a twice discounted mean squared error measure where the $(\frac{SSE}{n-df(\lambda)})$ is the unbiased estimate of error variance σ^2 as in familiar regression analysis, and thus represents some discounting by subtracting $df(\lambda)$ from n. The expression $(\frac{n}{n-df(\lambda)})$ further discounts this estimate. The purpose of the discounting is to reduce the tendency for GCV to under-smooth. The minimization of GCV with respect to λ will involve trying a large number of lambda values using numerical optimization algorithms.

4.3.2 Generalized Information Criteria

The Generalized information criterion (GIC) can be used to evaluate a variety of models within the framework of statistical functionals. It can be shown that the information criteria for evaluating models estimated by maximum likelihood, by maximum penalized likelihood, and by robust procedures can be derived in a unified manner.

Let $G(x)$ be the true distribution function with density $g(x)$ that generated data, and let $\hat{G(x)}$ be the empirical distribution function based on n observations, $x_n = x_1, x_2, ..., x_n$, drawn from $G(x)$. On the basis of the information contained in the observations, we choose a parametric model that consists of a family of probability distributions $f(x|\theta); \theta \in \Theta \subset R^n$, where $\theta = (\theta_1, ..., \theta_p)'$ is the p-dimensional vector of unknown parameters and Θ is an open subset of R_n. This specified family of probability distributions may or may not contain the true density $g(x)$, but it is expected that its deviation from the parametric model will not be too large. The adopted parametric model is estimated by replacing the unknown parameter vector θ by some estimate $\hat{\theta}$, for which maximum likelihood, penalized likelihood, or robust procedures may be used for estimating parameters.

In order to construct an information criterion that enables us to evaluate various types of statistical models, we employ a functional estimator that is Fisher consistent. Let us assume that the estimator $\hat{\theta_k}$ for the k^{th} parameter $\hat{\theta_k}$ is given by

$$\hat{\theta_k} = T_k(\hat{G}), \qquad k = 1, 2, \ldots, n \tag{4.20}$$

for a functional $T_k(\cdot)$. If we write the p-dimensional functional vector with $T_k(G)$ as the k^{th} element by

$$\mathbf{T}(G) = (T_1(G), T_2(G), ...T_p(G))' \tag{4.21}$$

then the p-dimensional estimator can be expressed as

$$\hat{\theta} = \mathbf{T}(\hat{G}) = (T_1(\hat{G}), T_2(\hat{G}), ...T_n(\hat{G}))' \tag{4.22}$$

Given a functional $T_k(G)$ $(k = 1, 2, ...n)$, the influence function, which is the directional derivative of the functional at the distribution G, is defined by

$$T_k^{(1)}(x; G) = \lim_{\epsilon \to 0} \frac{T_k((1-\epsilon)G + \epsilon\delta_x) - T_k(G)}{\epsilon} \tag{4.23}$$

where δ_x is a distribution function with a probably of 1 at point x. The influence function plays an essential role in the derivation of an information criterion. We define the p-dimensional vector of influence function having $T_k^{(1)}(x; G)$ as the k^{th} element by

$$\mathbf{T}^{(1)}(x; G) = (T_1^{(1)}(x; G), T_2^{(1)}(x; G), ..., T_p^{(1)}(x; G))' \tag{4.24}$$

An information criterion for evaluating the statistical model $f(x|\hat{\theta})$ with a p-dimensional functional estimator $\hat{\theta} = \mathbf{T}(\hat{G})$ is given by

$$GIC = -2\sum_{j=1}^{n} logf(x_j|\hat{\theta}) + 2/n\sum_{j=1}^{n} tr\{T^{(1)}(x_j;\hat{G})\frac{\delta logf(x_j|\theta)}{\delta\theta^T}|_{\theta=\hat{\theta}}\} \tag{4.25}$$

where $\mathbf{T}^{(1)}(x_j;G) = (T_1{}^{(1)}(x_j;\hat{G})), ..., T_n{}^{(1)}(x_j;\hat{G}))'$ and $T_k{}^{(1)}(x_j;\hat{G})$ is the empirical influence function defined by

$$T_k{}^{(1)}(x_j;\hat{G}) = \lim_{\epsilon\to 0}\frac{T_k((1-\epsilon)\hat{G} + \epsilon\delta_{x_j}) - T_k(\hat{G})}{\epsilon} \tag{4.26}$$

with δ_{x_j} being a point mass at x_j.

When selecting the best model we select the model with the smallest value of the information criterion GIC.

The GIC is used to evaluate statistical models constructed by various estimation procedures including the maximum likelihood and maximum penalised likelihood methods, and even the Bayesian approach.

4.3.3 Maximum Penalized Likelihood Methods

In the model 4.8, if it is assumed the noise term is Gaussian, the model becomes 4.27,

$$f(x_j) = \frac{1}{\sqrt{2\pi\sigma^2}}exp[-\frac{\{x_j - x(\mathbf{t}_j,\mathbf{c})\}^2}{2\sigma^2}] \tag{4.27}$$

where $\theta = (\mathbf{c}, \sigma^2)'$. The parametric model may be estimated by various procedures including maximum likelihood, robust procedures for handling outliers [Huber (1981), Hampel et al. (1986)]. Shrinkage estimators provide an alternative estimation method that may be used to advantage when the explanatory variables are highly correlated or when the number of explanatory variables is relatively large compared with the number of observations. In the estimation of nonlinear regression models for analyzing data with complex structure, the maximum likelihood method often yields unstable parameter estimates and complicated regression curves or surfaces. Instead of maximizing the log-likelihood function, we choose the values of unknown parameters to maximize the penalized log-likelihood function (or the regularized log-likelihood function)

$$l_\lambda(\theta) = \sum_{j=1}^{n} logf(x_j) - n/2\lambda H(\mathbf{c}) \tag{4.28}$$

This estimation procedure is referred to as the *maximum penalised likelihood method* or the *regularization method.*

Where $\sum_{j=1}^{n} logf(x_j)$ is a measure of goodness of fit to the data, while $n/2\lambda H(\mathbf{c})$ penalises the roughness of the regression function.The parameter $\lambda(> 0)$, called a smoothing parameter or a regularization parameter, performs the function of controlling the trade-off between the smoothness of the function and the goodness of fit to the data. A crucial aspect of model construction is the choice of the smoothing parameter λ. We consider the use of the GIC as a smoothing parameter selector. Candidate penalties or regularisation terms $H(\mathbf{c})$ with an m-dimensional parameter vector $\mathbf{c}$ can be expressed as (i) discrete approximations of the integration of a second order derivative that takes the curvature of the function into account, (ii) finite differences of the unknown parameters, and (iii) sums of squares of w_i are used, depending on the regression functions and data structure under consideration. In this paper we use finite differences of the unknown parameters expressed as,

$$H(\mathbf{c}) = \sum_{k=k+1}^{m} (\Delta^k c_k)^2 \tag{4.29}$$

where Δ represents the difference operator such that $\Delta c_i = c_i - c_{i-1}$.

The regularisation term can often be represented as the quadratic function $\mathbf{c}'K\mathbf{c}$ of the parameter vector $\mathbf{c}$, where $\mathbf{K}$ is a known $m \times m$ non-negative definite matrix. The regularisation term $H(c)$ based on the difference operator can be expressed as

$$H(\mathbf{c}) = \mathbf{c}'\mathbf{D}_k'\mathbf{D}_k\mathbf{c} = \mathbf{c}'\mathbf{K}\mathbf{c} \tag{4.30}$$

where D_k is an $(m-k) \times m$ matrix given by

$$D_k = \begin{bmatrix} {}_kC_0 & -{}_kC_1 & \dots & (-1)^k_kC_k & 0 & \dots & 0 \\ 0 & {}_kC_0 & -{}_kC_1 & \dots & (-1)^k_kC_k & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & {}_kC_0 & -{}_kC_1 & \dots & (-1)^k_kC_k \end{bmatrix} \tag{4.31}$$

with the binomial coefficient ${}_kC_k$. The regularisation term used in this paper is a second-order difference term given by

$$D_2 = \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & -2 & 1 \end{bmatrix} \tag{4.32}$$

We now consider the penalised log-likelihood function expressed as

$$l_\lambda(\theta) = \sum_{\alpha=1}^{n} logf(x_j) - n/2\lambda \mathbf{c}' K \mathbf{c} \tag{4.33}$$

Let $\hat{\theta}_n$ be the estimator that maximises (4.33). It can be seen that the estimator $\hat{\theta}_p$ is given as the solution of the implicit equation

$$\sum_{j=1}^{n} \psi_n(x_j, \theta) = 0 \tag{4.34}$$

where

$$\psi_n(y_j, \theta) = \frac{\delta}{\delta\theta}\{logf(y_j|x_j; \mathbf{j}) - \frac{\lambda}{2}\mathbf{w}' K \mathbf{w}\} \tag{4.35}$$

Therefore, an information criterion for evaluating the model $f(y|x; \hat{\theta}_n)$ estimated by regularisation can easily be obtained within the framework of robust estimation.

4.3.4 Information criterion for a model estimated by regularisation

Konishi.S, and Kitagawa.G(2008) show that an information criterion for the model $f(y|x; \hat{\theta}_n)$ with $\hat{\theta}_n$ obtained by maximising (4.33) is given by

$$GIC_n = -2\sum_{j=1}^{n} logf(x_j) + 2tr\{R(\psi_n, \hat{G})^{-1} Q(\psi_n, \hat{G})\} \tag{4.36}$$

where $R(\psi_n, \hat{G})$ and $Q(\psi_n, \hat{G})$ are $(K+1)$ x $(K+1)$ matrices.

By setting $l_j(\theta) = logf(x_j)$ with $\theta = (\mathbf{c}', \sigma^2)'$, these matrices can be expressed as

$$\frac{\delta\psi(x_j, \theta)^T}{\delta\theta} = \begin{bmatrix} \frac{\delta^2 l_j(\theta)}{\delta\mathbf{c}\delta\mathbf{c}'} - \lambda K & \frac{\delta^2 l_j(\theta)}{\delta\mathbf{c}\delta\sigma^2} \\ \\ \frac{\delta^2 l_j(\theta)}{\delta\sigma^2\delta\mathbf{c}^T} & \frac{\delta^2 l_\alpha(\theta)}{\delta\sigma^2\delta\sigma^2} \end{bmatrix}, \tag{4.37}$$

$$\psi_p(x_j, \theta_n)\frac{\delta logf(x_j)}{\delta\theta'} = \begin{bmatrix} \frac{\delta l_j(\theta)}{\delta\mathbf{c}}\frac{\delta l_j(\theta)}{\delta\mathbf{c}'} - \lambda K\mathbf{c}\frac{\delta l_j(\theta)}{\delta\mathbf{c}'} & \frac{\delta l_j(\theta)}{\delta\mathbf{c}}\frac{\delta l_j(\theta)}{\delta\sigma^2} - \lambda K\mathbf{c}\frac{\delta l_j(\theta)}{\delta\sigma^2} \\ \\ \frac{\delta l_j(\theta)}{\delta\sigma^2}\frac{\delta l_j(\theta)}{\delta\mathbf{c}'} & \{\frac{\delta l_j(\theta)}{\delta\sigma^2}\}^2 \end{bmatrix} \tag{4.38}$$

Since the estimated model $f(x_j)$ depends on a smoothing parameter λ the choice of λ is a crucial part of nonlinear modeling. Selection of the smoothing parameter in the modeling

process can be viewed as a model selection and evaluation problem. Therefore, an information criterion for evaluating the model $f(x_j)$ estimated by regularization may be used as a smoothing parameter selector. By evaluating statistical models determined according to the various values of the smoothing parameter, we take the optimal value of the smoothing parameter to be that which minimizes the value of GIC_n.

4.3.5 Nonlinear Regression Modeling via Basis Expansions

In this section, we consider the problem of evaluating nonlinear regression models constructed by the method of regularization. The information criterion GIC is applied to the choice of smoothing parameters and the number of basis functions in the model building process. Suppose we have n independent observations $(x_j, x(t_j)); j = 1, 2, ..., n$, where x_j are random response variables and $x(t_j)$ are $p - dimensional$ vectors of the explanatory variables. In order to extract information from the data, we use the Gaussian nonlinear regression model

$$x_j = (\mathbf{x}(t_j)) + \varepsilon_j, \quad j = 1, 2, \ldots, n \tag{4.39}$$

where $x(\cdot)$ is an unknown smooth function and the errors ε_j are independently, normally distributed with mean zero and variance σ^2. The problem to be considered is estimating the function $x(\cdot)$ from the observed data,x_j, for which we use a regression function expressed as a linear combination of a prescribed set of m basis functions in the following:

$$x(\mathbf{t}_j) = \sum_{i=1}^{K} c_i \phi_i(\mathbf{t}_j) \tag{4.40}$$

where $\phi_k(\mathbf{t})$ are real-valued functions of a $p - dimensional$ vector of explanatory variables $\mathbf{t} = (t_1, t_2, \ldots, t_n)'$.

The regression model based on the basis expansion is represented by

$$x_j = \sum_{k=1}^{K} c_k \phi_k(\mathbf{t}_j) + \varepsilon_j = \mathbf{c}'\phi(\mathbf{t}_j) + \varepsilon_j, \quad j = 1, 2, \ldots, n \tag{4.41}$$

Then a regression model with Gaussian noise is expressed as a probability density function:

$$f(x_j) = \frac{1}{\sqrt{2\pi\sigma^2}} exp[-\frac{\{x_j - \mathbf{c}'\phi(\mathbf{t}_j)\}^2}{2\sigma^2}] \tag{4.42}$$

where $\theta = (\mathbf{c}', \sigma^2)'$. The unknown parameter vector θ is estimated by maximising the

penalised log-likelihood function:

$$\begin{aligned} l_\lambda(\theta) &= \sum_{j=1}^{n} logf(x_j) - \frac{n}{\lambda 2}\mathbf{c}'\mathbf{K}\mathbf{c} \\ &= -\frac{n}{2}log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{n}\{y_j - \mathbf{c}'\phi(\mathbf{t}_j)\}^2 - \frac{n}{\lambda 2}\mathbf{c}'\mathbf{K}\mathbf{c} \\ &= -\frac{n}{2}log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{x_j} - \Phi\mathbf{c})'(\mathbf{x_j} - \Phi\mathbf{c}) - \frac{n}{\lambda 2}\mathbf{c}'\mathbf{K}\mathbf{c} \end{aligned} \tag{4.43}$$

By differentiating $l_\lambda(\theta)$ with respect to $\theta = (\beta', \sigma^2)'$ and setting the result equal to 0, Konishi.S, and Kitagawa (2008) show that we have the maximum penalised likelihood estimators for $\mathbf{c}$ and σ^2 respectively given by

$$\hat{\mathbf{c}} = (\Phi'\Phi + n\lambda\hat{\sigma}^2 K)^{-1}\Phi'\mathbf{x} \quad and \quad \hat{\sigma}^2 = \frac{1}{n}(\mathbf{x} - \mathbf{\Phi}\hat{\mathbf{c}})'(\mathbf{x} - \mathbf{\Phi}\hat{\mathbf{c}}) \tag{4.44}$$

When $\mathbf{R}$ in Equation 4.14 is equal to $n\sigma^2 K$ and $\mathbf{W} = \mathbf{I}$, Equation 4.44 is equal to 4.14.

Since the estimator $\hat{\mathbf{c}}$ in (4.44) depends on the variance estimator $\hat{\sigma}^2$, in practice it is calculated using the following method. First, put $\beta = \lambda\hat{\sigma}^2$ and determine $\hat{\mathbf{c}} = \mathbf{\Phi'\Phi} + n\beta_0\mathbf{K})^{-1}\mathbf{\Phi'}\mathbf{x}$ for a given $\beta = \beta_0$. Then, after determining the variance estimator σ^2,obtain the value of the smoothing parameter as $\lambda = \beta/\hat{\sigma}^2$.

The statistical model is obtained by replacing the unknown parameters $\mathbf{c}$ and σ^2 in (4.42) with their estimators $\hat{\mathbf{c}}$ and $\hat{\sigma}^2$ and is of the form

$$f(x_j) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}}exp[-\frac{\{x_j - \hat{\mathbf{c}}'\phi(\mathbf{t}_j)\}^2}{2\hat{\sigma}^2}] \tag{4.45}$$

The estimators $\hat{\mathbf{c}}$ and $\hat{\sigma}^2$ depend on the smoothing parameter λ (or β) and also the number m of basis functions. The optimal values of these adjusted parameters have to be chosen by a suitable criterion, for which we use an information criterion for evaluating the statistical model $f(x_j)$.

4.3.6 Information Criterion for a statistical model constructed by regularised expansions

Suppose that $f(x_j)$ in (4.45) is the Gaussian nonlinear regression model based on basis functions. Then an information criterion for the model $f(x_j)$ estimated by regularization is given by

$$GIC_PB = n(log2\pi + 1) + nlog(\hat{\sigma}^2) + 2tr\{R(\psi_p, \hat{G})^{-1}Q(\psi_j, \hat{G})\} \tag{4.46}$$

where $\hat{\sigma}^2$ is given (4.44), and the $(m+1) \times (m+1)$ matrices $R(\psi_n, \hat{G})$ and $Q(\psi_n, \hat{G})$ are, respectively

$$R(\psi_p, \hat{G}) = \frac{1}{n\hat{\sigma}^2} \quad \begin{bmatrix} \mathbf{\Phi}'\mathbf{\Phi} + n\lambda\hat{\sigma}^2\mathbf{K} & \frac{1}{\hat{\sigma}^2}\mathbf{\Phi}'\Lambda\mathbf{1}_n \\ \frac{1}{\hat{\sigma}^2}\mathbf{1}_n'\Lambda\mathbf{\Phi} & \frac{n}{2\hat{\sigma}^2} \end{bmatrix},$$

$$Q(\psi_p, \hat{G}) = \frac{1}{n\hat{\sigma}^2} \quad \begin{bmatrix} \frac{1}{\hat{\sigma}^2}\mathbf{\Phi}'\Lambda^2\mathbf{\Phi} - \lambda\mathbf{K}\mathbf{c}\mathbf{1}_n^T\Lambda\mathbf{\Phi} & \frac{1}{2\hat{\sigma}^4}\mathbf{\Phi}'\Lambda^3\mathbf{1}_n - \frac{1}{2\hat{\sigma}^2}\mathbf{\Phi}^T\Lambda\mathbf{1}_n \\ \frac{1}{2\hat{\sigma}^4}\mathbf{1}_n^T\Lambda^3\mathbf{\Phi} - \frac{1}{2\hat{\sigma}^2}\mathbf{1}_n^T\Lambda\mathbf{\Phi} & \frac{1}{4\hat{\sigma}^6}\mathbf{1}_n^T\Lambda^4\mathbf{1}_n - \frac{n}{4\hat{\sigma}^2} \end{bmatrix} \tag{4.47}$$

where $\mathbf{1}_n = (1, 1, \ldots, 1)'$ is an $n-dimensional$ vector, the elements of which are all 1, and Λ is an n x n diagonal matrix defined by

$$\Lambda = diag[x_1 - \hat{c}'\phi(\mathbf{t}_1), x_2 - \hat{c}'\phi(\mathbf{t}_2), \ldots, x_n - \hat{c}'\phi(\mathbf{t}_n)] \tag{4.48}$$

With respect to the number m of basis functions and the values of the smoothing parameter λ (or β), we select the values of $(\hat{m}, \hat{\lambda})$ that minimize the information criterion GIC_PB as the optimal values. In applying this technique to practical problems, the smoothness can also be controlled using λ, by fixing the number of basis functions.

4.4 Fitting Functional Regression Models

When estimating the functional regression models we fit Equation 4.3. To estimate the regression coefficient predicting the responses, we define β as in Equation 4.4.

We then fit 4.3 by minimizing the penalized sum of squares which is defined as follows:

$$PENSSE_\lambda(\alpha_0, \beta) = \sum [y_i - \alpha_0 + \int x_i(t)\beta(t)dt]^2 + \lambda \int [D^2\beta(t)]^2 dt \tag{4.49}$$

Where $\int [D^2\beta(t)]^2 dt$ is the penalty. To obtain a least squares estimate we define a matrix $\mathbf{Z}$ containing basis function expansions used to represent $\beta_k(t)$ and a penalty matrix. We define $\mathbf{Z}$ as:

$$\mathbf{Z} = \begin{bmatrix} 1 & \int x_1 1(t)\mathbf{\Phi}_1(t)dt & \ldots & \int x_1 q(t)\mathbf{\Phi}_q(t) \\ \vdots & \ldots & \ldots & \ddots \\ 1 & \int x_n 1(t)\mathbf{\Phi}_1(t)dt & \ldots & \int x_n q(t)\mathbf{\Phi}_q(t) \end{bmatrix} \tag{4.50}$$

The basis expansions used to represent $\beta_k(t)$ is $\mathbf{\Phi}_k$. The penalty matrix is defined by:

$$\mathbf{R}(\lambda) = \begin{bmatrix} 0 & \dots & \dots & \dots \\ 0 & \lambda_1 R_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ & 0 & \dots & \lambda_q R_q \end{bmatrix} \tag{4.51}$$

Where R_k is the penalty matrix associated with the smoothing penalty for β_k and λ_k is the corresponding smoothing parameter. Ramsay, J.et al (2009).

We then create a vector that stores the estimated coefficients, $\hat{\alpha}$, and the coefficients defining each estimated coefficient function $\hat{\beta}_k(t)$ which is estimated by least squares. This vector is defined as:

$$\hat{\mathbf{b}} = (\mathbf{Z}'\mathbf{Z} + \mathbf{R}(\lambda))^{-1}\mathbf{Z}'\mathbf{y} \tag{4.52}$$

The objects in Equation 4.52 are then used to create the appropriate functional data objects. Now when we fit the model we define β using these functional data objects which incorporate roughness penalty smoothing. We let the data define the smoothing level by using GCV to choose the λ value used in the model. To do this we let $\alpha_\lambda^{(-i)}$ and $\beta_\lambda^{(-i)}$ be the estimated regression parameters estimated without the ith observation. The resulting CV score is defined as:

$$CV(\lambda) = \sum_{i=1}^{N} [y_i - \alpha_\lambda^{(-i)} - \int x_i(t)\beta_\lambda^{(-i)} dt]^2 \tag{4.53}$$

Observing that

$$\hat{\mathbf{y}} = \mathbf{Z}(\mathbf{Z}'\mathbf{Z} + \mathbf{R}(\lambda))^{-1}\mathbf{Z}'\mathbf{y} = \mathbf{H}\mathbf{y} \tag{4.54}$$

Which can then be generalized and written as

$$GCV(\lambda) = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{(n - Tr(H))^2} \tag{4.55}$$

4.4.1 Confidence Intervals

After fitting the functional linear regression we used confidence intervals to measure the precision of our estimates of each $\hat{\beta}(t)$. We assumed the errors, ε_i are independent and

normally distributed with a mean of zero and variance of σ_e^2. The covariance of ε is defined as

$$\Sigma = \sigma_e^2 I \tag{4.56}$$

And hence the sampling variance of the estimated parameter vector $\hat{\mathbf{b}}$ is given by

$$Var[\hat{\mathbf{b}}] = (\mathbf{Z}'\mathbf{Z} + \mathbf{R}(\lambda))^{-1}\mathbf{Z}'\Sigma\mathbf{Z}(\mathbf{Z}'\mathbf{Z} + \mathbf{R})^{-1} \tag{4.57}$$

We use Equation 4.57 to obtain the confidence intervals seen in our functional regression models.

We use the F-ratio and squared multiple correlation to assess the fit of our models and subsequently the predictive power of these models.

The F-ratio is defines as

$$F = \frac{Var[\hat{\mathbf{y}}]}{\frac{1}{n}\sum(y_i - \hat{y}_i)^2} \tag{4.58}$$

Multiple correlation is defined as

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2 \sum(y - \bar{y})^2}} \tag{4.59}$$

After exploring differing methods we opted for using basis functions together with roughness penalty smoothing to ensure the function fits the data well and that bias is kept low.

In Chapter 5 we discuss the practical implementation of the methodology and the methods we used to clean and explore the data. The outcome of this exploration is what informed the choice of methodology used in the analysis.

Chapter 5

Analysis

5.1 Data Cleaning

Researchers believe that trees assume different shapes depending on their location, and in particular,believe that trees in the dry north and central regions are taller with fewer branches while those in the cool, wet south are shorter and rounder to produce more seeds.

With 754 weather stations in our weather data set, it was important to find the weather stations which were closest to our tree locations to use as proxies for the weather at the exact sites. Figure 5.1 shows the full set of weather stations and tree locations in the complete dataset. Table 2.2 shows a full set of the weather stations with their corresponding longitude and latitude coordinates as well as the elevation.The most useful weather stations are the ones closest to the tree locations and at similar altitudes. Six weather stations were chosen based on their distance from the tree locations as shown in Figure 5.2 and Table 5.1. However, a number of problems present themselves. First, the elevation plays a big part in the type of weather condition an area experiences which means that with some of the tree locations, the nearest weather station was at a significantly different elevation making for a poor proxy. Second, for relative accuracy it was important that we use weather station data recordings that had a complete set of information for the same period the trees were observed.As a result the total number of weather stations used in the analysis was cut down to 3; J.G.H Wath Airport, Gobabeb and Springbok, described in detail in Table 5.1.

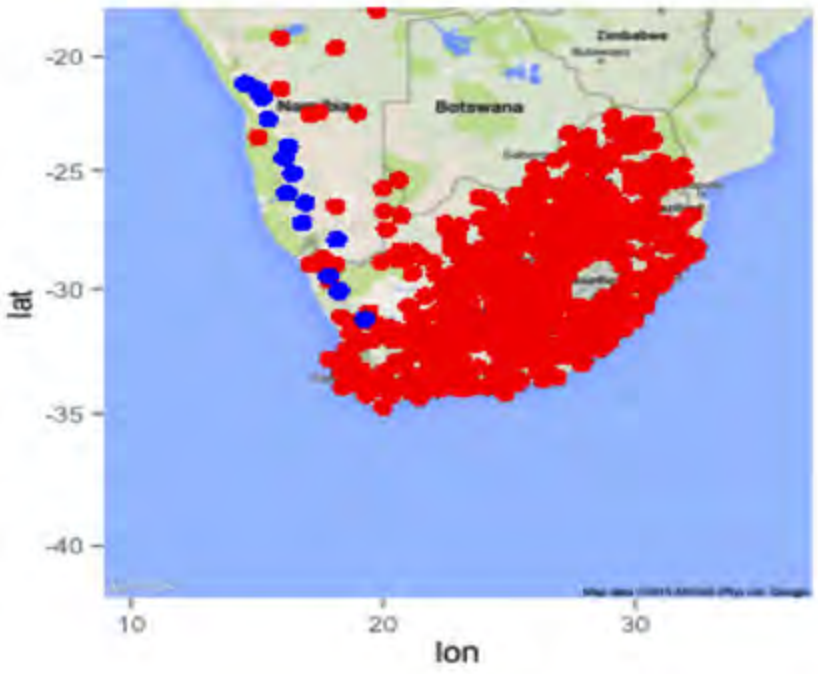


Figure 5.1: Full Set of Weather Stations and Tree Locations

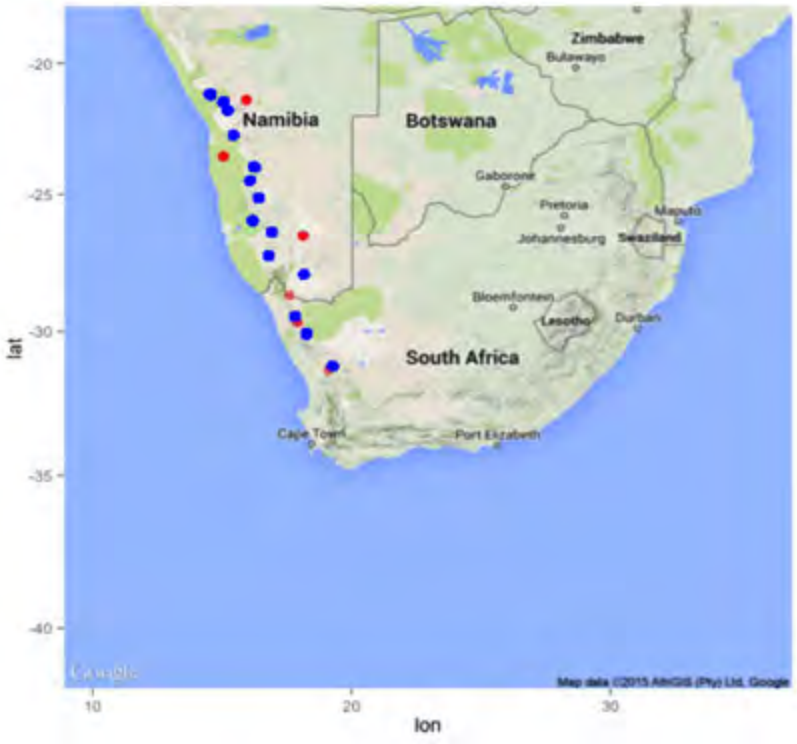


Figure 5.2: Selected Weather Stations Used in Analysis and Tree Locations

Figure 5.3: Selected Weather Stations Used in Analysis and Tree Locations

Table 5.1: Table Showing Details of Selected Weather Stations

Weather Station	Latitude	Longitude	Elevation
JGH Van der Wath Airport	-26.53	18.12	1 217
Springbok	-29.67	17.9	1 007
Gobabeb	-23.57	15.05	400

5.1.1 From discrete to functional form

Date vectors were created for consistency as tree data and weather station data were from different sources. From the 3 selected weather stations, the maximum and minimum temperature as well as the rainfall was extracted as the predictors of interest. As one would expect, using one weather station as a proxy for the weather in multiple locations presents additional challenges. One such challenge is collinearity. Collinearity occurs when two or more predictor variables in a multiple regression model are highly correlated. When variables are highly correlated it means that one variable can be linearly predicted using the other variables with a significant degree of accuracy (Haitovsky, 1969). To address these challenges, we sampled from the weather station of interest for each of the locations so as to have slightly differing functions for each location, thus avoiding the problem of

collinearity. The weather data was observed daily but for the rainfall, we converted it to monthly because rainfall data is relatively irregular as it does not rain everyday. The rainfall data was also tested for skewness and was found to be skew to the left, as expected, with a coefficient of skewness of -1.026692. This can be seen in Figure 5.4 below. The effect of this skewness can particularly be seen in much larger lambda values relative to those produced from symmetric data. We use the log of rainfall because the rainfall data had many missing values as well as values of zero.

To determine the best smoothing parameter to use in our analysis we used both GCV and GIC methods and ultimately decided on the GCV smoothing parameter to create our functions. We used the general-purpose optimization function 'Optim' in R and the method 'Brent' as our problem is one-dimensional. When fitting the functions we constructed functional data objects with the function 'smooth.basis' in R, which smooths the data using a roughness penalty. Smooth curves defined by the B-spline basis function system are used to fit the discrete data on curves for each of the climate variables used as predictor variables. The number of basis functions used for temperature is 500, which is roughly 5 basis functions for each month, and 228 for rainfall which is one basis function for each month. The fitting criterion used by 'smooth.basis' is weighted least squares. The function 'smooth.basis' from the R package 'fda'(Ramsay.J,2014), requires a functional parameter to be specified. In this analysis we use functional data objects that specify the roughness penalty as defined by the linear differential operator object and the smoothing parameter. It can be said that using a good roughness penalty can yield more satisfactory results than simply varying the number of basis functions in the expansion. The function 'eval.fd',also from the R package 'fda' (Ramsay.J,2014), is then used to evaluate the resulting smooth data contained in the functional data object created by 'smooth.basis' over the dates of interest.

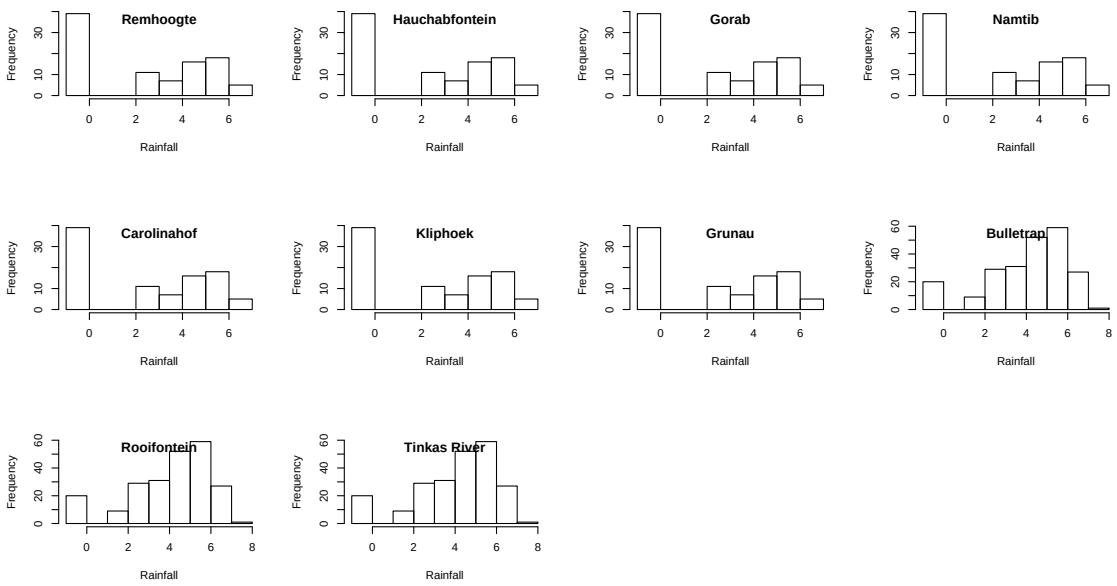


Figure 5.4: Log Rainfall Histograms

5.1.2 Explore data

The functions were fitted against the data to test the fit for maximum and minimum temperature as well as rainfall. The results can be seen in Figures 5.5, 5.6, and 5.7.

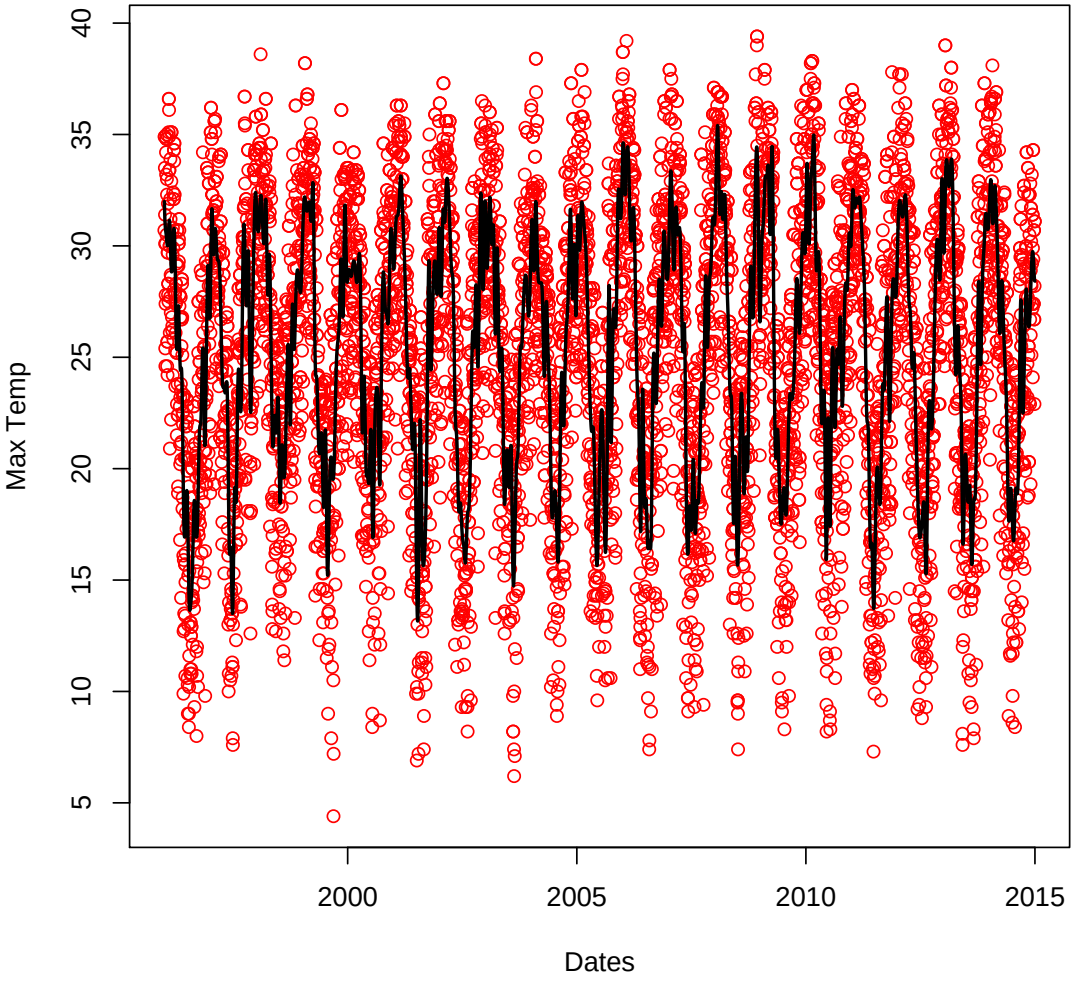


Figure 5.5: Fitted Function: Maximum Temperature

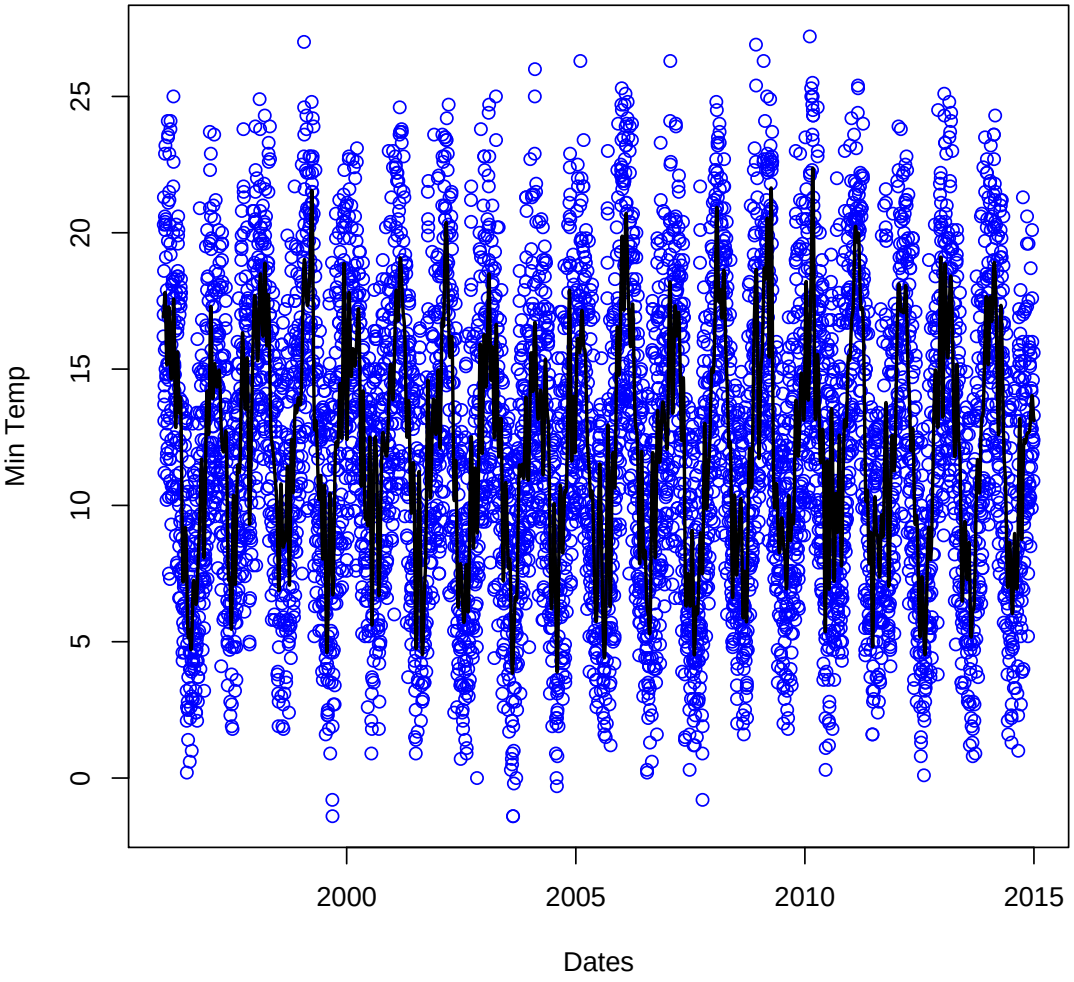


Figure 5.6: Fitted Function: Minimum Temperature

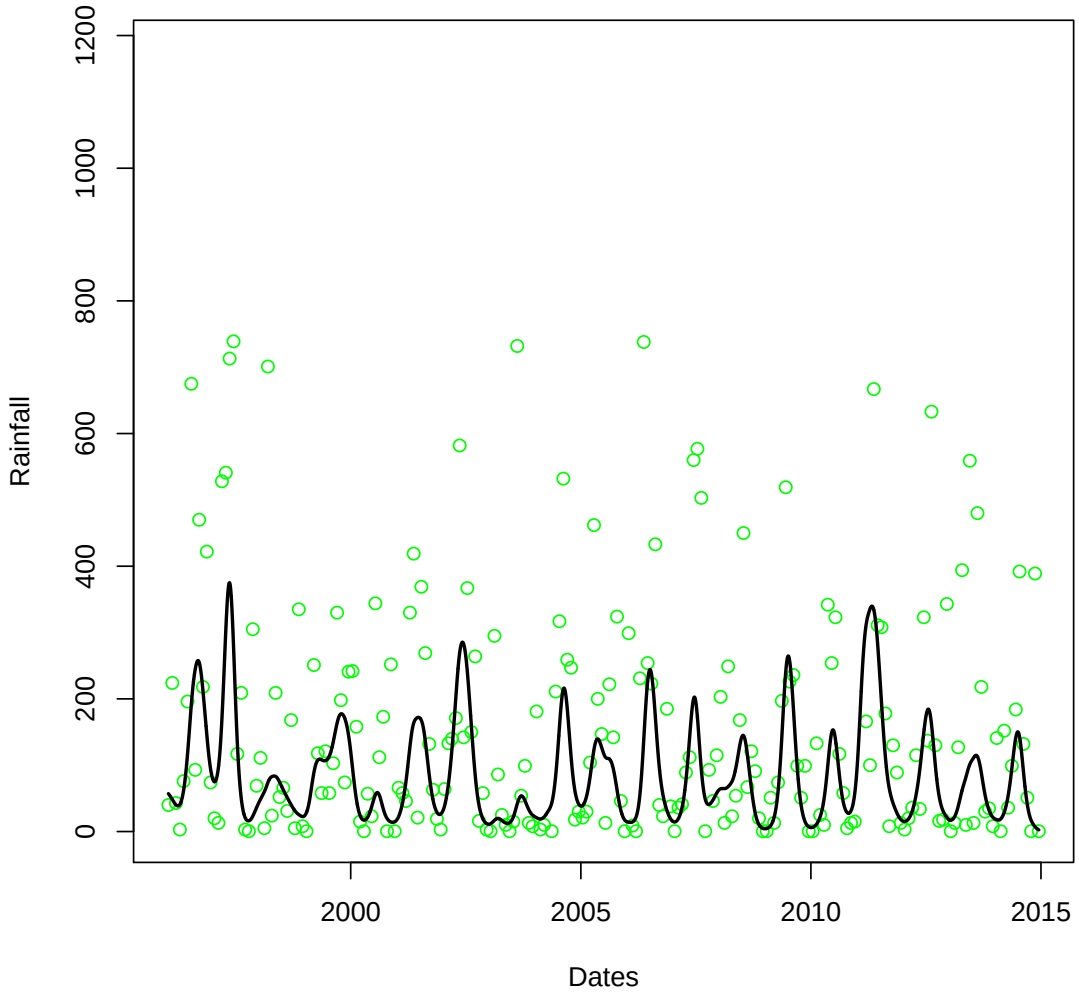


Figure 5.7: Fitted Function: Rainfall

We also compared the fit of functions from different locations that sampled from the same weather station. Figures 5.8, 5.9, 5.10 show the maximum temperature fits from the three weather stations. Figures 5.11, 5.12, 5.13 show the minimum temperature fits from the three locations. Figures 5.14, 5.15, 5.16 show the rainfall fits sampled from the three locations.

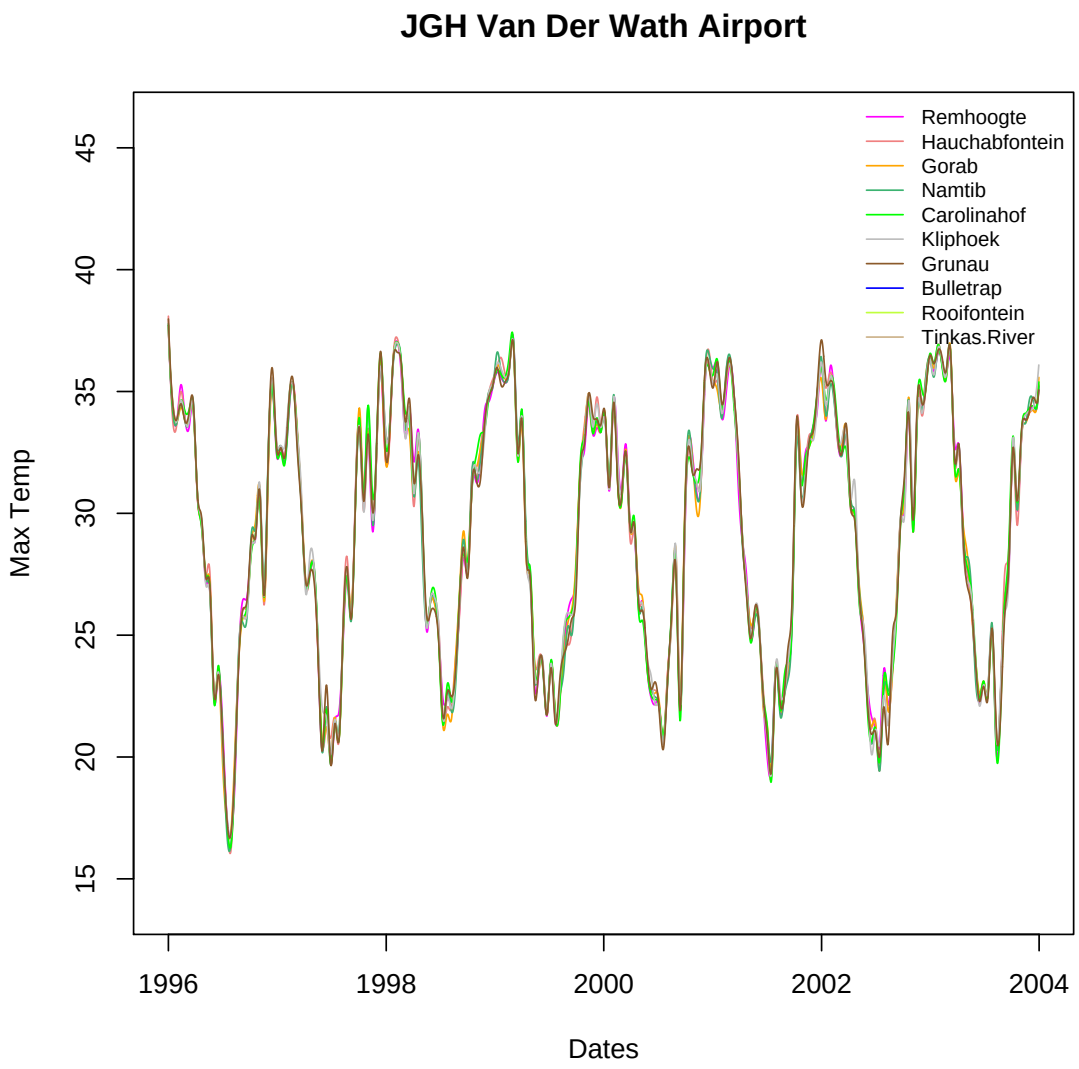


Figure 5.8: Maximum Temperature Station 1

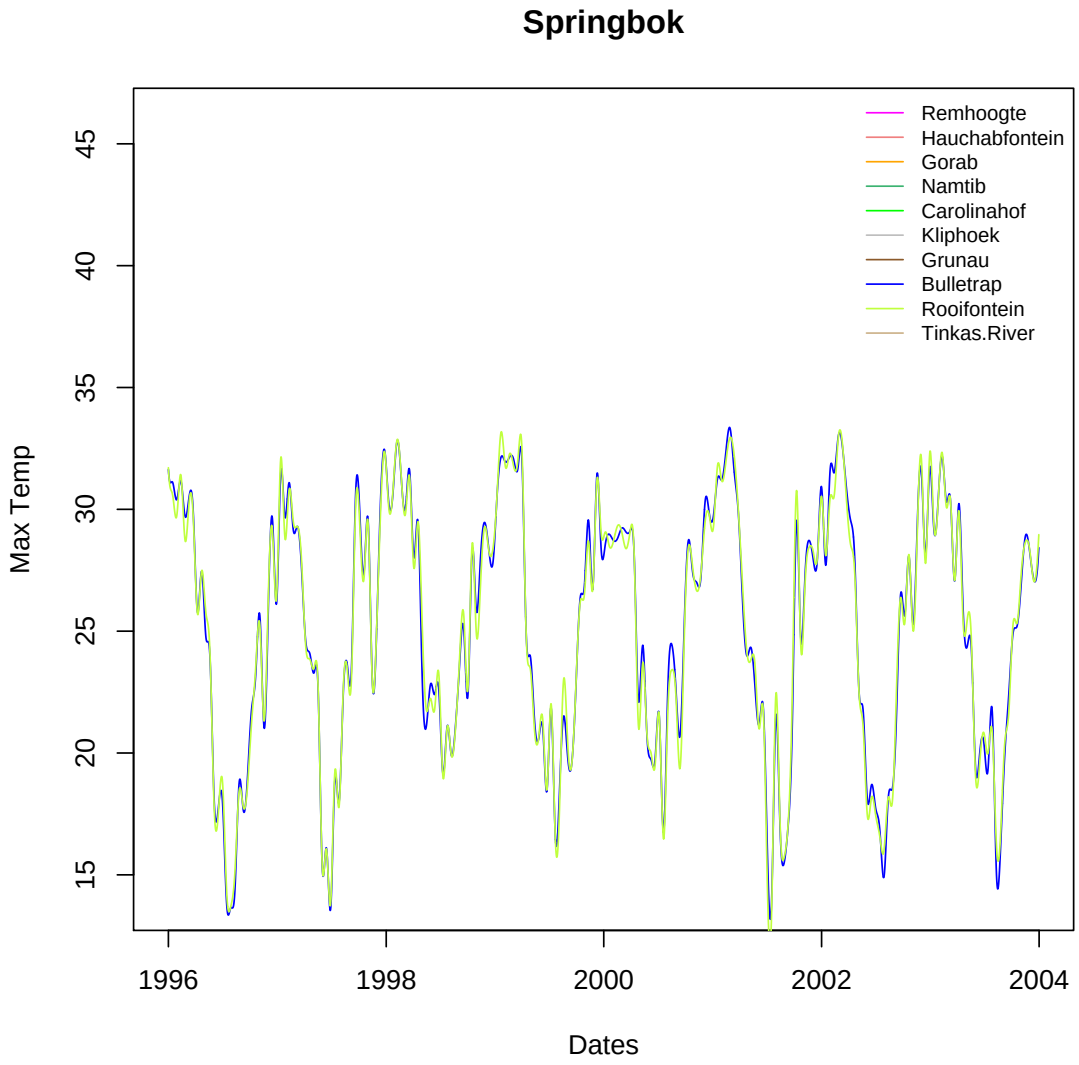


Figure 5.9: Maximum Temperature Station 2

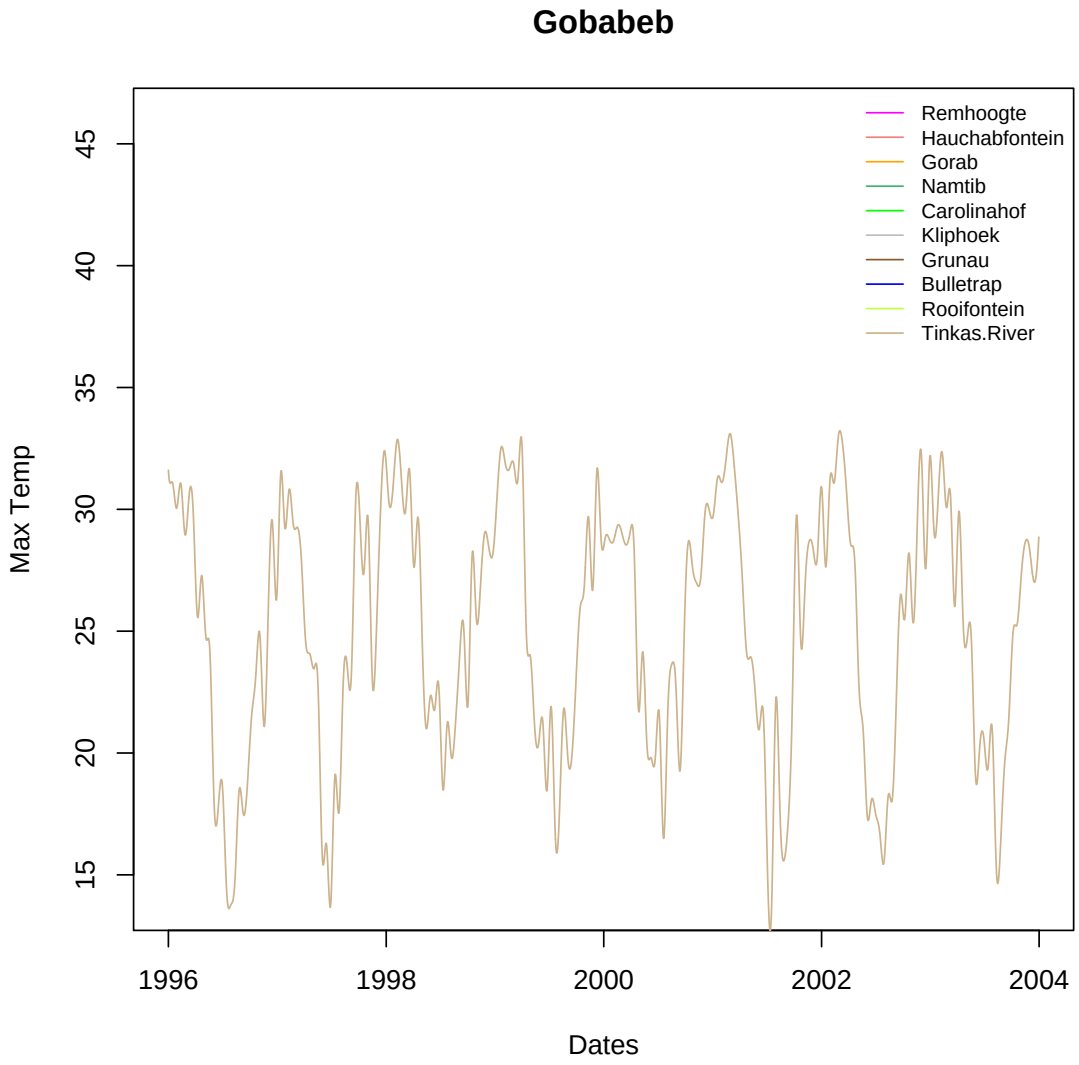


Figure 5.10: Maximum Temperature: Station 3

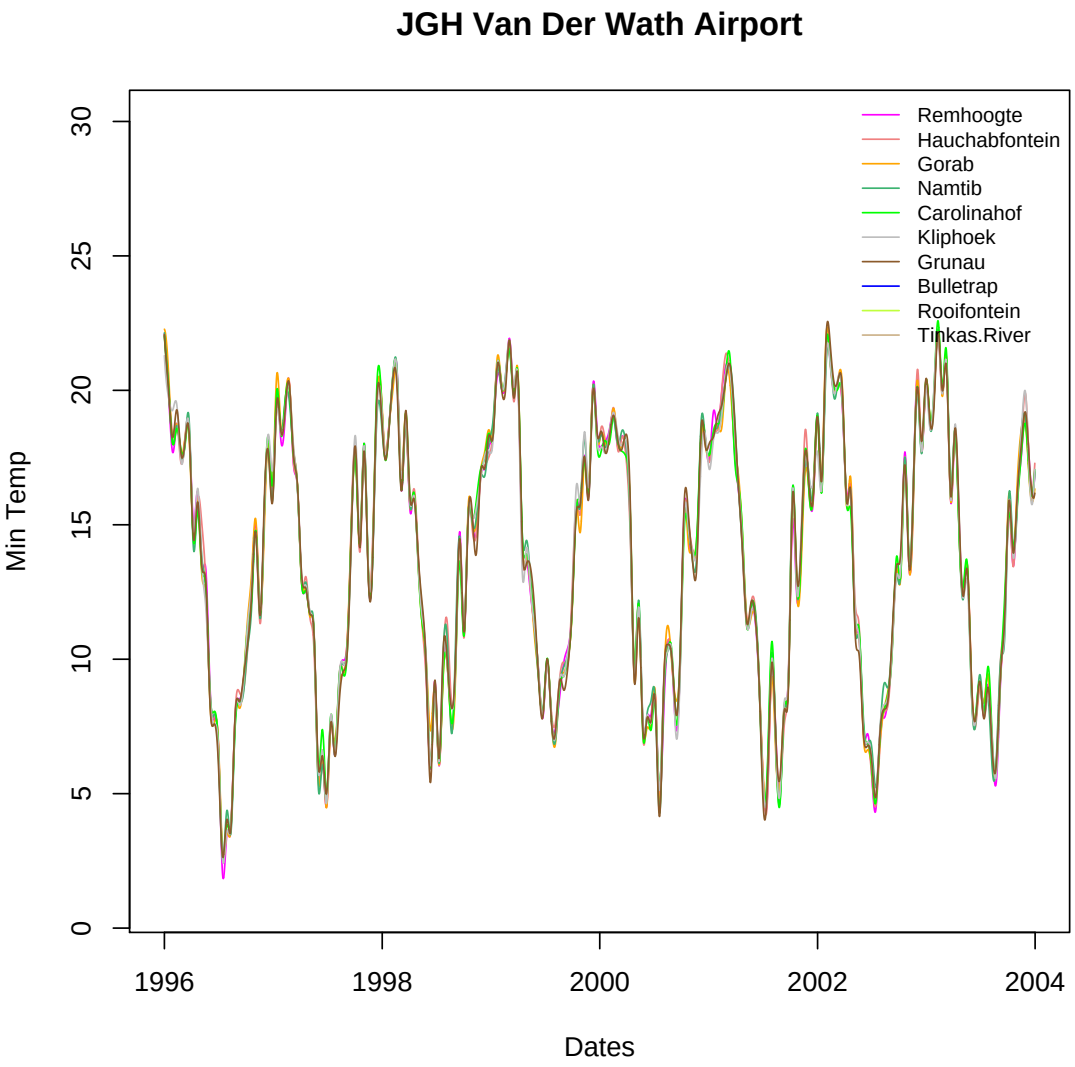


Figure 5.11: Minimum Temperature: Station 1

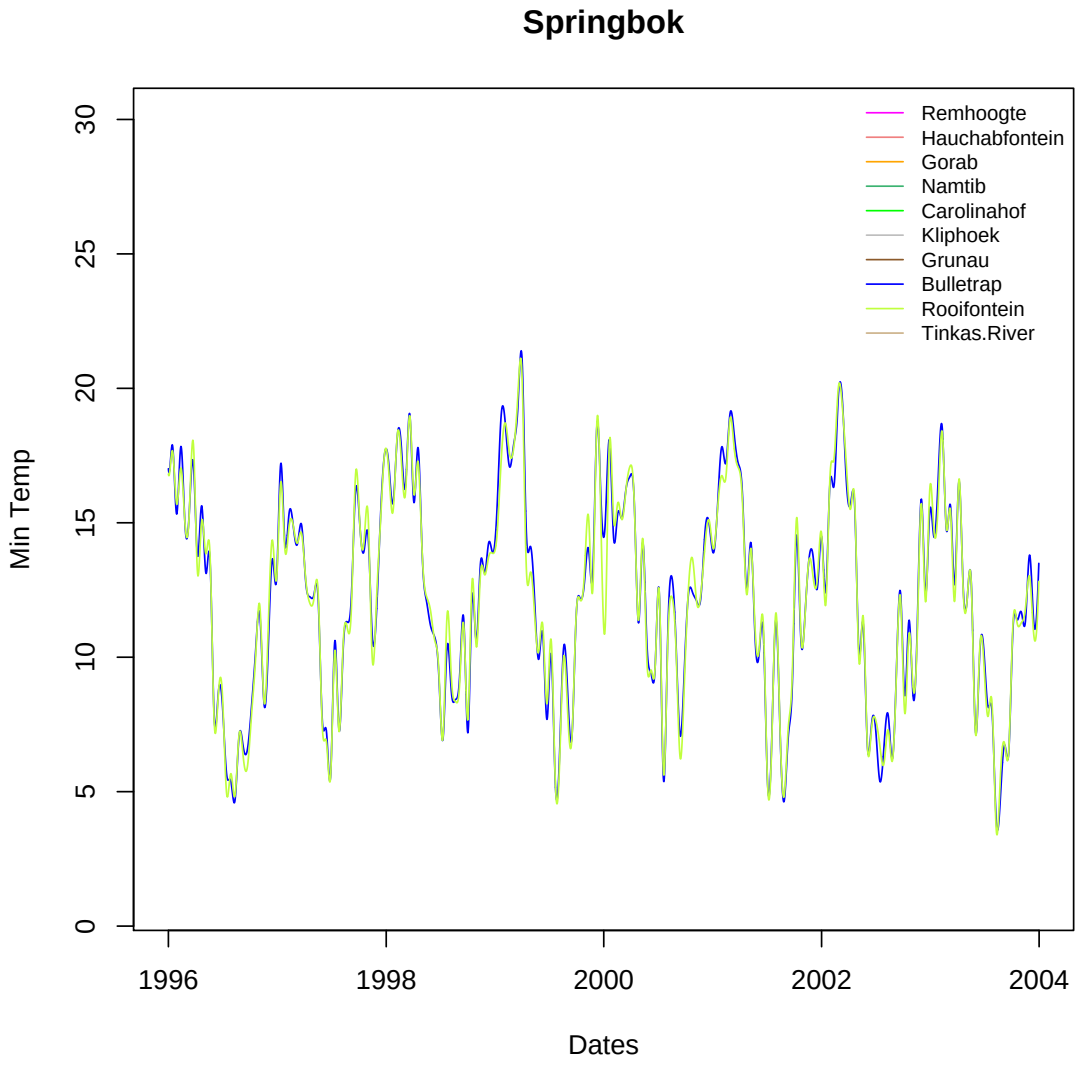


Figure 5.12: Minimum Temperature: Station 2

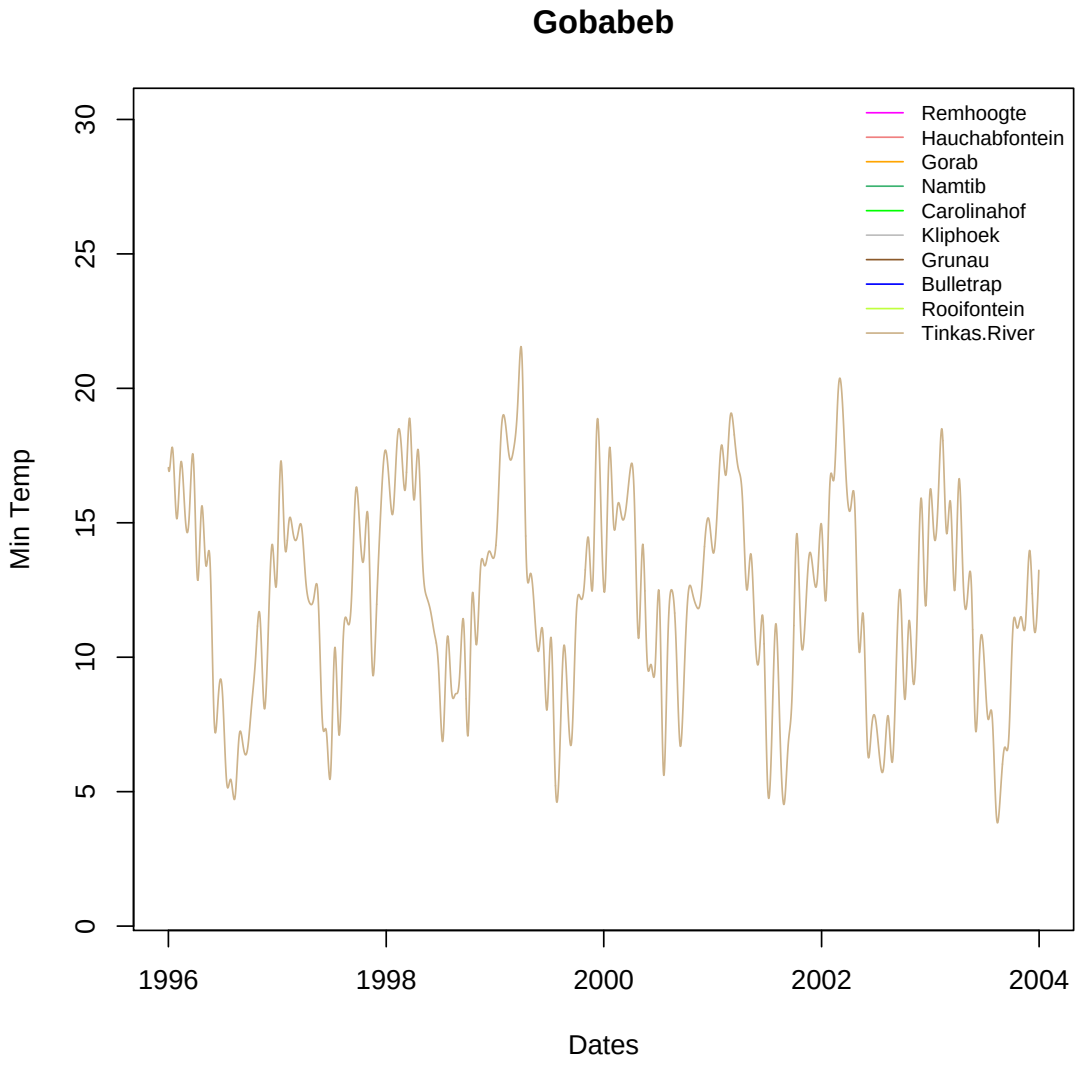


Figure 5.13: Minimum Temperature: Station 3

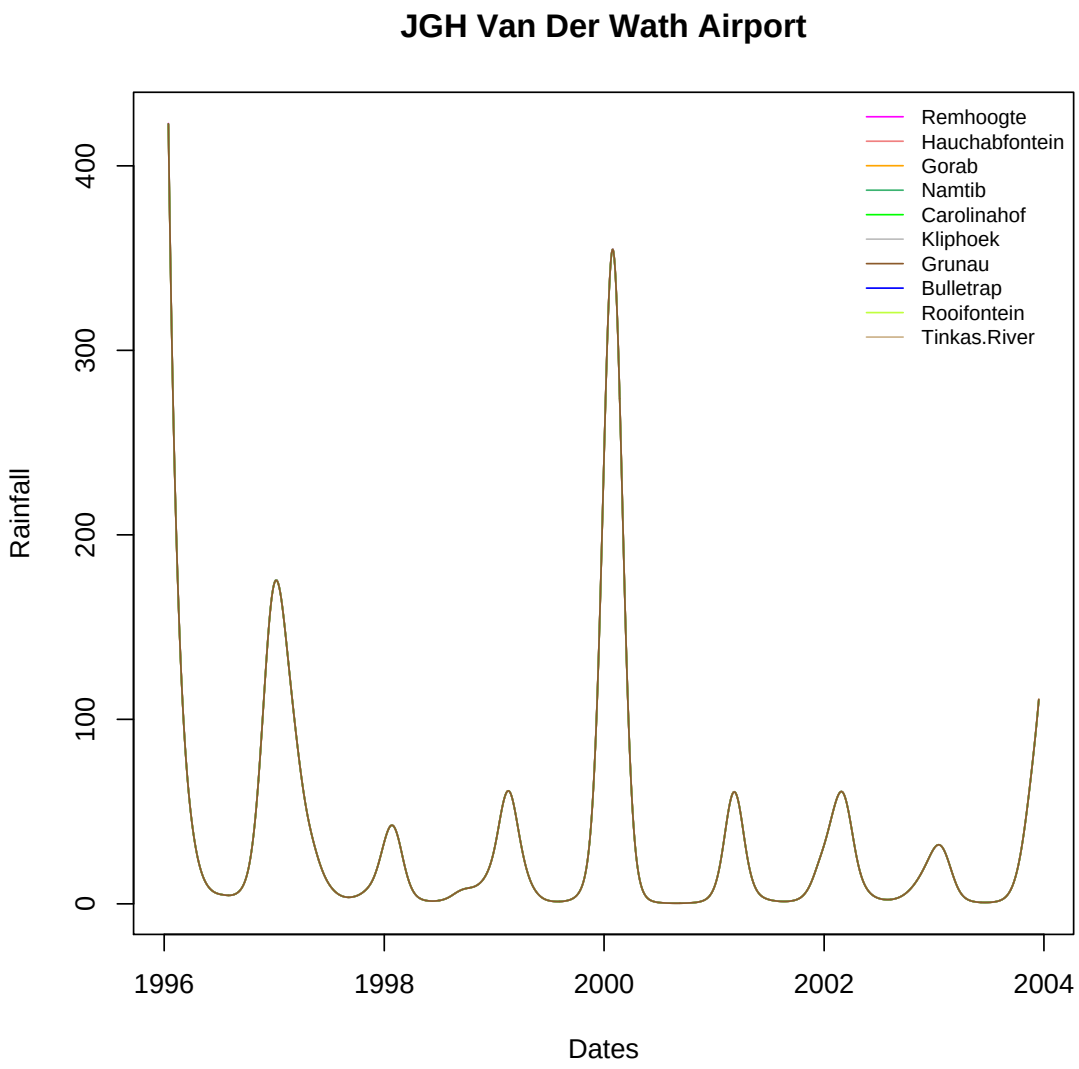


Figure 5.14: Rainfall: Station 1

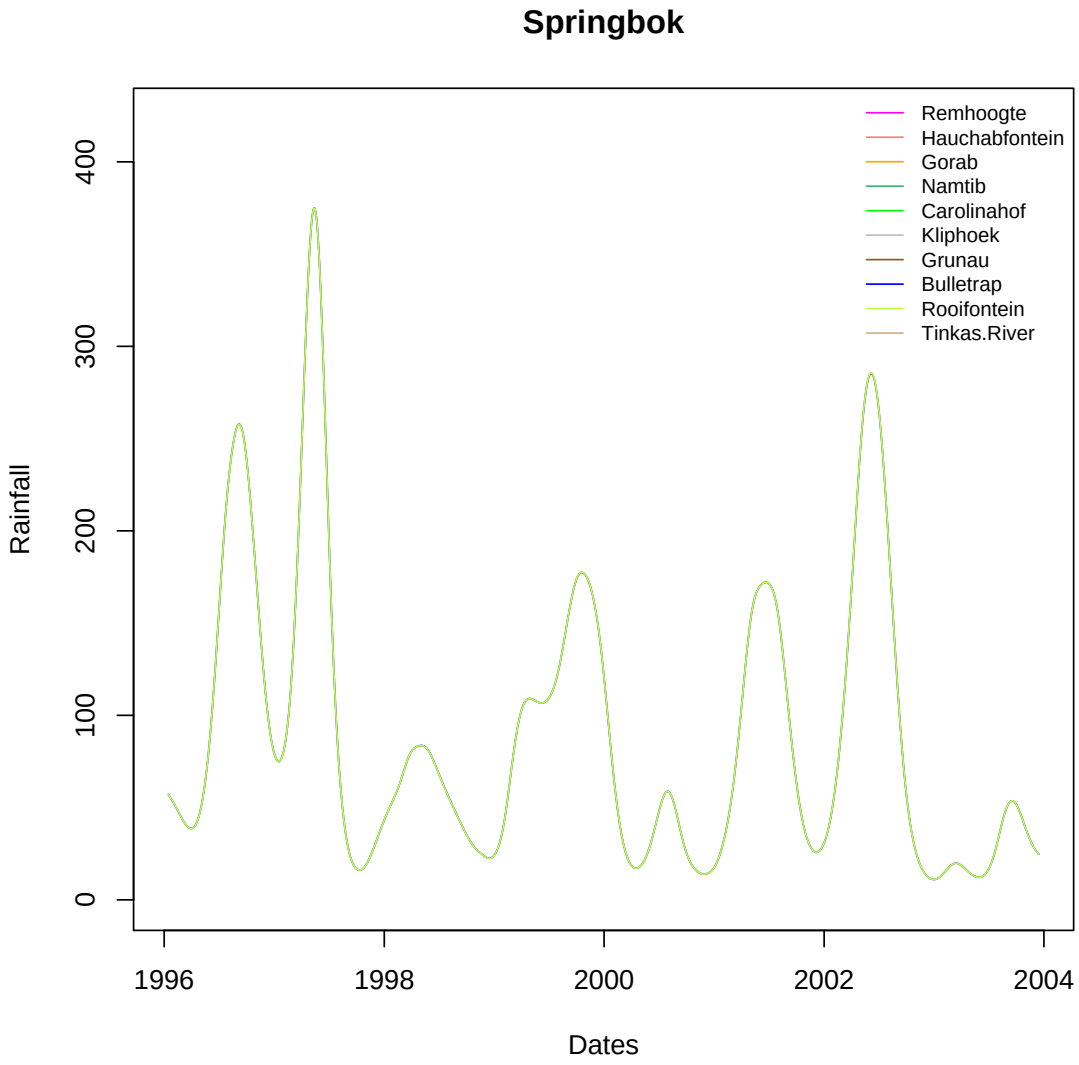


Figure 5.15: Rainfall: Station 2

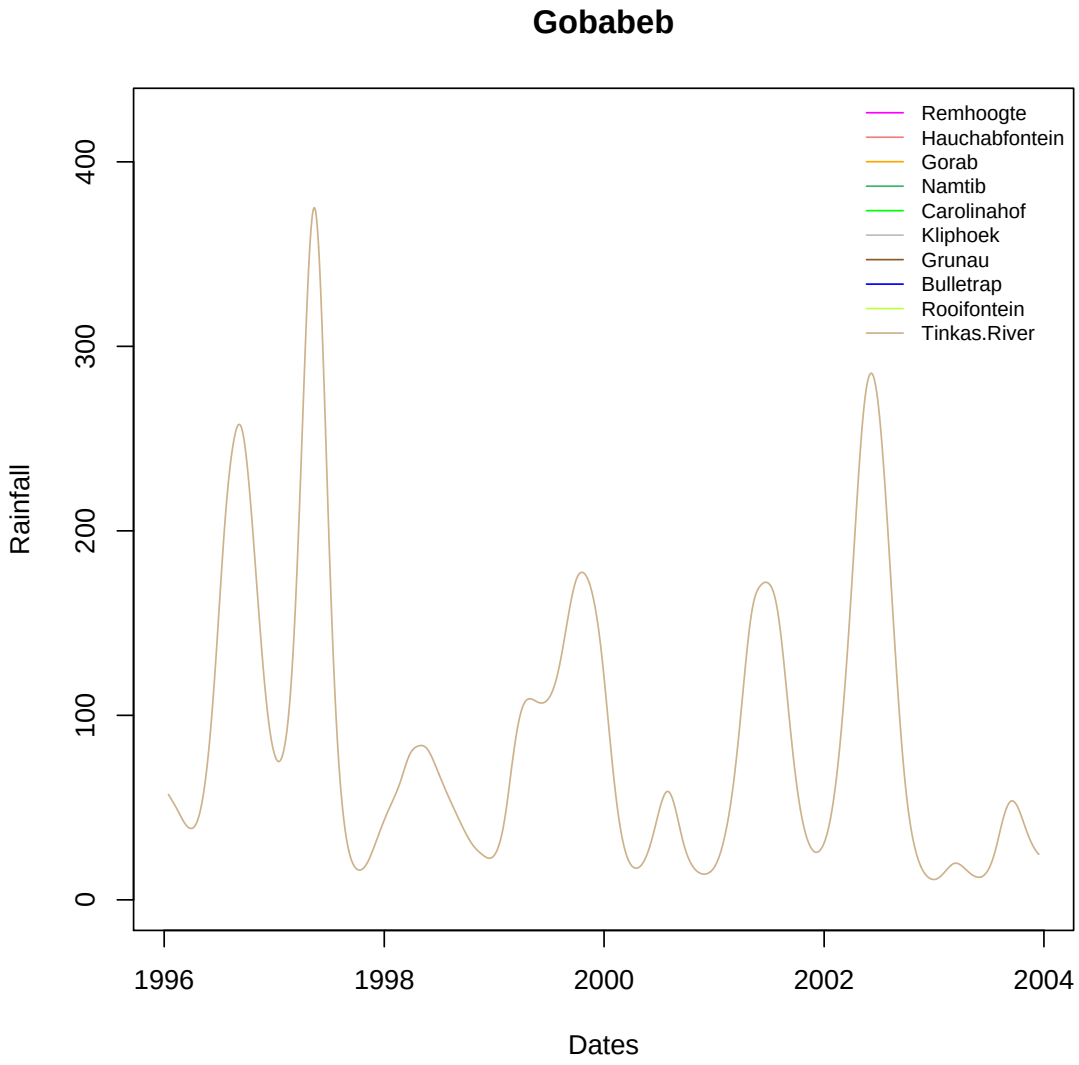


Figure 5.16: Rainfall: Station 3

Seven tree locations used weather station JGH Van der Wath Airport as a proxy for the true weather conditions in these locations as shown in Figure 5.2. The co-ordinates of these tree locations can be seen along with the elevation for each location. The coordinates for the weather station are also shown, in Table5.1.

Table 5.2: Table Showing Tree Locations using Weather Station JGH Van der Wath Airport

Tree Location	Latitude	Longitude	Elevation
Remhoogte	-23.98	16.27	1 370
Hauchabfontein	-24.58	16.07	1 200
Gorab	-25.14	16.41	1 460
Namtib	-25.99	16.16	1 300
Carolinahof	-26.40	16.90	1 260
Kliphoek	-27.26	16.78	1 240
Grunau	-27.94	18.16	940
JGH Van der Wath Airport	**-26.53**	**18.12**	**1 217**

The Sprinbok weather station was used for two tree locations as described in Table 5.3 with details of the weather station in Table 5.1.

Table 5.3: Table Showing Tree Locations using Weather Station Springbok

Tree Location	Latitude	Longitude	Elevation
Bulletrap	-29.45	17.83	840
Rooifontein	-30.05	18.25	780
Springbok	**-29.67**	**17.90**	**1 007**

Tinkas River was the only tree location that used the weather station Gobabeb as proxy and the details for the tree location and the weather station are shown in Table 5.4 and Table 5.1 respectively.

Table 5.4: Table Showing Tree Locations using Weather Station Gobabeb

Tree Location	Latitude	Longitude	Elevation
Tinkas River	-22.76	15.43	750
Gobabeb	**-23.57**	**15.05**	**400**

The response variables used in the functional regression models constructed with the continuous functions generated in the analysis above are shown in Table 5.5.

Table 5.5: Table Showing Response Variables

Variable Name
Canopy Diameter
Total Height
Height of First Branch
Basal Circumferenc
Circumference of First Branch
Number of branches Off main Stem
Number of Dichotomous Branches

Figure 5.17 shows a picture of a real Quiver tree in the desert.

Figure 5.17: Aloe Dichotoma also known as Quiver tree

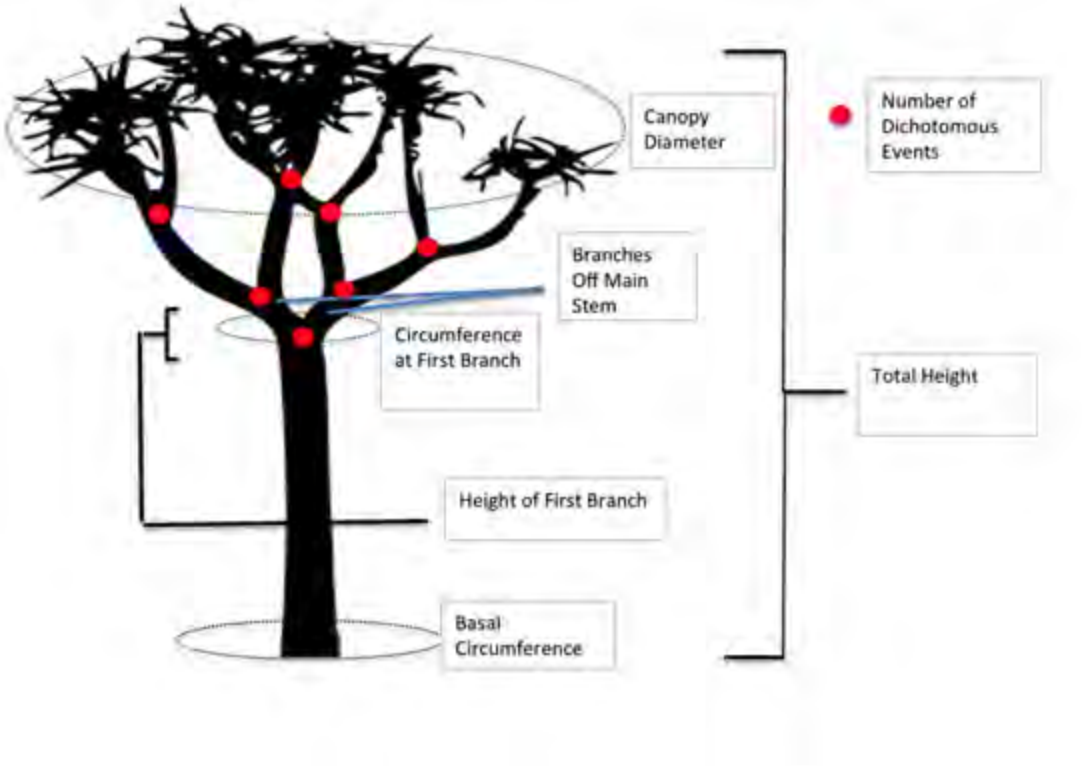


Figure 5.18: Labeled Tree Diagram

A diagrammatic representation of a Quiver tree is given in Figure 5.18. This is used to illustrate the definition of the different response variables.

The canopy diameter refers to everything above the trunk of the tree including all the stems and branches of the tree. The total height refers to the total height of the tree from the base to the canopy. The height of the first branch is the height of the first branch off the trunk of the tree. Basal circumference is the circumference measured at the base of the tree. Circumference at first branch is the circumference measured at the first branching point from the trunk of the tree. Branches off main stem refers to the number of branches off the main stem. Strictly speaking Quiver trees as the name Aloe dichotoma suggests, should always branch dichotomously but researchers observed that occasionally that wasn't the case. The deviation from the norm could be due to damage early in the tree's life that allows more than two branches to emerge. Although this could be random, it would be interesting if this was prevalent in one or more tree populations as this may indicate a constant form of disturbance or some other variant explained by climatic conditions.

Number of dichotomous branches refers to the total number of dichotomous events in the canopy.

In final parts of the analysis we are interested in how the architecture of Quiver trees differs between populations specifically in terms of latitude. Researchers expect to see taller trees in the north with less branches and possibly other difference that may be of interest. We compute the functional linear regression models with the associated F-ratio and the squared multiple correlation coefficient to assess the fit of our one predictor models used in the analysis

In Chapter 6 we present the results from the analysis.

Chapter 6

Results

This chapter contains all the functional regression models for each of the scalar response variables. We model each predictor variable separately using one predictor models for each response variable.

6.1 Canopy Diameter

In Figure 6.2,we see that the downward trend in the coefficient function between 1996 and 1999 agrees with lower than usual maximum winter temperatures in 1996 and 1997 and slightly lower than usual maximum summer temperatures in 1997, at least for Bulletrap, Tinkas River and Rooifontein. Clearly the lower maximum temperatures had a detrimental effect on canopy diameter. There is no lag in the effect of the maximum temperature on canopy diameters since fairly high maximum temperatures where observed in the summers of 1998 and 1999 with exceptionally high maximum winter temperatures in 1998, and the regression coefficient function turns upward between 1998 and 1999. Lower maximum summer temperatures in the 2000 summer and exceptionally lower maximum winter temperatures in 2001, especially for Bulletrap, Rooifontein and Tinkas River corresponds with the second downward slope of the regression coefficient function between 2000 and 2001. Another upward slope corresponds to the high maximum summer temperatures in 2002 and 2003 while lower maximum temperatures are observed for the first part of the 2004 summer, agreeing with the downward slope of the regression coefficient function towards 2004. It should be noted that the downward slope could be accentuated by fairly low maximum winter temperatures in Bulletrap, Rooifontein and Tinkas River and instability due to nearing the end of the time series.

Figure 6.1 shows the low dimensional regression coefficient function β with no roughness penalty smoothing. Comparing Figures 6.1 and 6.2, that maximum temperature has an

effect on canopy diameter only when it is relatively warm. This suggests that the substantial oscillations seen in Figure 6.1 over other conditions is irrelevant in defining the relationship.

The squared multiple correlation for this model is 0.98. The F-ratio is 16.82 with 7 and 2 degrees of freedom. It is important to not note that because a smoothing parameter has been used in these models, the F-distribution only represents an approximation to the null distribution for all these models.

The model uses a 95% confidence interval level and because these intervals are given point wise, they do not take into account and bias or the choice of smoothing parameter.

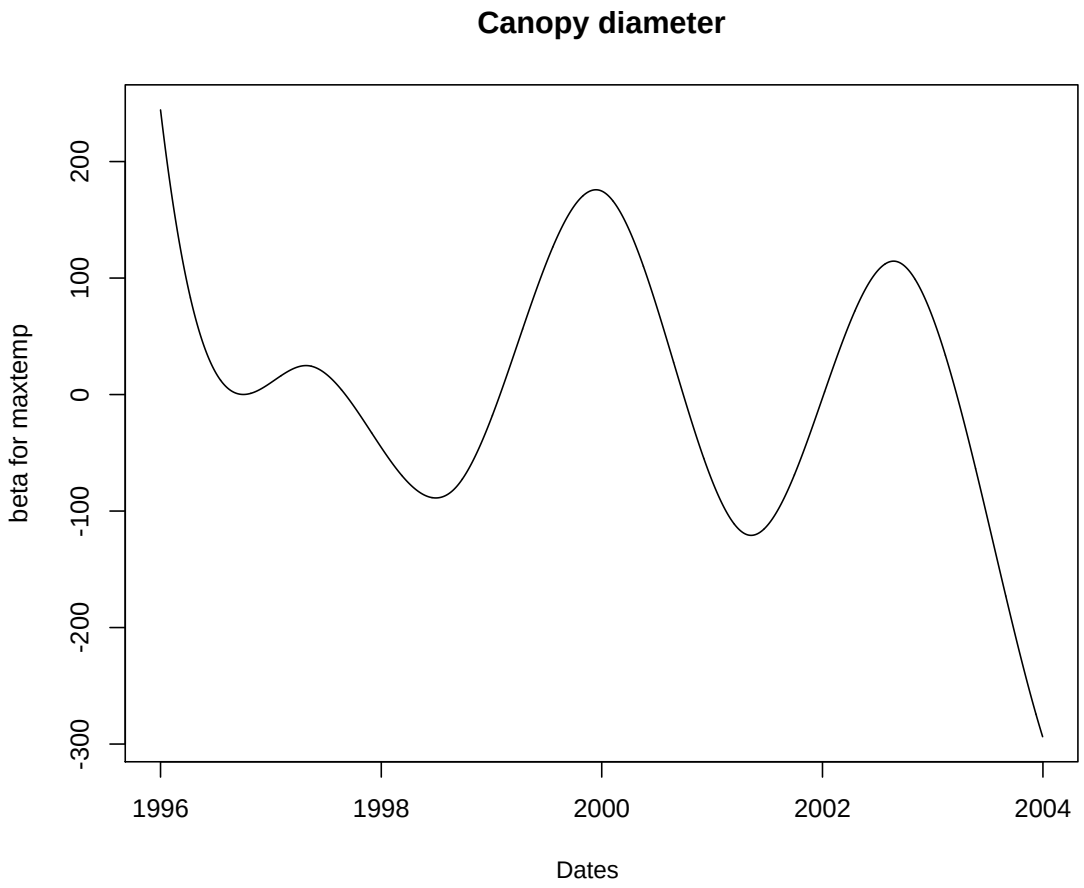
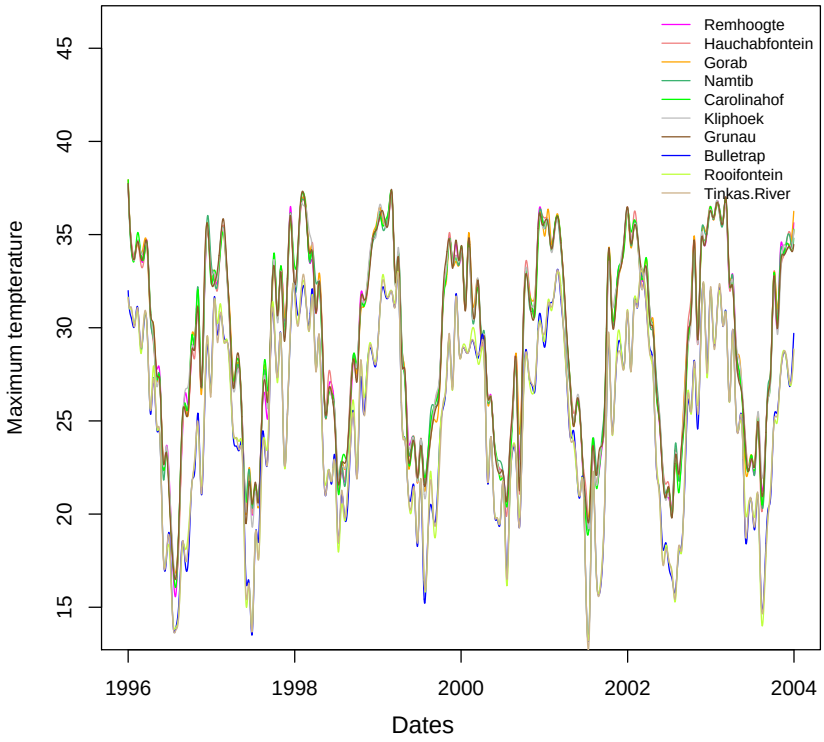


Figure 6.1: Maximum Temperature Beta Function



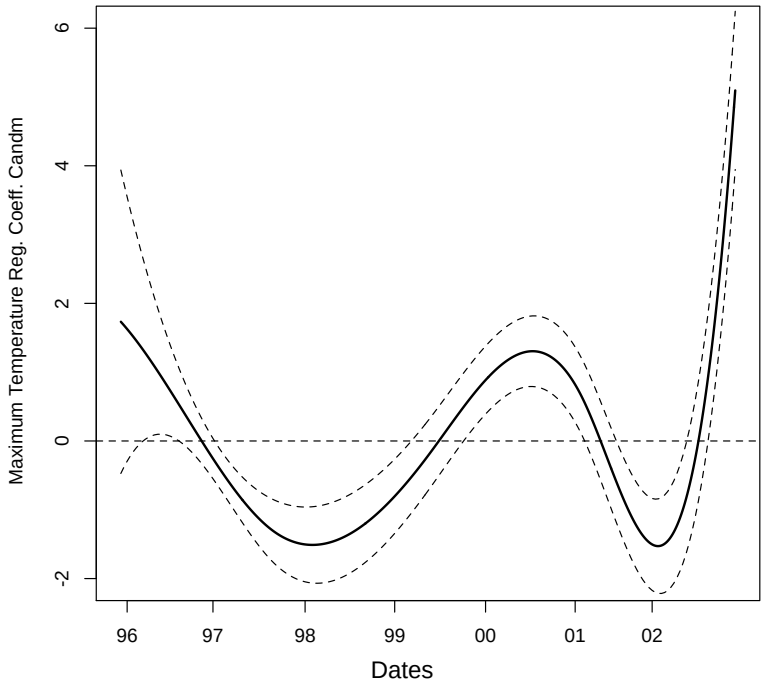


Figure 6.2: Maximum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Canopy Diameter with Confidence Intervals

In Figure 6.3 it appears that there is no important relationship between minimum temperature and canopy diameter as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.61 and a squared multiple correlation of 0.68

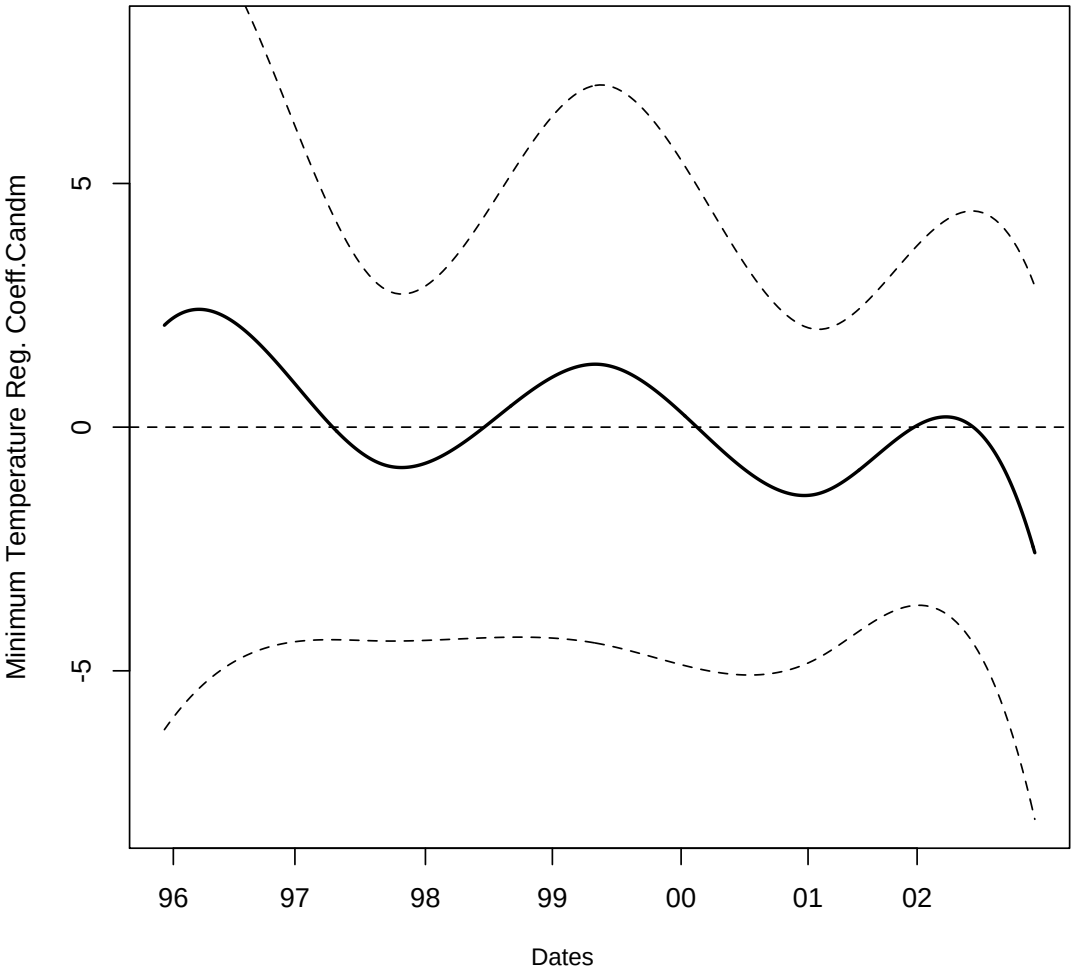


Figure 6.3: Penalized Minimum Temperature Beta Function with Confidence Intervals

One would have expected that rainfall would have some effect on the canopy diameter but

the results shown in Figure 6.4 suggest that the relationship between rainfall and canopy diameter(if any) is irrelevant in explaining the size of the canopy diameter. The F-ratio of this model with 7 and 2 degrees of freedom is 1.33 with a squared multiple correlation of 0.82.

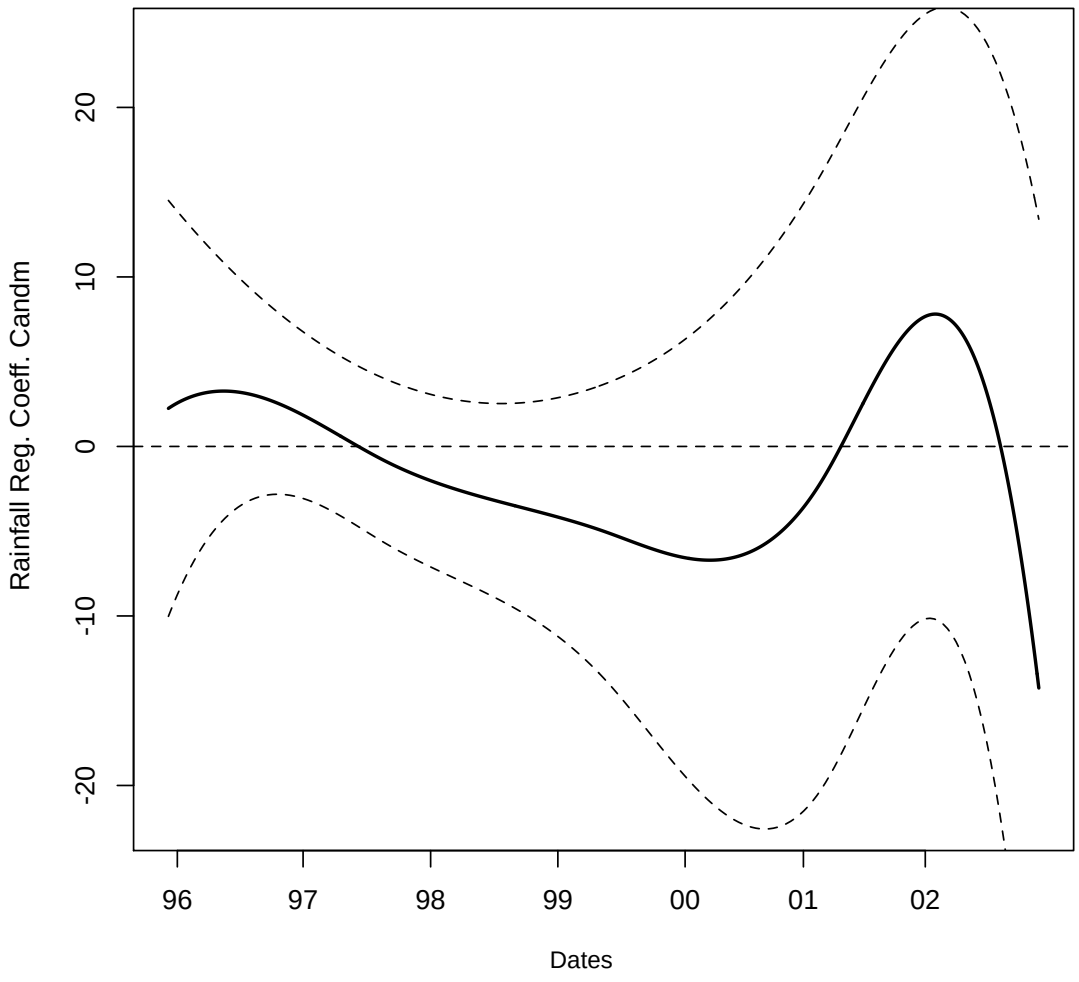
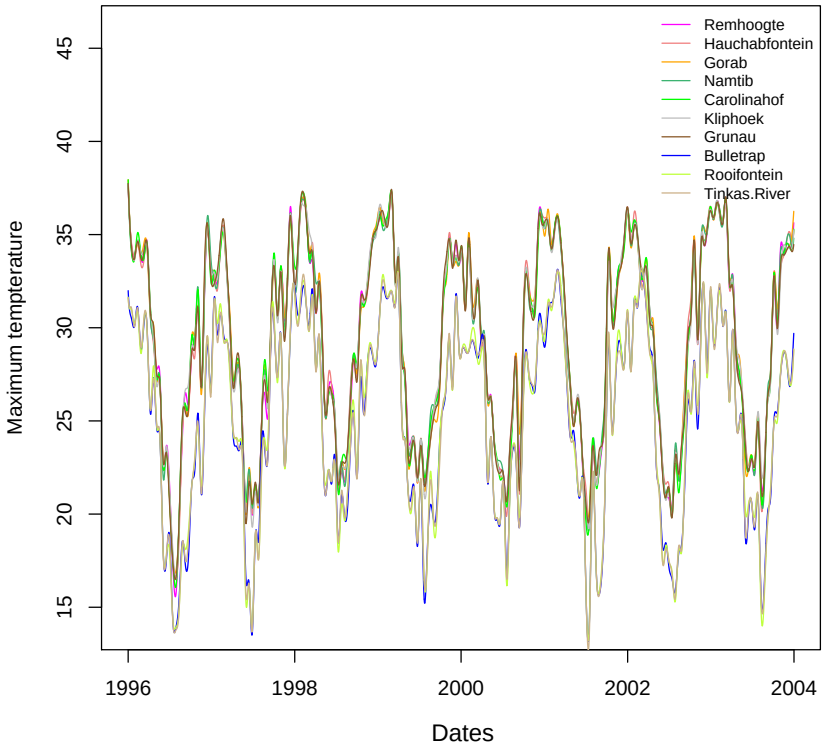


Figure 6.4: Penalized Rainfall Beta Function with Confidence Intervals

In Figure 6.5,represents the difference in the predicted value of canopy diameter for each

unit difference in maximum temperature if minimum temperature remains constant. This means that if maximum temperature differed by one unit and minimum temperature did not differ, the canopy diameter will be most influenced when maximum temperature is relatively high in summer and winter months. This corresponds with a upward slope between 1999 and 2001 where the maximum temperatures experienced in winter and summer were relatively higher. This means that when considering the effect of both maximum temperature the locations that experience relatively warmer weather conditions around both winter and summer will be the ones most affected.



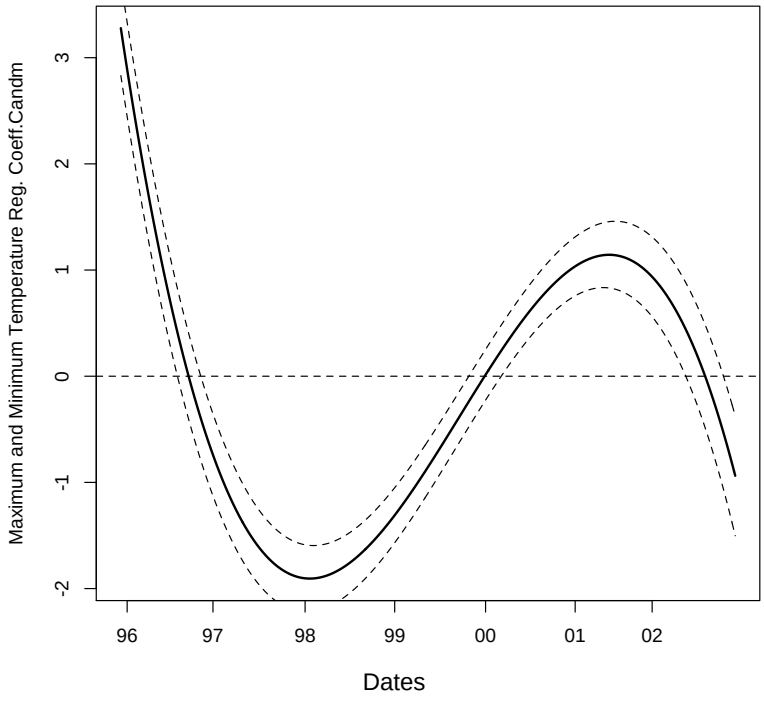
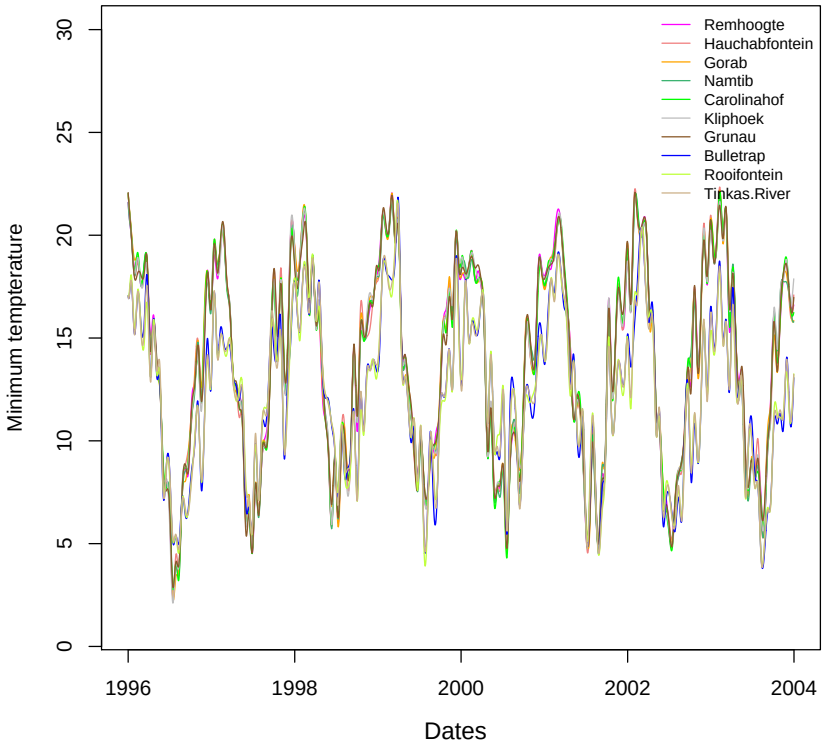


Figure 6.5: Maximum and Minimum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Canopy Diameter with Confidence Intervals

Figure 6.6, represents the difference in the predicted value of canopy diameter for each unit difference in minimum temperature if maximum temperature remains constant. This means that if minimum temperature differed by one unit and maximum temperature did not differ, the canopy diameter will be most influenced when minimum temperature is relatively low in the winter months and higher in the summer months. This would translate to a slower growth during winter months when the minimum temperature is relatively and faster growth once it gets warmer in the summer months where higher minimum temperatures are experienced. This corresponds to a downward regression coefficient slope when temperatures are relatively low such as prior to 1998 in Figure 6.6 and an upward slope post 2001 when temperatures are relatively higher. The confidence intervals confirm a significant effect when temperature is much lower and when it is relatively higher(and increasing).



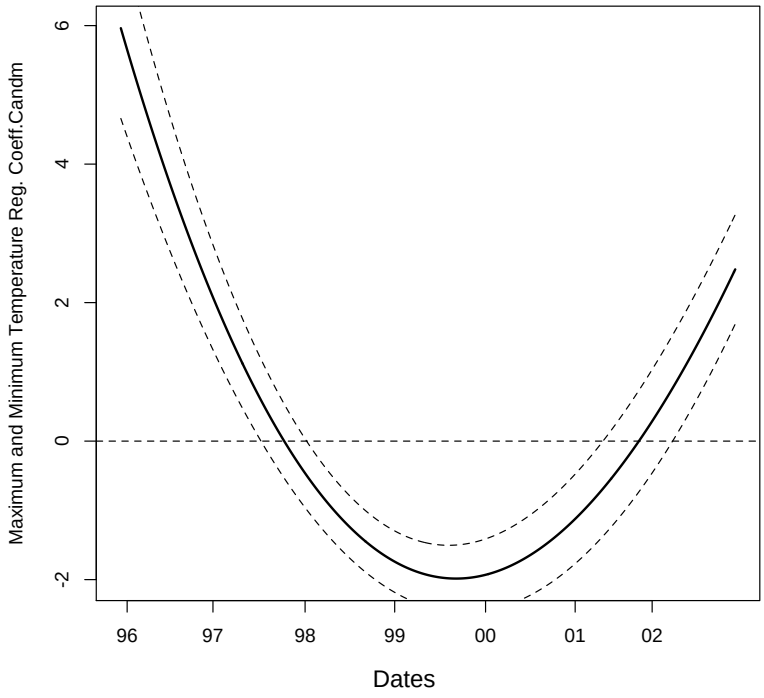


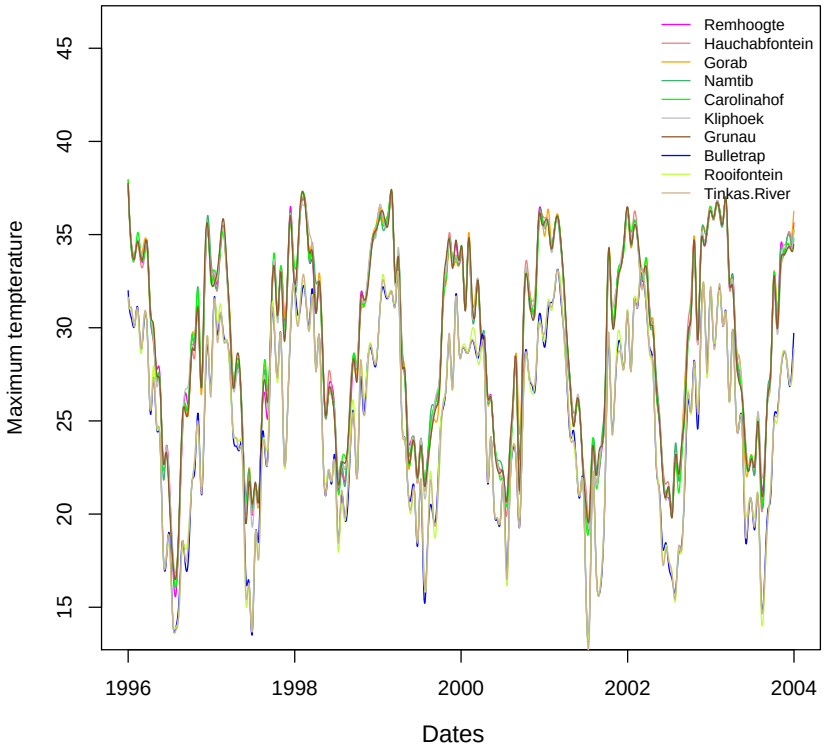
Figure 6.6: Maximum and Minimum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Canopy Diameter with Confidence Intervals

6.2 Total Height

In Figure 6.7,we see that the downward trend in the coefficient function between 1996 and 1998 agrees with lower than usual maximum winter temperatures in 1996 and 1997 and slightly lower than usual maximum summer temperatures in 1997, at least for Bulletrap, Tinkas River and Rooifontein. There is also a slight lag in the effect of the maximum temperature on total height since fairly high maximum temperatures where observed in the summers of 1998 and 1999 with exceptionally high maximum winter temperatures in 1998, but the regression coefficient function only turns upward between 1999 and 2000.The general upward slope of the regression coeffiecient funtion between 1998 and 2001 corresponds to relatively high summer maximum temperature values and relatively high winter maximum temperature values in the same period.Lower maximum summer temperatures in the 2000 summer and exceptionally lower maximum winter temperatures in 2001, especially for Bulletrap, Rooifontein and Tinkas River corresponds with the downwards slope of the regression coefficient function between 2000 and 2002. Clearly the much lower maximum temperatures in winter of 2001, particularly in Bulletrap, Tinkas River and Rooifontein, had a detrimental effect on total height. It should be noted that the downward slope could be accentuated by fairly low maximum winter temperatures in Bulletrap, Rooifontein and Tinkas River and instability due to nearing the end of the time series.

The confidence intervals in Figure 6.7 suggests that maximum temperature has an effect on total height when it is relatively warm. Otherwise across most periods the confidence interval includes zero suggest unimportant effects (if any).

The squared multiple correlation for this model is 0.89. The F-ratio is 2.39 with 7 and 2 degrees of freedom.



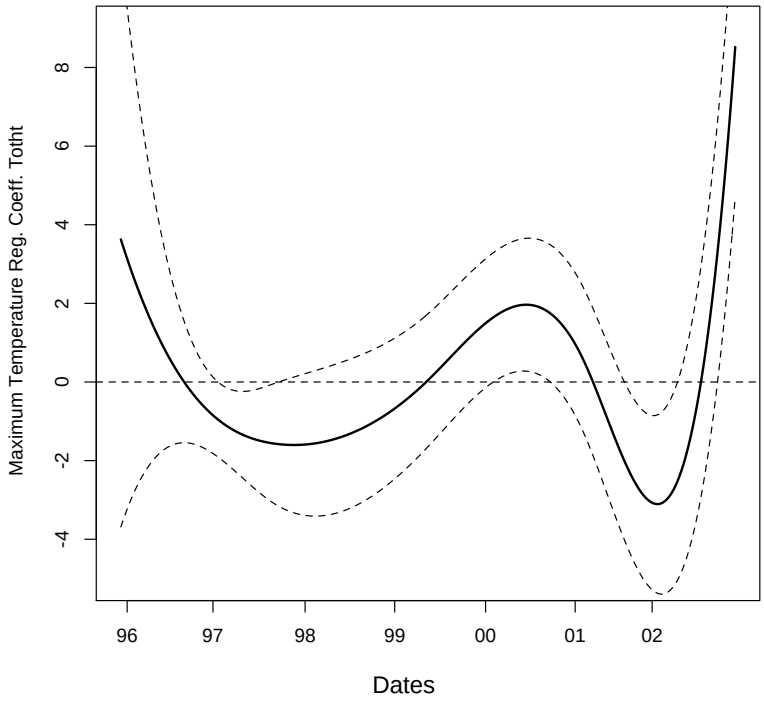


Figure 6.7: Maximum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Total Height with Confidence Intervals

In Figure 6.8 it appears that there is no important relationship between minimum temperature and total height as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.29 and a squared multiple correlation of 0.5.

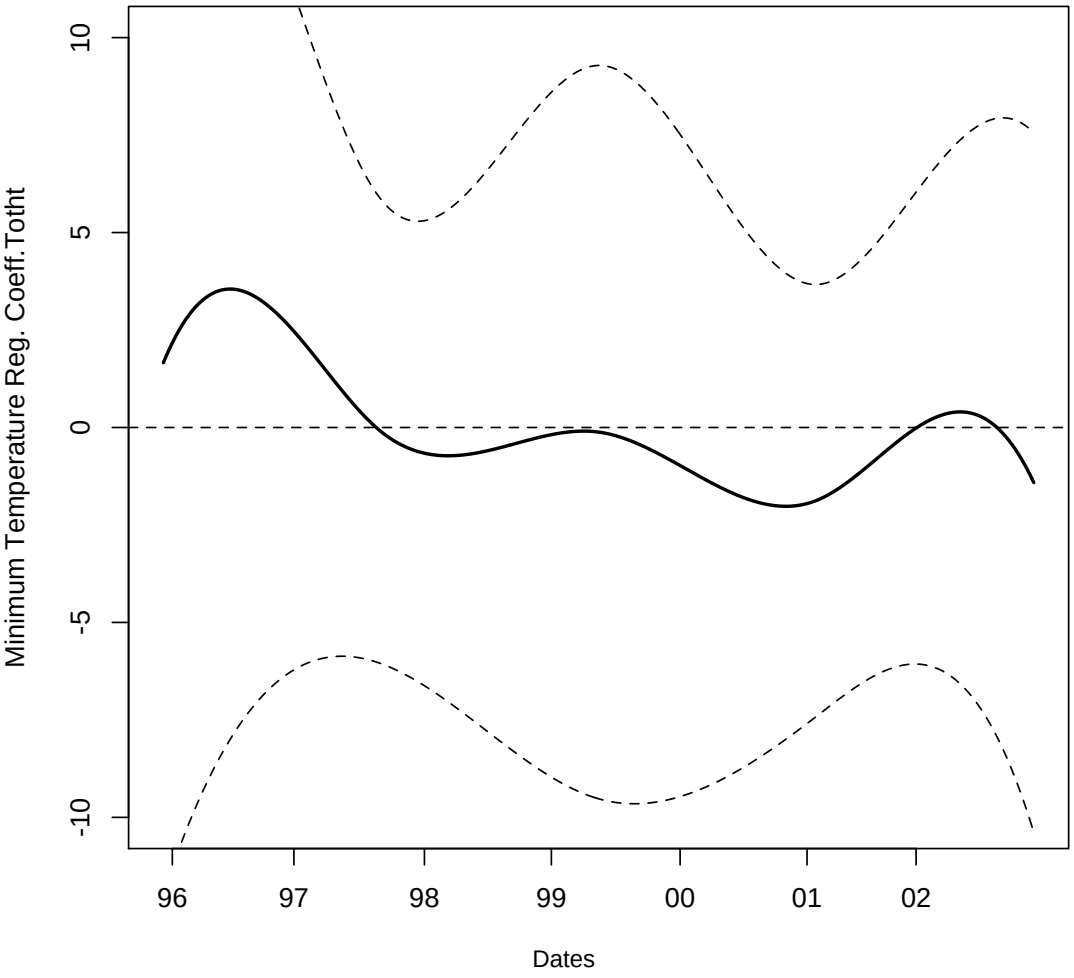


Figure 6.8: Penalized Minimum Temperature Beta Function with Confidence Intervals

In Figure 6.9 it appears that there is no important relationship between rainfall and total

height as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.65 and a squared multiple correlation of 0.57.

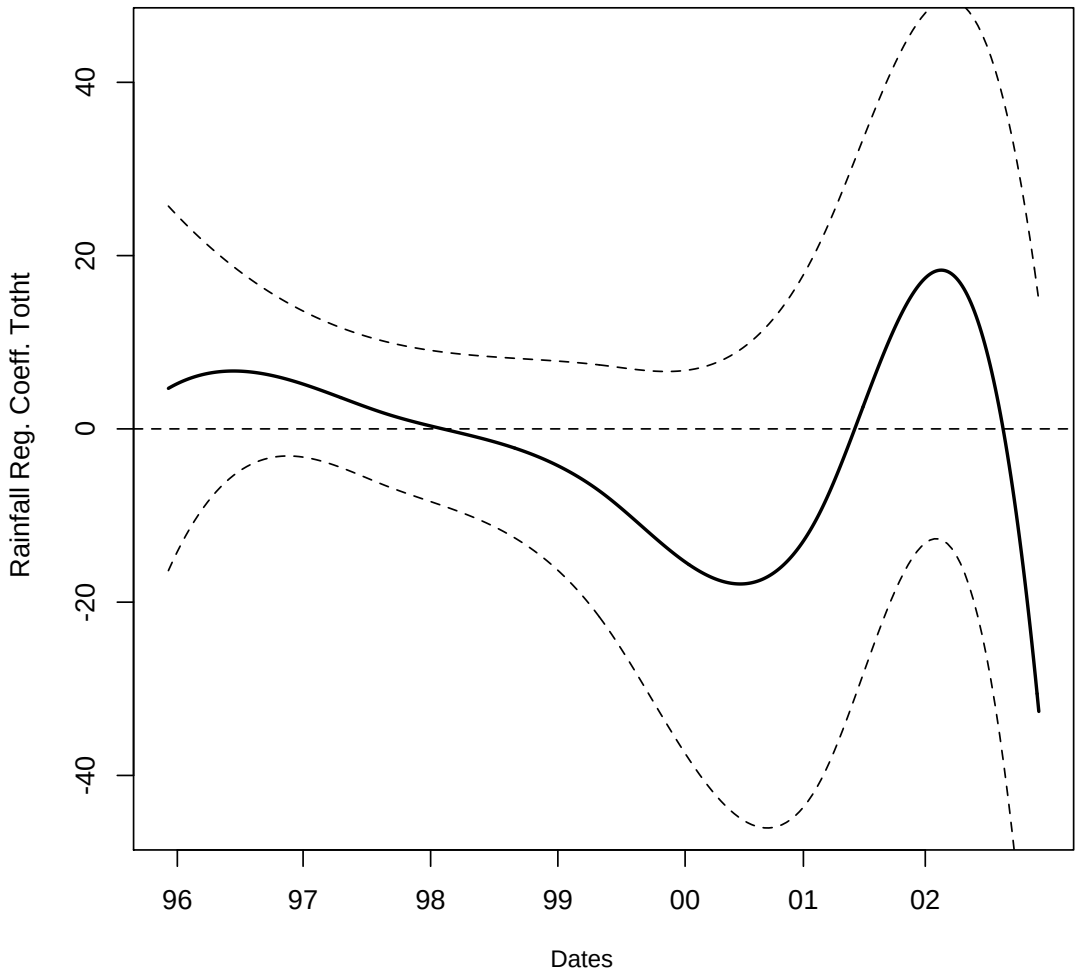
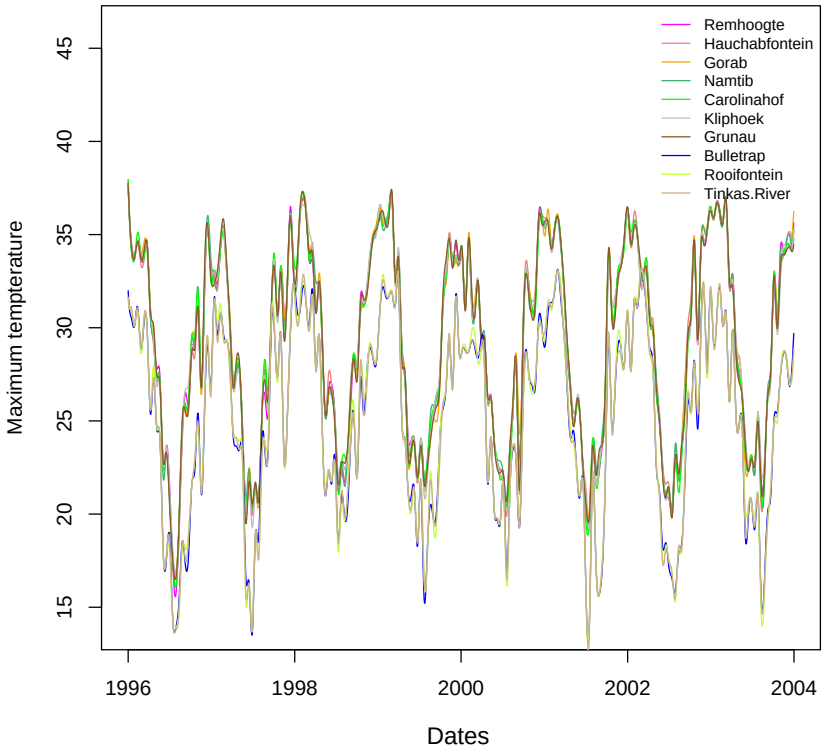


Figure 6.9: Penalized Rainfall Beta Function with Confidence Intervals

In Figure 6.10,represents the difference in the predicted value of total height for each unit difference in maximum temperature if minimum temperature remains constant. This means that if maximum temperature differed by one unit and minimum temperature did not differ,

the total height will be most influenced when maximum temperature is relatively high in summer and winter months. This corresponds with a upward slope between 1999 and 2001 where the maximum temperatures experienced in winter and summer were relatively higher. This means that when considering the effect of both maximum temperature the locations that experience relatively warmer weather conditions around both winter and summer will be the ones most affected.



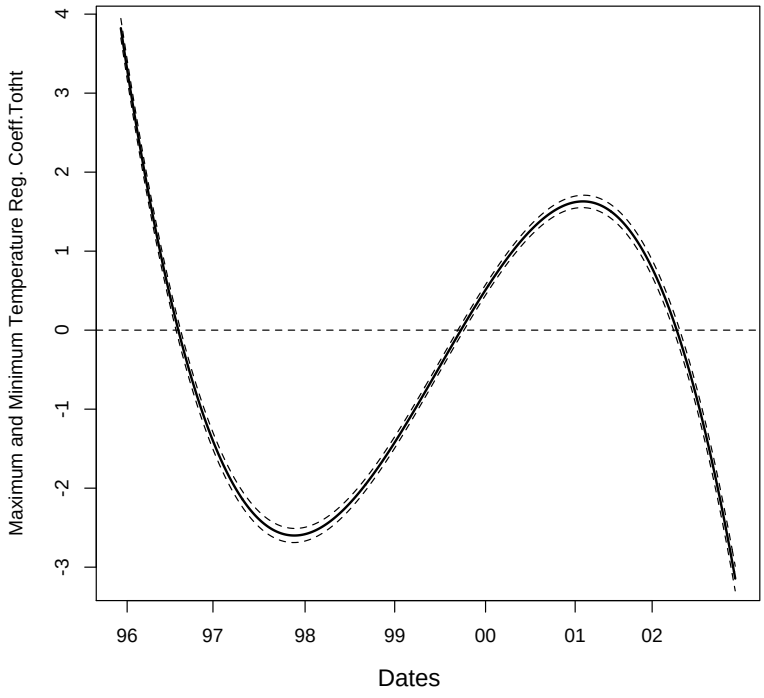
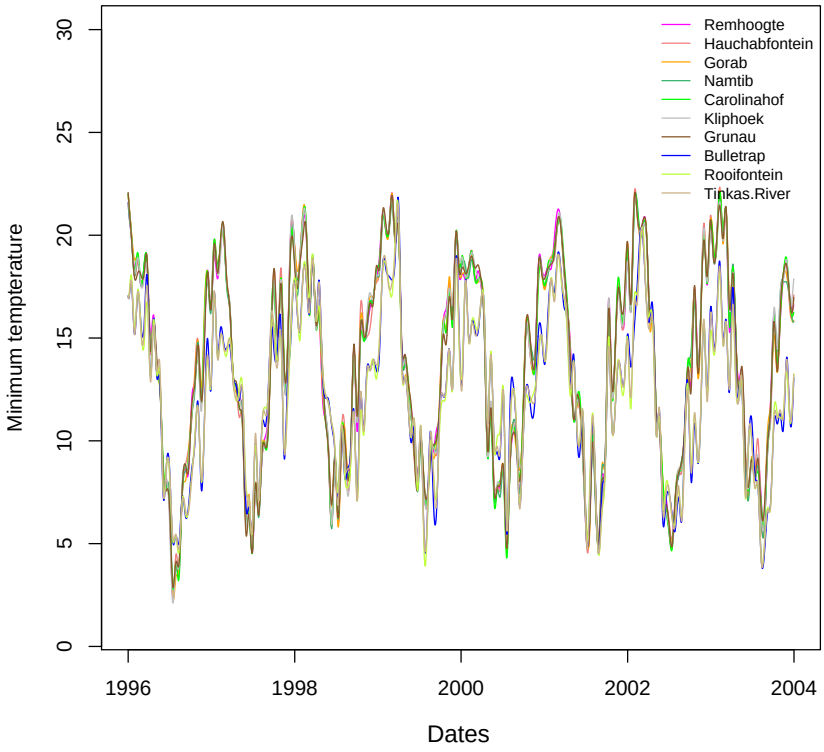


Figure 6.10: Maximum and Minimum Temperature : All Locations and Penalized Maximum and MinimumTemperature Beta Function for Total Height with Confidence Intervals

Figure 6.11, represents the difference in the predicted value of total height for each unit difference in minimum temperature if maximum temperature remains constant. This means that if minimum temperature differed by one unit and maximum temperature did not differ, the total height will be most influenced when minimum temperature is relatively low in the winter months and higher in the summer months. This would translate to a slower growth during winter months when the minimum temperature is relatively lower and faster growth once it gets warmer in the summer months where higher minimum temperatures are experienced. This corresponds to a downward regression coefficient slope when temperatures are relatively low such as prior to 1998 in Figure 6.11 and an upward slope post 2001 when temperatures are relatively higher. The confidence intervals confirm a significant effect when temperature is much lower and when it is relatively higher(and increasing).



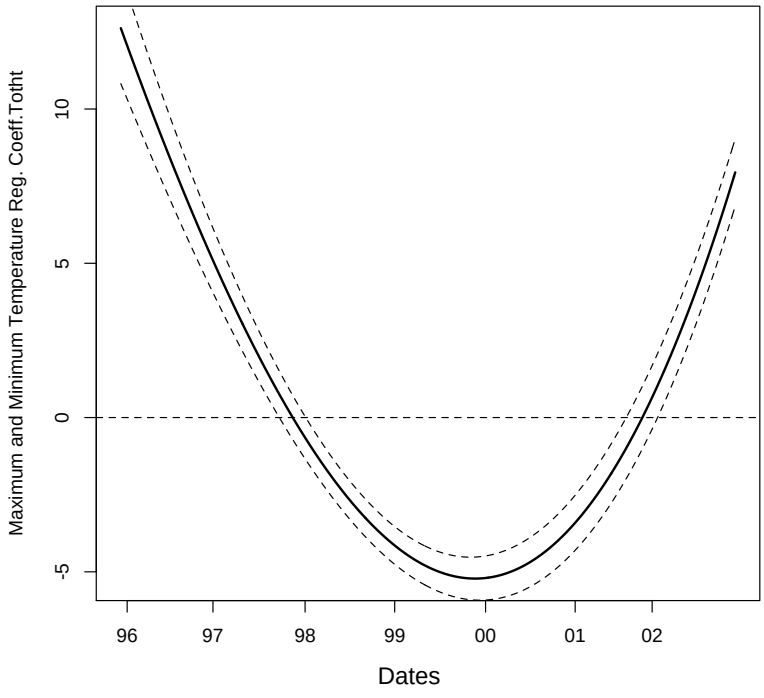


Figure 6.11: Maximum and Minimum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Total height with Confidence Intervals

6.3 Height of First Branch

In Figure 6.12 it appears that there is no important relationship between maximum temperature and height of the first branch as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 1.37 and a squared multiple correlation of 0.83.

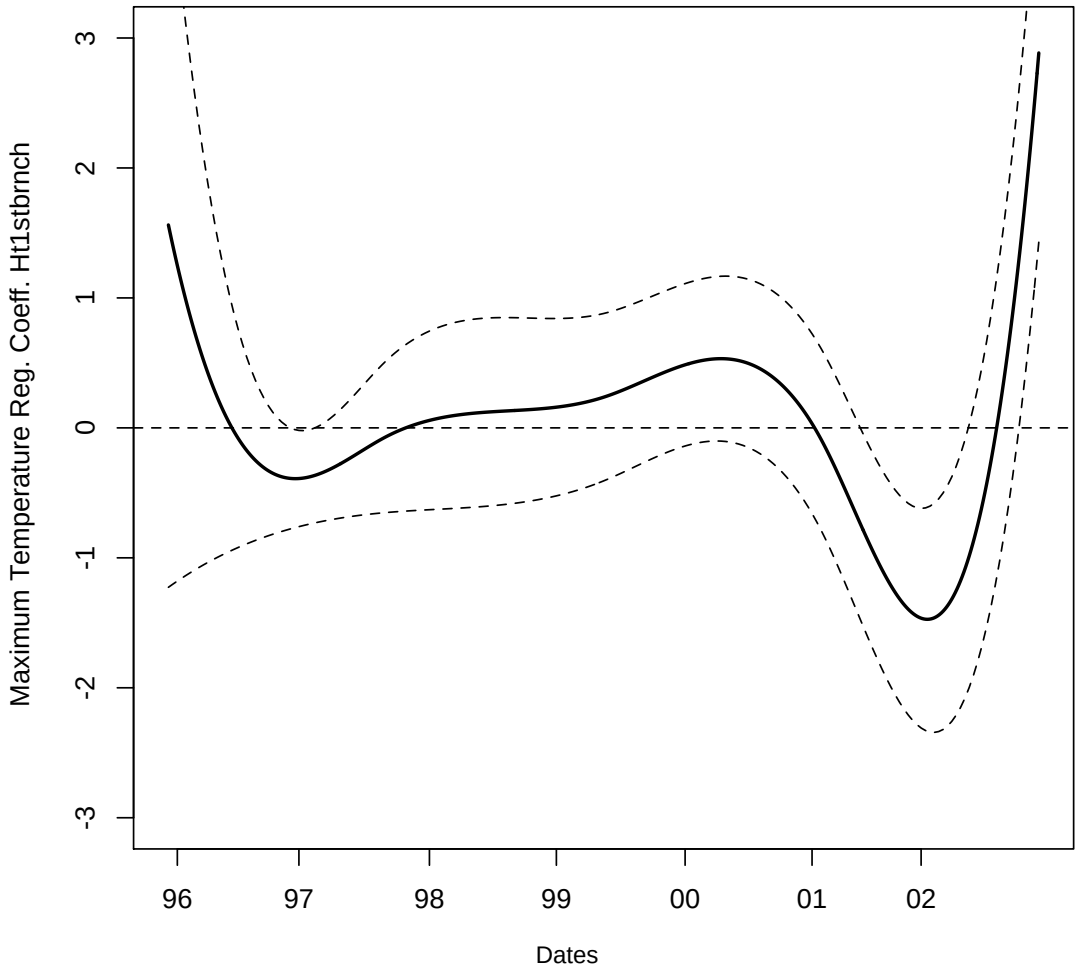


Figure 6.12: Penalized Maximum Temperature Beta Function with Confidence Intervals

In Figure 6.13 it appears that there is no important relationship between minimum temperature and height of the first branch as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 1.09 and a squared multiple correlation of 0.79.

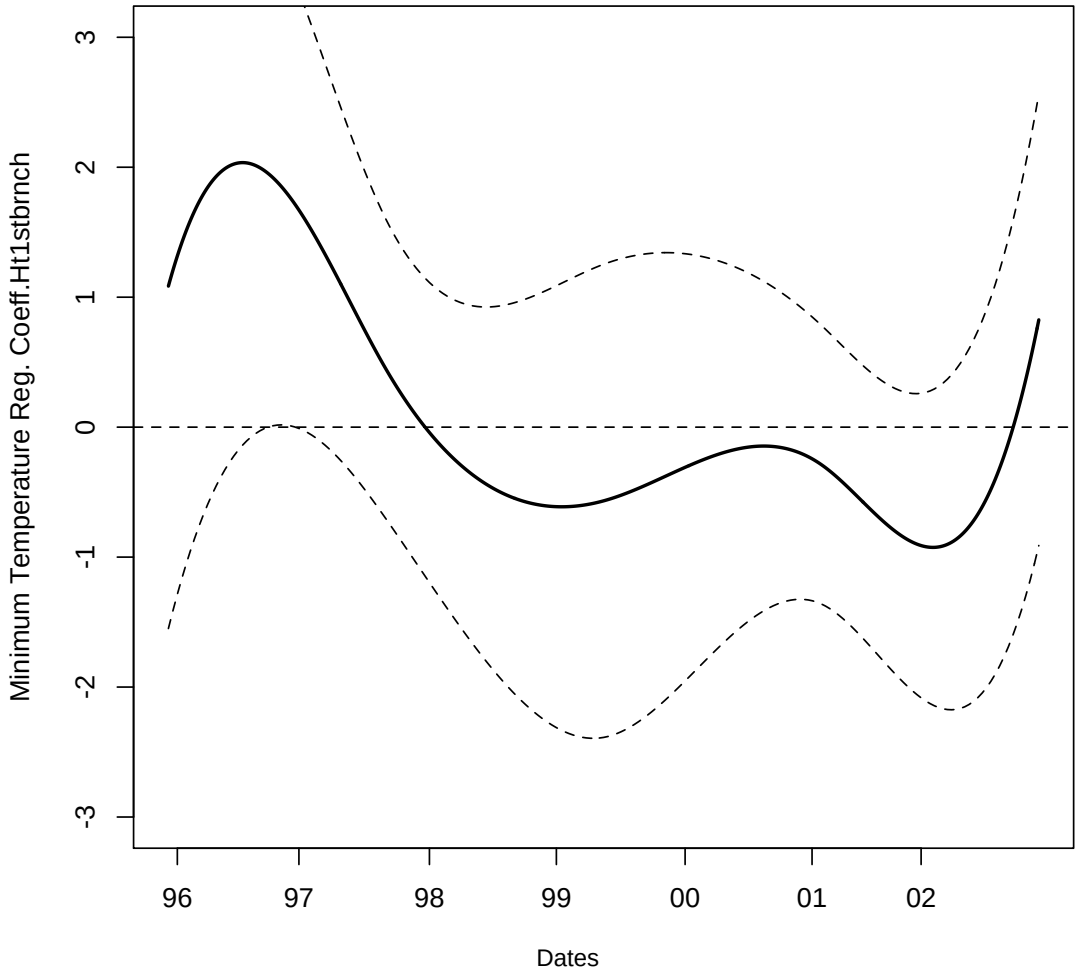


Figure 6.13: Penalized Minimum Temperature Beta Function with Confidence Intervals

In Figure 6.14 it appears that there is no important relationship between rainfall and

height of the first branch as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.96 and a squared multiple correlation of 0.77.

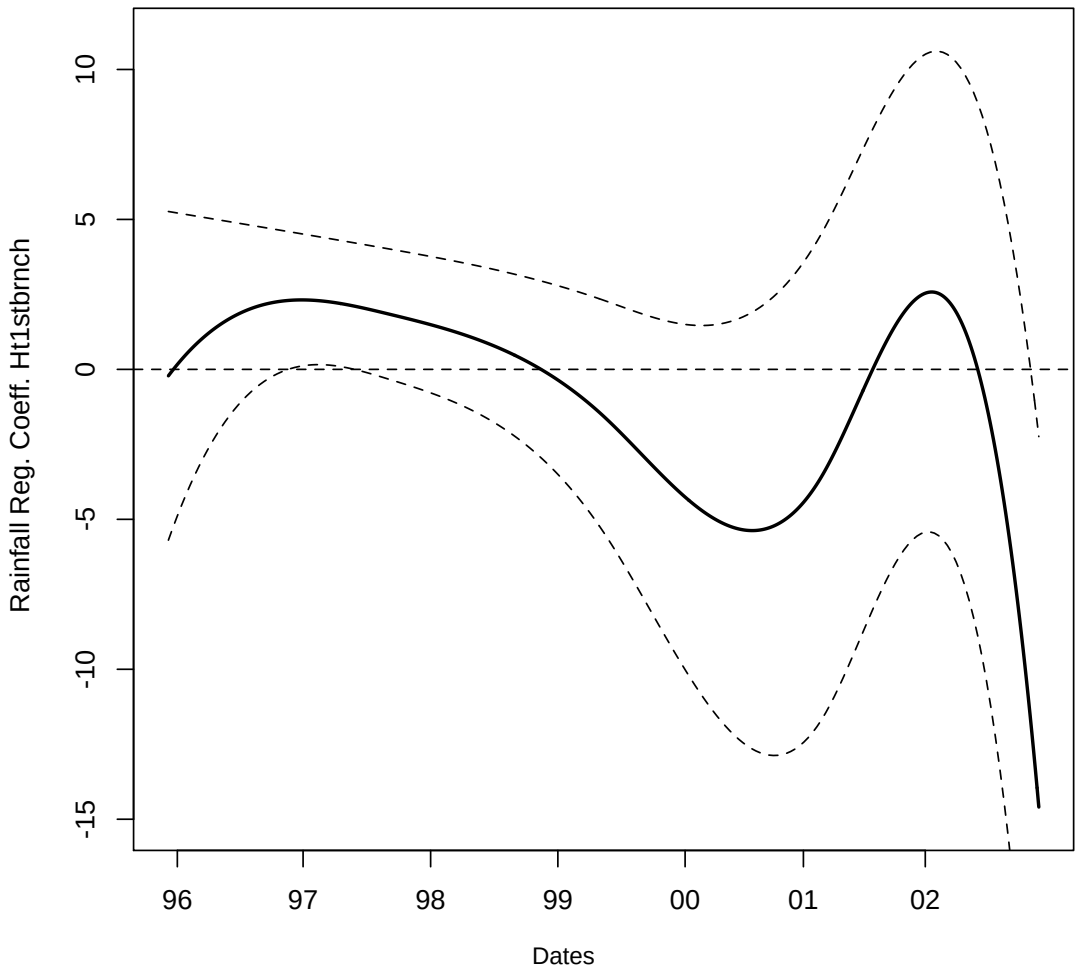


Figure 6.14: Penalized Rainfall Beta Function with Confidence Intervals

In Figure 6.15 it appears that there is no important relationship between height of the first branch temperature, even when considering both maximum and minimum temperature

effects together because the confidence intervals across the entire function contain zero.

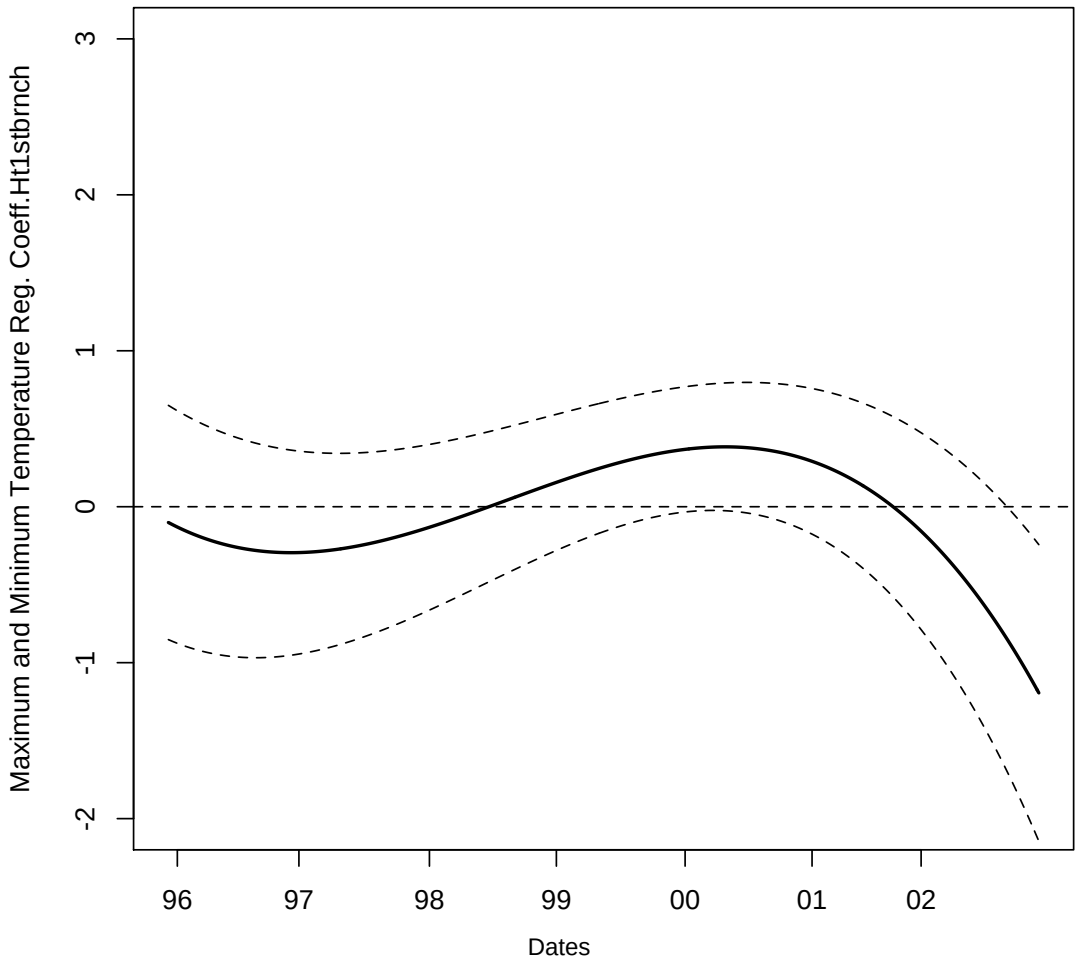


Figure 6.15: Maximum and Minimum Temperature Beta Function

In Figure 6.16 it appears that there is no important relationship between the height of the first branch and temperature, when considering the effect of minimum temperature when maximum temperature is kept constant because the confidence intervals across the entire function contain zero.

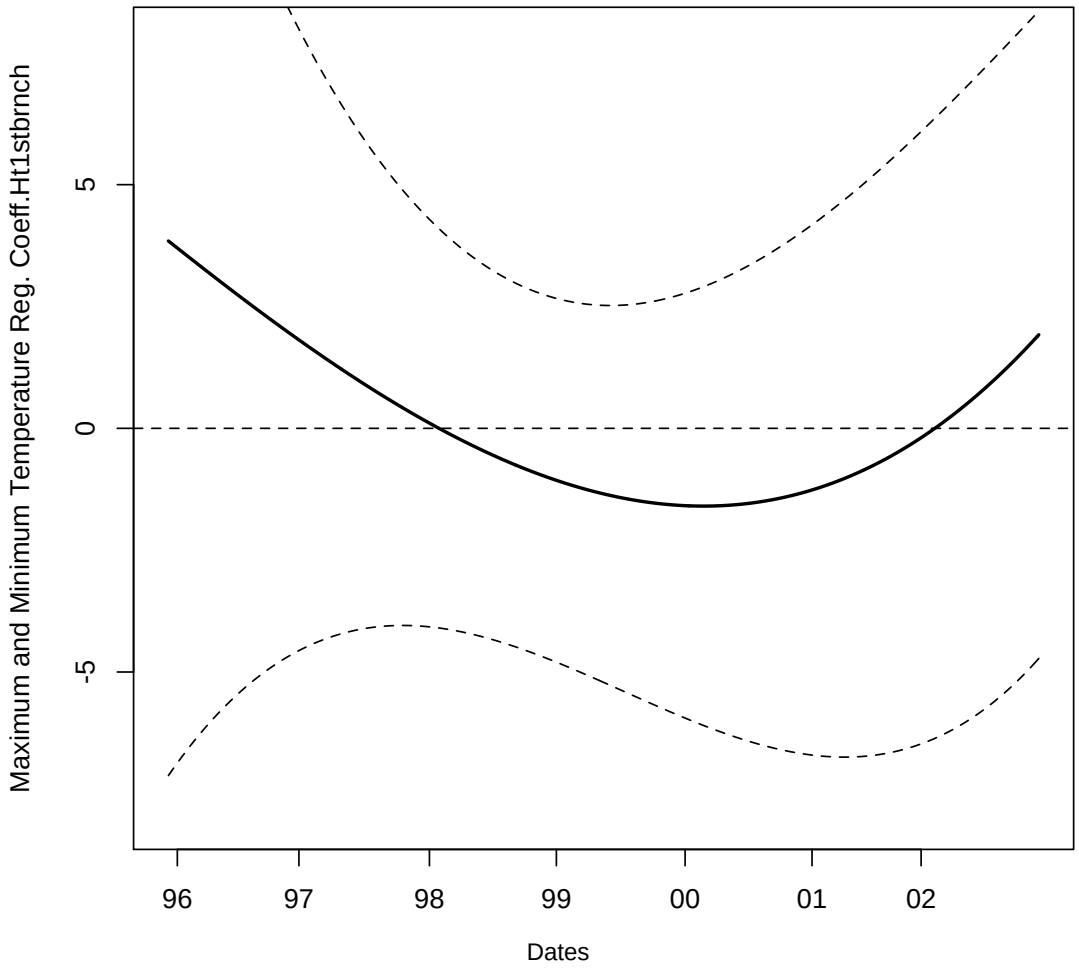


Figure 6.16: Maximum and Minimum Temperature Beta Function

6.4 Basal Circumference

In Figure 6.17,we see that the downward trend in the coefficient function between 1996 and 1997 agrees with lower than usual maximum winter temperatures in 1996 and 1997 and slightly lower than usual maximum summer temperatures in 1997, at least for Bulletrap,

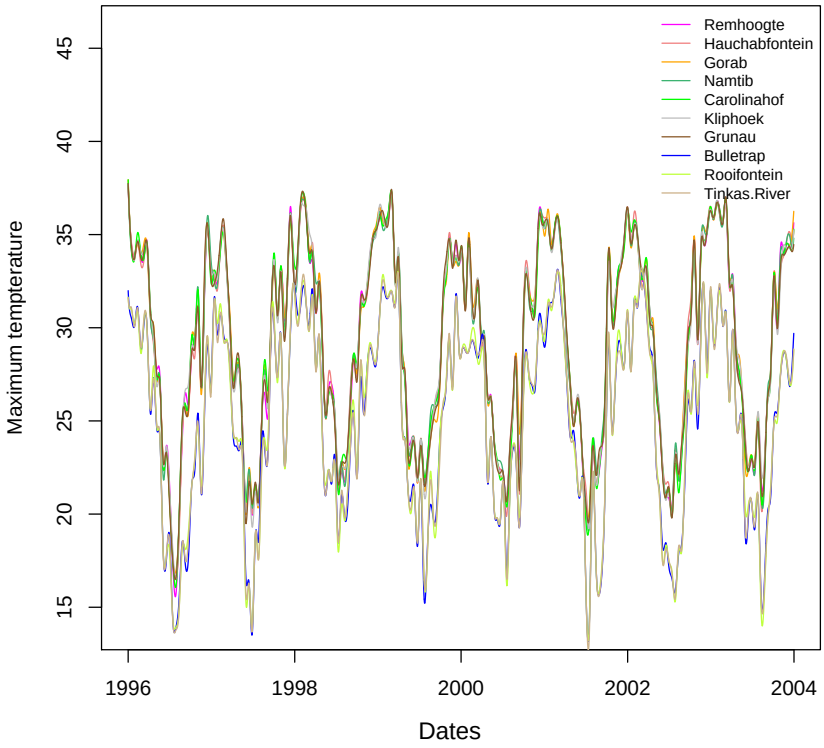
Tinkas River and Rooifontein.

There is a slight lag in the effect of the maximum temperature on basal circumference since fairly high maximum temperatures where observed in the summers of 1998 and 1999 with exceptionally high maximum winter temperatures in 1998, but the regression coefficient function only turns upward between 1999 and 2000. During the period 1997 to 1999 the effect on the regression coefficient is fairly steady.

The relatively low maximum summer temperatures in the 2000 have no apparent effect as the regression coefficient continues to increase. however, the sharp decrease in the regression coefficient slope in 2001 reflects the much lower maximum temperature observed in the winter of 2001and possibly a delayed response to the low temperature experienced in 2000. Clearly the lower maximum temperatures had a detrimental effect on basal circumference which may have been exacerbated by the delayed response discussed.Another upward slope in 2002 corresponds to the high maximum summer temperatures in 2002 and 2003 while lower maximum temperatures are observed for the first part of the 2004 summer, agreeing with the downward slope of the regression coefficient function towards 2004. It should be noted that the downward slope could be accentuated by fairly low maximum winter temperatures in Bulletrap, Rooifontein and Tinkas River and instability due to nearing the end of the time series.

In Figure 6.17, we see that a predictor for high basal circumference is a relatively high maximum temperature around the months of August and September. The 95% confidence intervals between 1996 and mid 1999 contain zero which suggests that the influence of maximum temperature on basal circumference in those periods is irrelevant. We see a peak between mid 1999 and mid 2001. This pattern suggests a contrast between late autumn and early summer with more emphasis on late autumn.This also suggests that the impact of maximum temperature observed between mid 1999 and 2001 is most likely the trend expected.

The squared multiple correlation for this model is 0.97. The F-ratio is 10.92 with 7 and 2 degrees of freedom



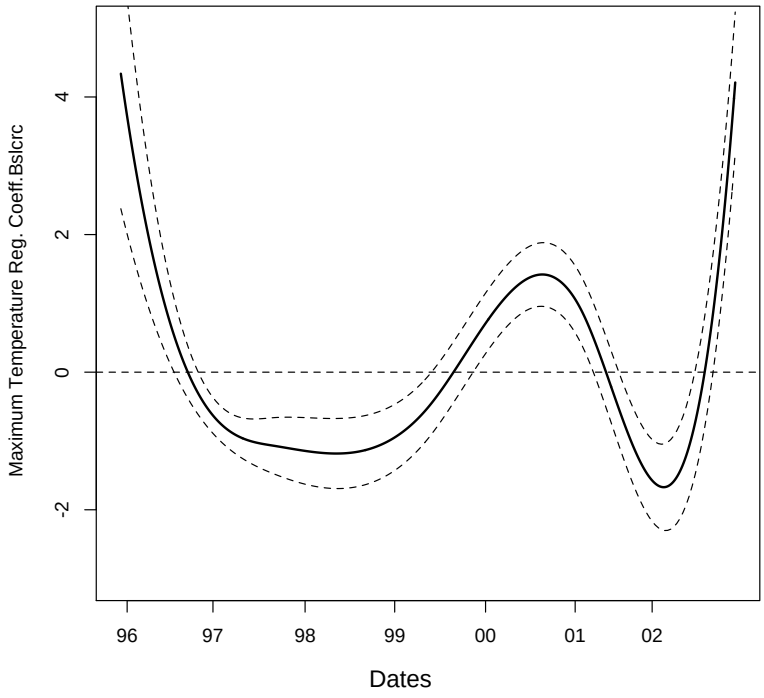


Figure 6.17: Maximum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Basal Circumference with Confidence Intervals

In Figure 6.18 it appears that there is no important relationship between minimum temperature and basal circumference as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.29 and a squared multiple correlation of 0.83.

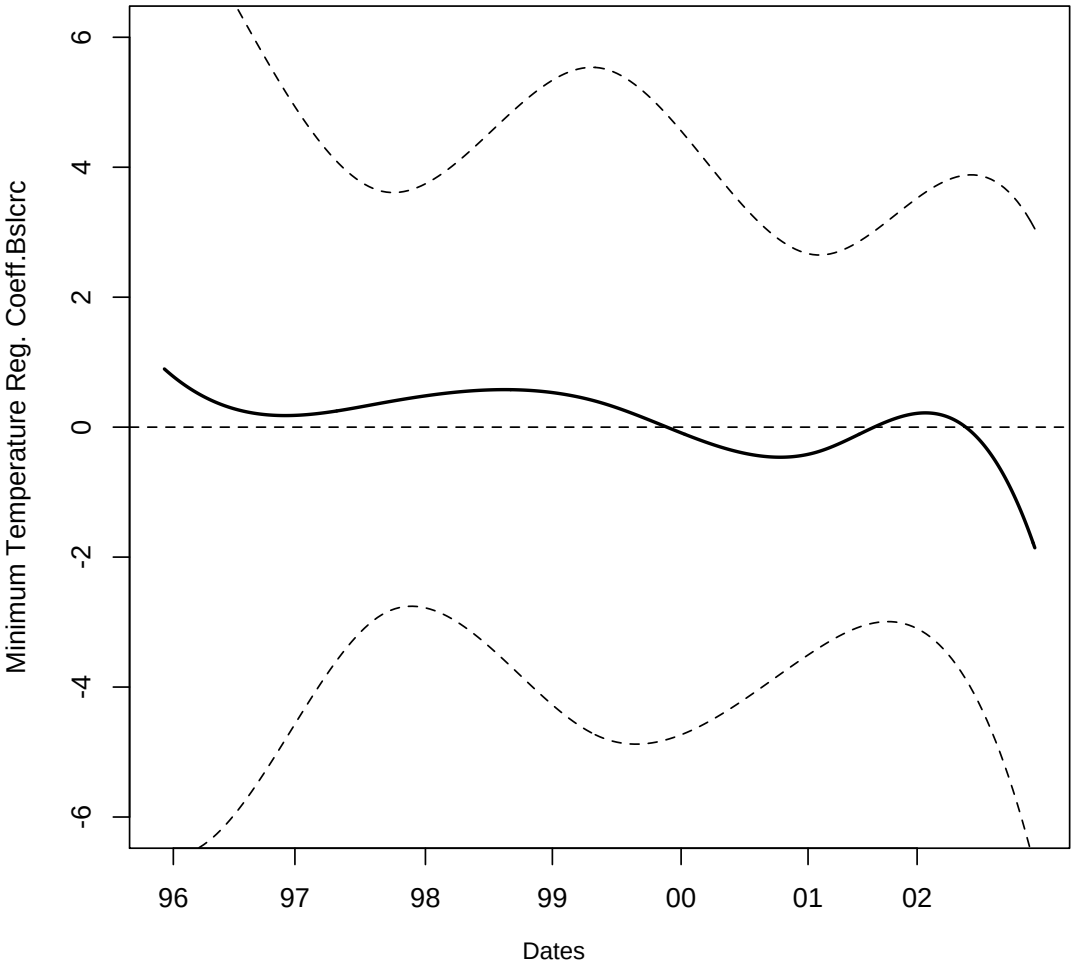


Figure 6.18: Penalized Minimum Temperature Beta Function with Confidence Intervals

In Figure 6.19 it appears that there is no important relationship between rainfall and basal

circumference as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.40 and a squared multiple correlation of 0.58.

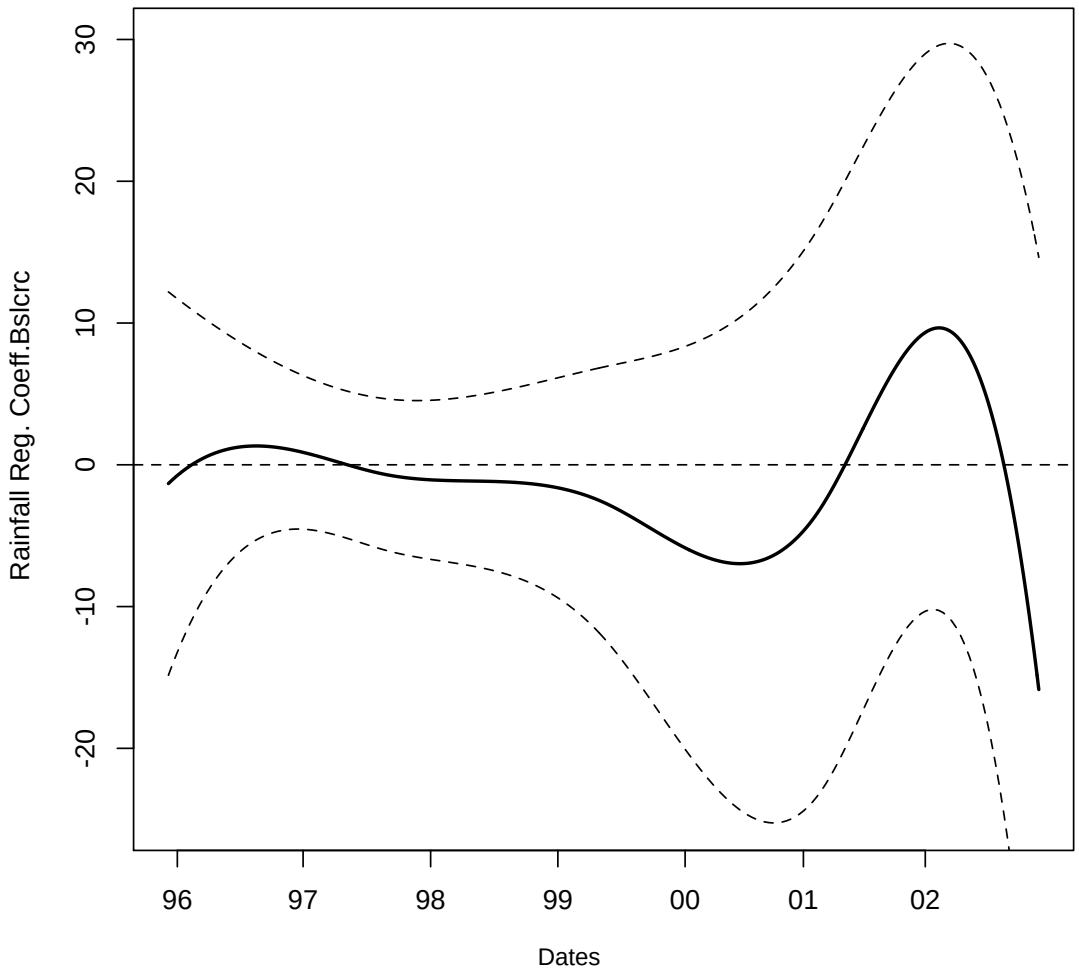
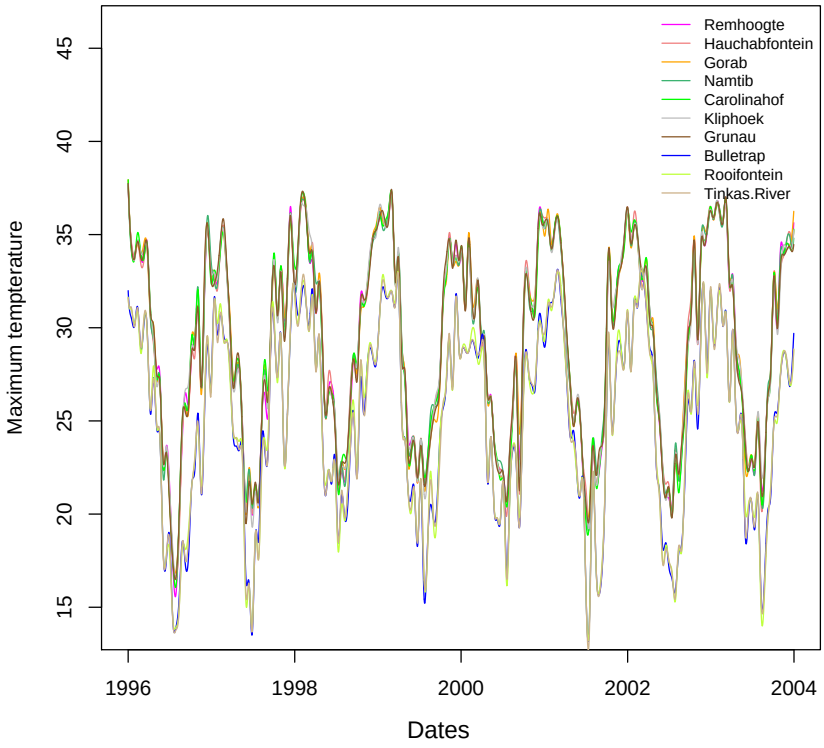


Figure 6.19: Penalized Rainfall Beta Function with Confidence Intervals

In Figure 6.20,represents the difference in the predicted value of basal circumference for each unit difference in maximum temperature if minimum temperature remains constant.

This means that if maximum temperature differed by one unit and minimum temperature did not differ, the basal circumference will be most influenced when maximum temperature is relatively high in summer and winter months. This corresponds with a upward slope between 1999 and 2001 where the maximum temperatures experienced in winter and summer were relatively higher. This means that when considering the effect of both maximum temperature the locations that experience relatively warmer weather conditions around both winter and summer will be the ones most affected.



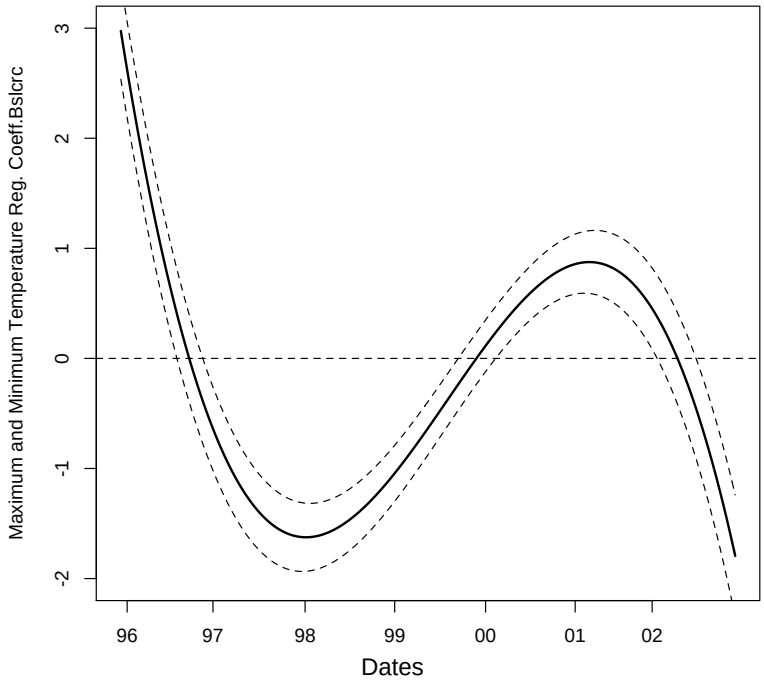
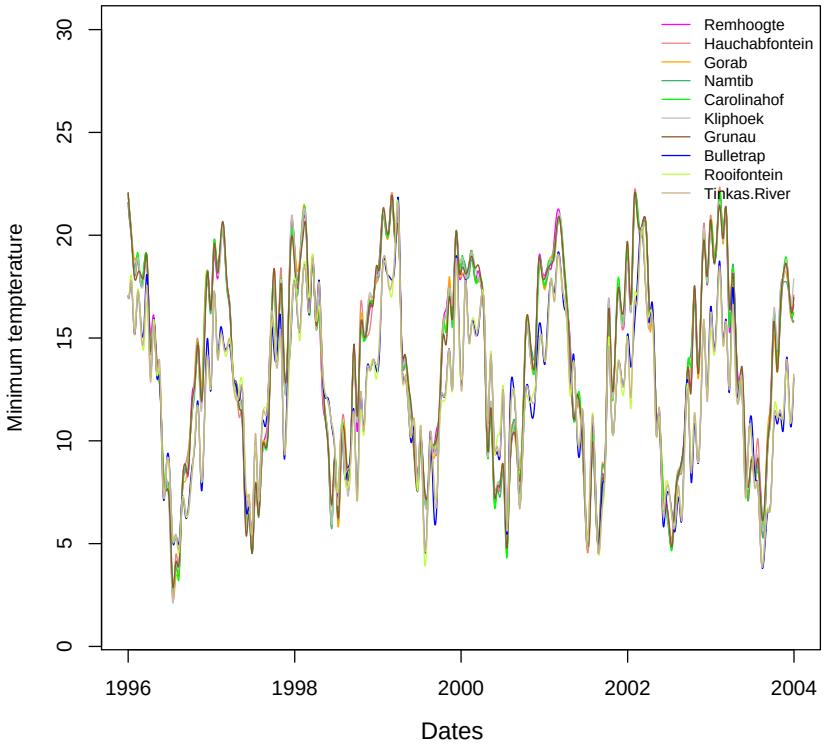


Figure 6.20: Maximum and Minimum Temperature : All Locations and Penalized Maximum and MinimumTemperature Beta Function for Basal Circumference with Confidence Intervals

Figure 6.21, represents the difference in the predicted value of basal circumference for each unit difference in minimum temperature if maximum temperature remains constant. This means that if minimum temperature differed by one unit and maximum temperature did not differ, the basal circumference will be most influenced when minimum temperature is relatively low in the winter months and higher in the summer months. This would translate to a slower growth during winter months when the minimum temperature is relatively and faster growth once it gets warmer in the summer months where higher minimum temperatures are experienced. This corresponds to a downward regression coefficient slope when temperatures are relatively low such as prior to 1998 in Figure 6.21 and an upward slope post 2001 when temperatures are relatively higher. The confidence intervals confirm a significant effect when temperature is much lower and when it is relatively higher(and increasing).



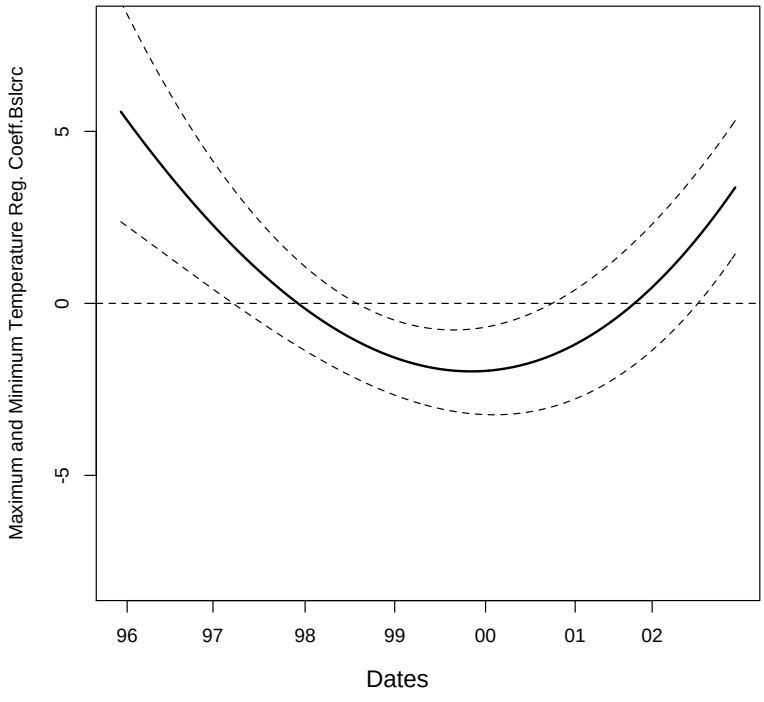
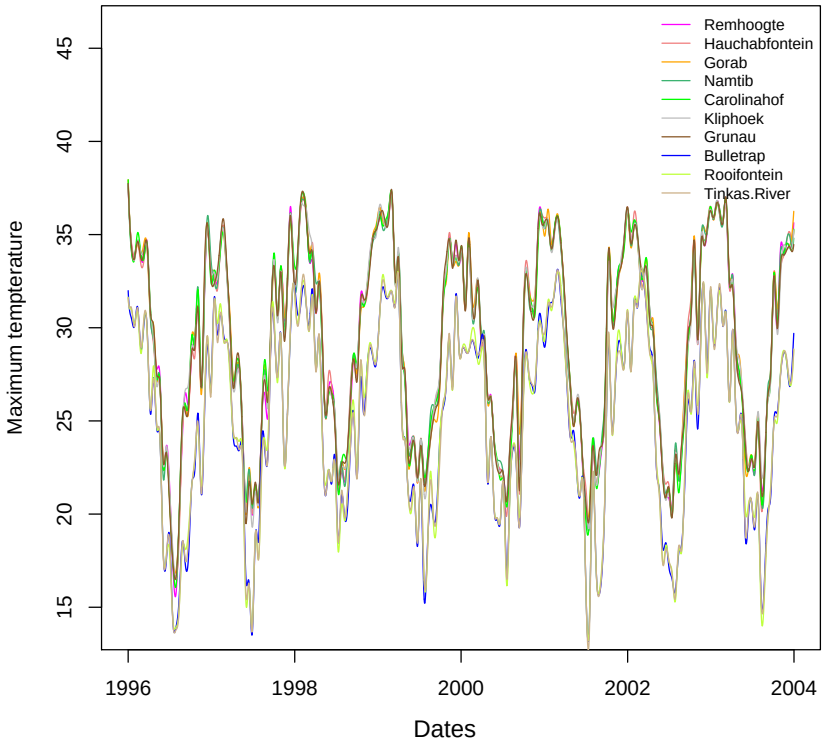


Figure 6.21: Maximum and Minimum Temperature : All Locations and Penalized Maximum and MinimumTemperature Beta Function for Basal Circumference with Confidence Intervals

6.5 Circumference at First Branch

In Figure 6.22,we see that the downward trend in the coefficient function between 1996 and mid 1997 agrees with lower than usual maximum winter temperatures in 1996 and 1997 and slightly lower than usual maximum summer temperatures in 1997, at least for some stations. The rising summer temperature and rising winter temperature values between 1998 and 1999 correspond to the gradual upward slope from 1998, leading to a steeper incline from 1999 to 2001. In the winter of 2001 there are exceptionally lower maximum temperature values which explain the sharp drop in the regression coefficient for circumference at first branch suggesting that much lower maximum temperature values have a detrimental effect on circumference at first branch, particularly for Bulletrap, Tinkas River and Rooifontein Another upward slope corresponds to the high maximum summer temperatures in 2002 and 2003 while lower maximum temperatures are observed for the first part of the 2004 summer, agreeing with the downward slope of the regression coefficient function towards 2004. It should be noted that the downward slope could be accentuated by fairly low maximum winter temperatures in Bulletrap, Rooifontein and Tinkas River and instability due to nearing the end of the time series.

In Figure 6.22, we see that a predictor for high basal circumference is a relatively high maximum temperature around the months of August and September. The 95% confidence intervals between 1996 and mid 1999 contain zero which suggests that the influence of maximum temperature on circumference at first branch in those periods is irrelevant. We see a peak between mid 1999 and mid 2001. This pattern suggests a contrast between late autumn and early summer with more emphasis on late autumn.This also suggests that the impact of maximum temperature observed between mid 1999 and 2001 is most likely the trend expected.The squared multiple correlation for this model is 0.99. The F-ratio is 51.03 with 7 and 2 degrees of freedom



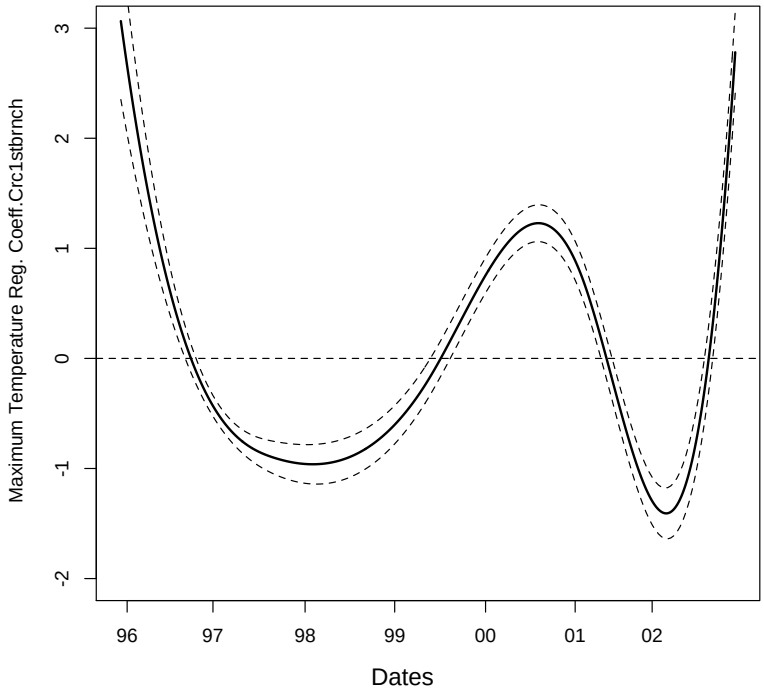


Figure 6.22: Maximum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Circumference at first Branch with Confidence Intervals

In Figure 6.23 it appears that there is no important relationship between minimum temperature and circumference at first branch as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.25 and a squared multiple correlation of 0.46.

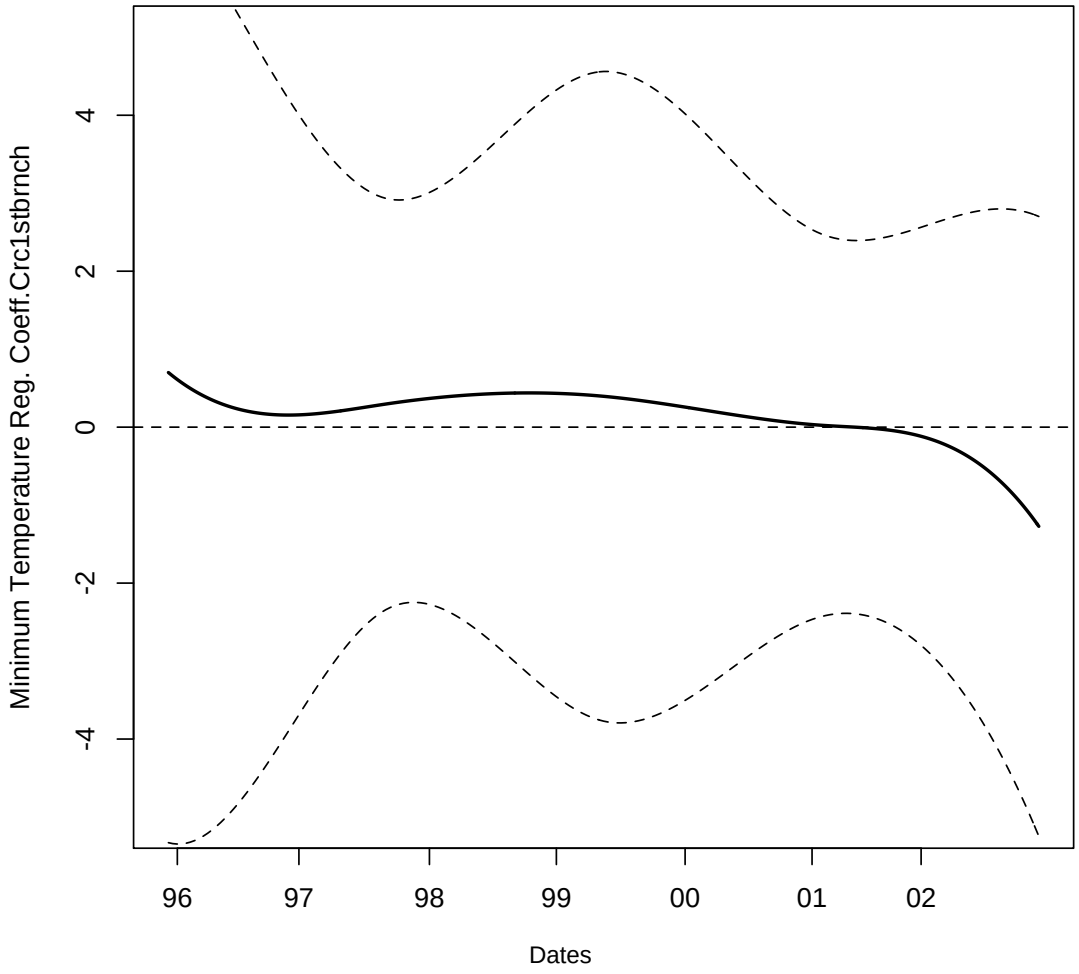


Figure 6.23: Penalized Minimum Temperature Beta Function with Confidence Intervals

In Figure 6.24 it appears that there is no important relationship between rainfall and

circumference at first branch as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.62 and a squared multiple correlation of 0.68.

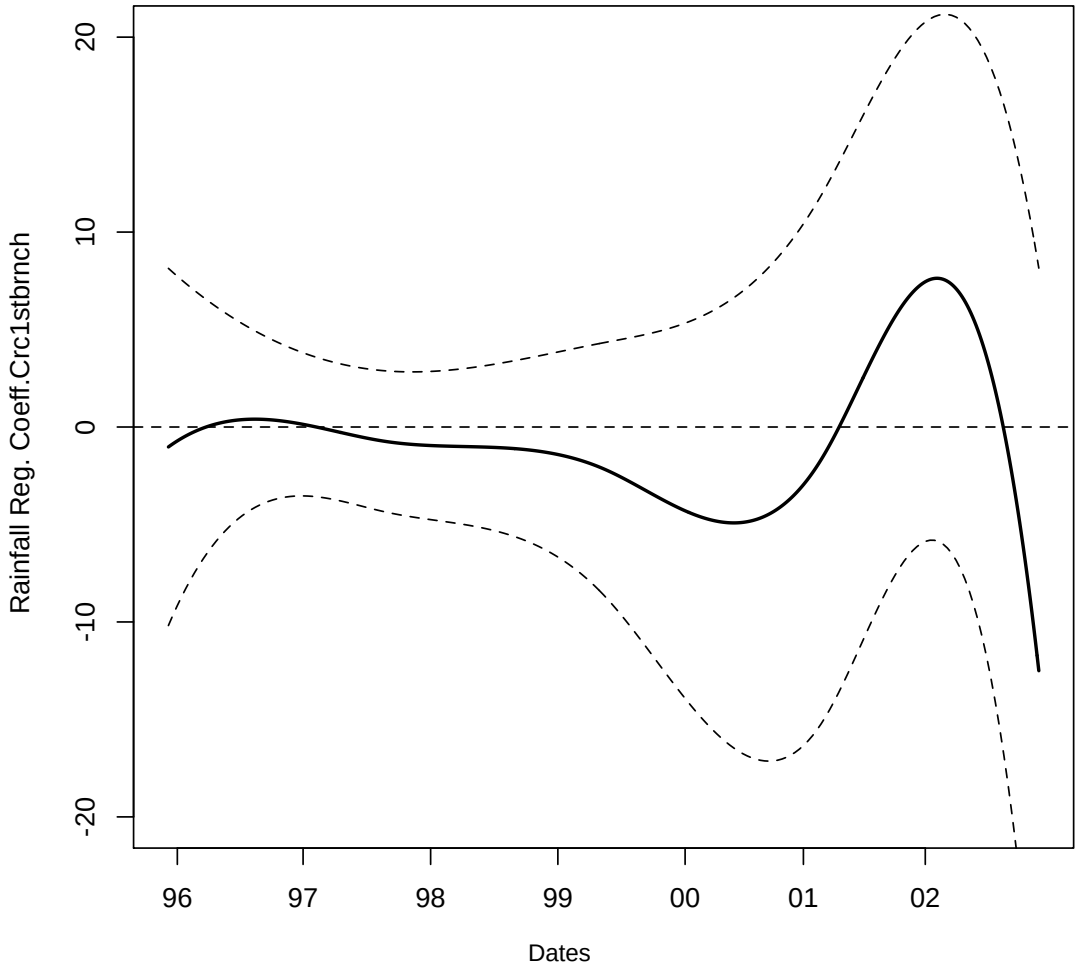
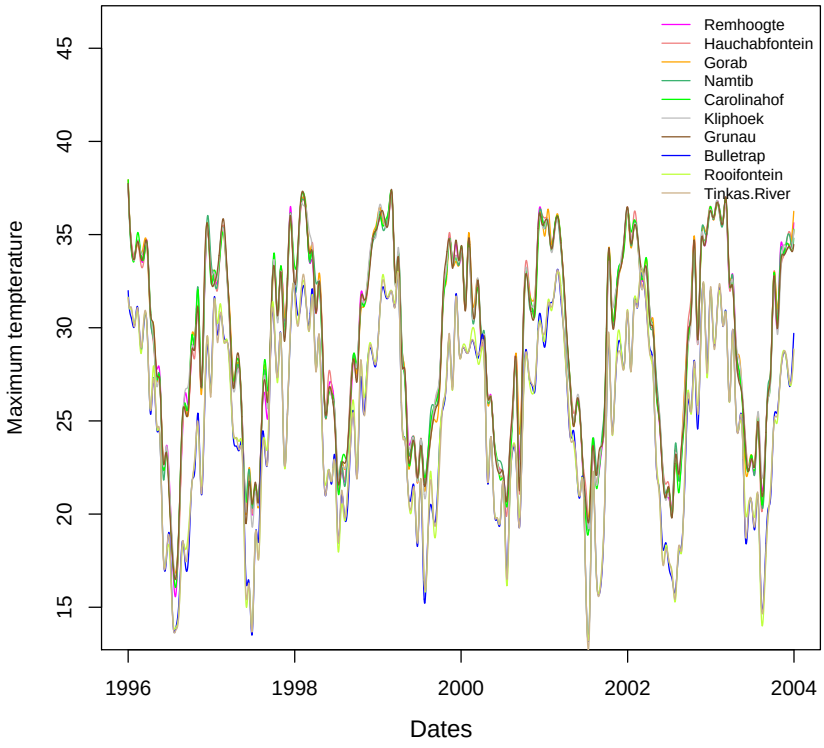


Figure 6.24: Penalized Rainfall Beta Function with Confidence Intervals

In Figure 6.25,represents the difference in the predicted value of circumference of the first branch for each unit difference in maximum temperature if minimum temperature remains

constant. This means that if maximum temperature differed by one unit and minimum temperature did not differ, the circumference of the first branch will be most influenced when maximum temperature is relatively high in summer and winter months. This corresponds with a upward slope between 1999 and 2001 where the maximum temperatures experienced in winter and summer were relatively higher. This means that when considering the effect of both maximum temperature the locations that experience relatively warmer weather conditions around both winter and summer will be the ones most affected.



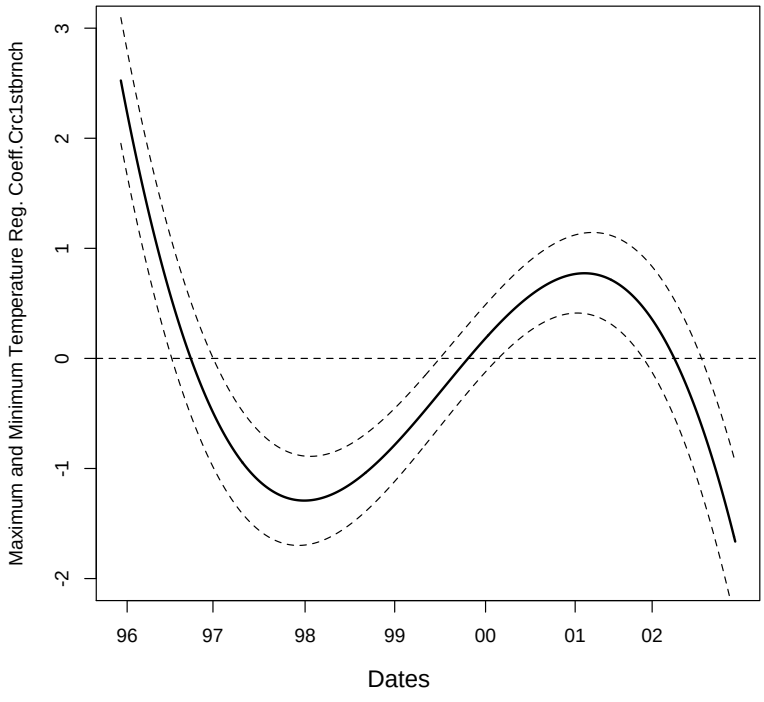


Figure 6.25: Maximum and Minimum Temperature : All Locations and Penalized Maximum and MinimumTemperature Beta Function for Circumference at First Branch with Confidence Intervals

In Figure 6.26 it appears that there is no important relationship between the circumference at first branch and temperature, when considering the effect of minimum temperature when maximum temperature is kept constant because the confidence intervals across the entire function contain zero.

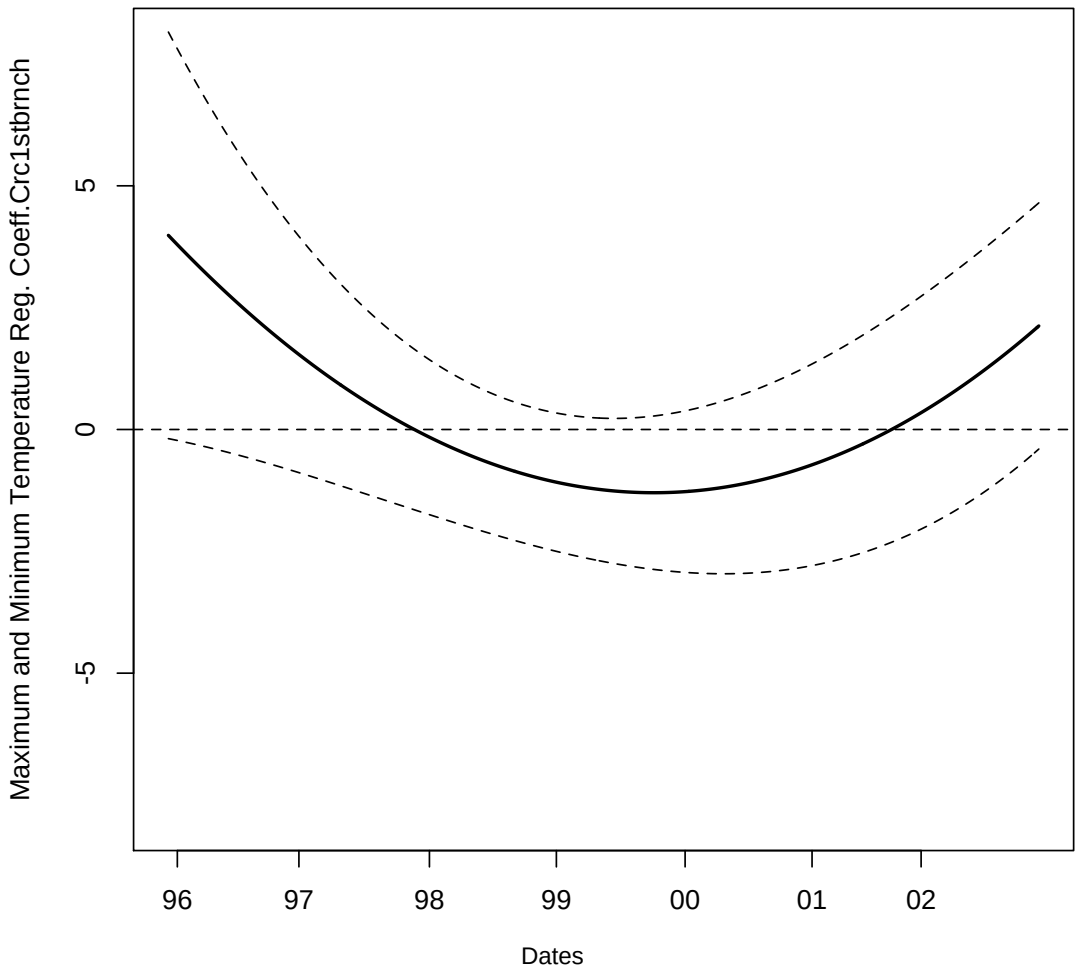


Figure 6.26: Maximum and Minimum Temperature Beta Function

6.6 Number of Branches Off Main Stem

In Figure 6.27 it appears that there is no important relationship between maximum temperature and number of branches off the main stem as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.82 and a squared multiple correlation of 0.74.

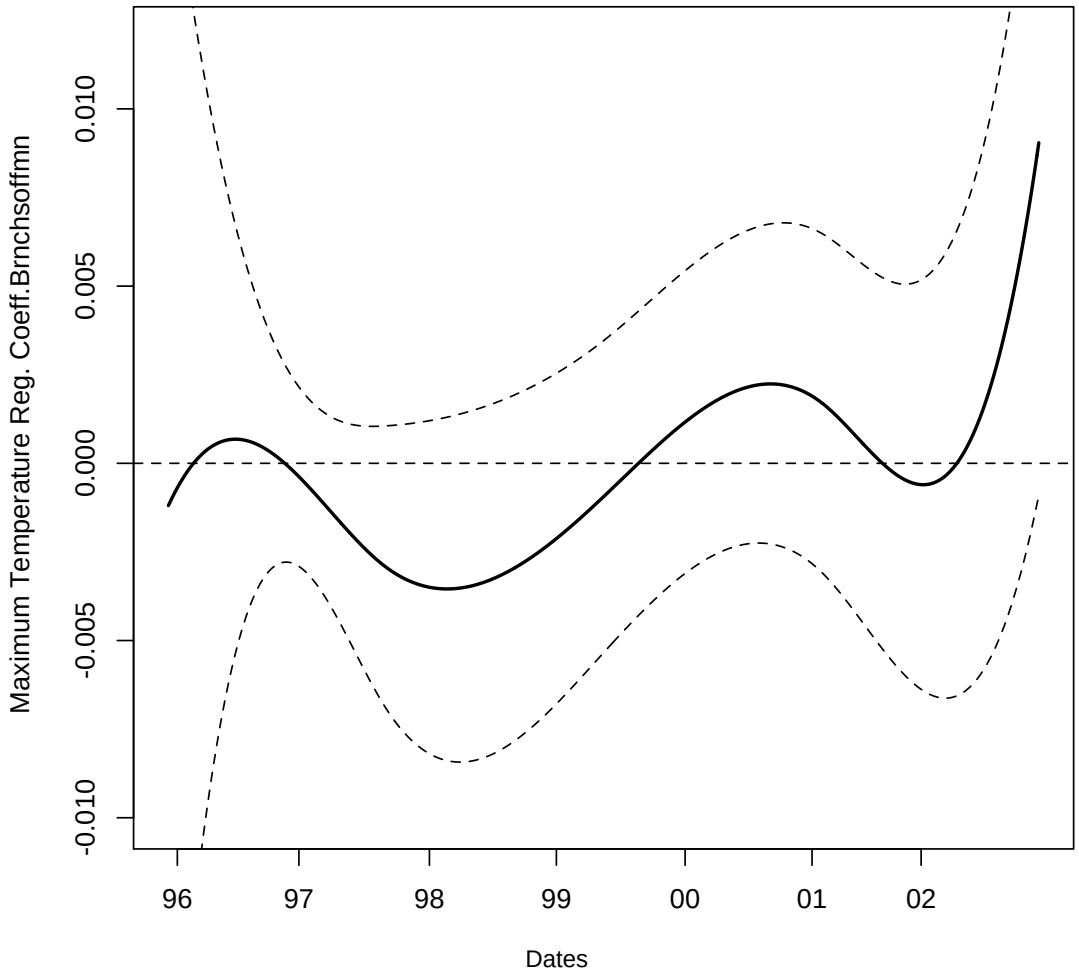


Figure 6.27: Penalized Maximum Temperature Beta Function with Confidence Intervals

In Figure 6.28 it appears that there is no important relationship between minimum temperature and number of branches off the main stem as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.57 and a squared multiple correlation of 0.66.

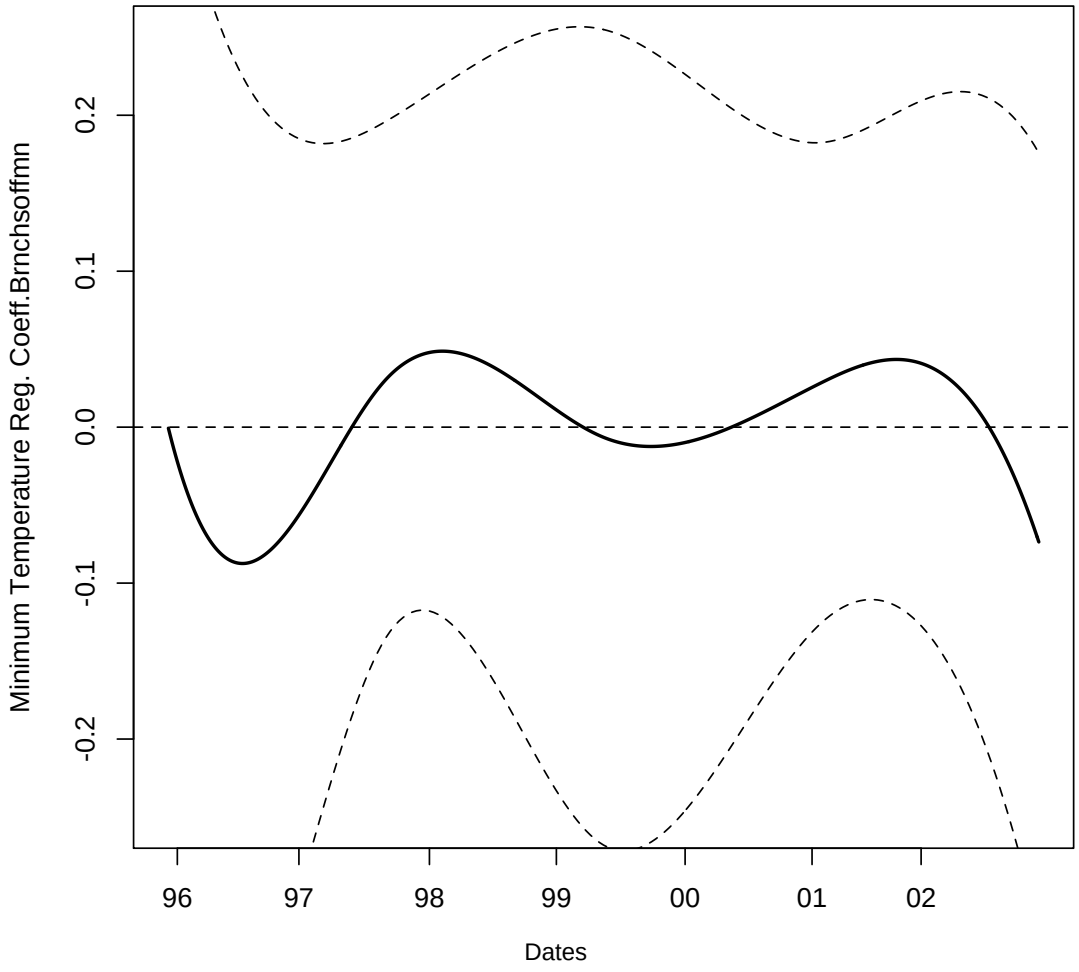


Figure 6.28: Penalized Minimum Temperature Beta Function with Confidence Intervals

In Figure 6.29 it appears that there is no important relationship between rainfall and

number of branches off the main stem as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 4.77 and a squared multiple correlation of 0.94.

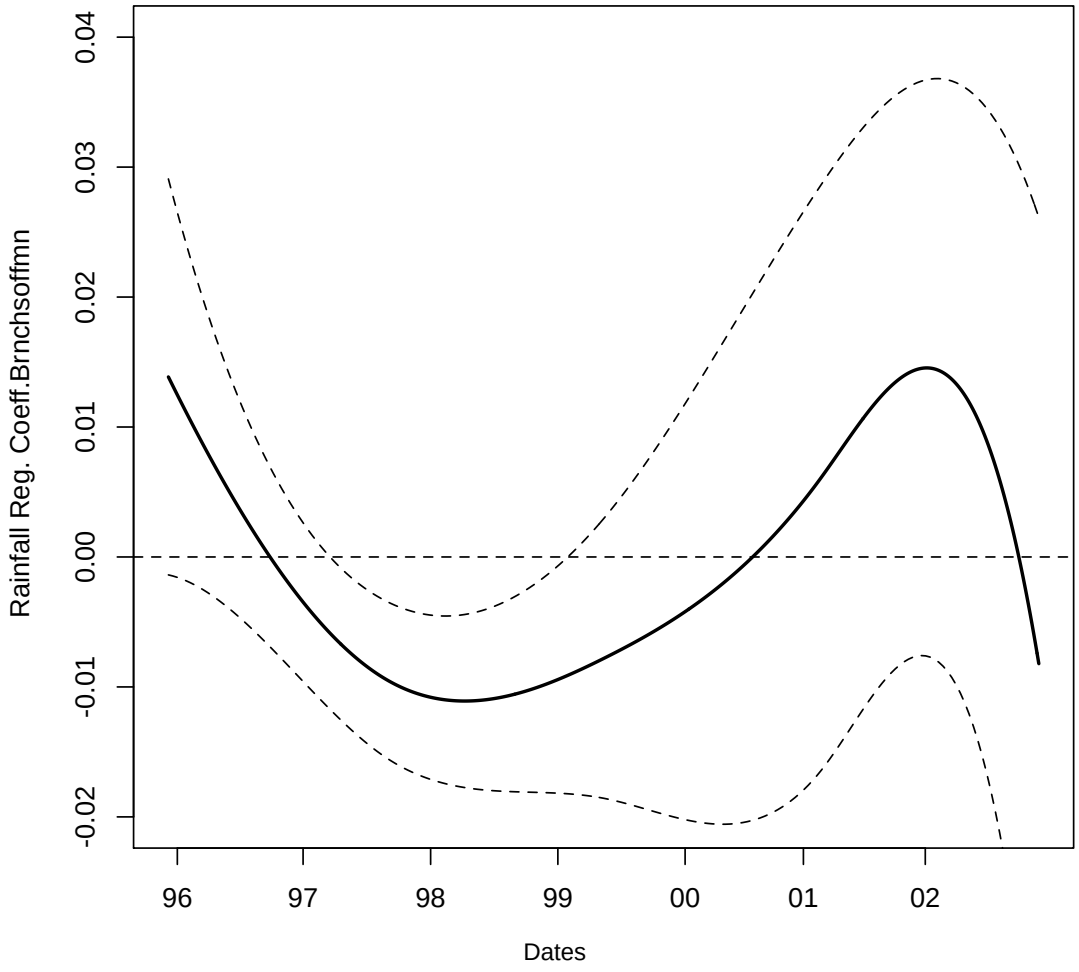
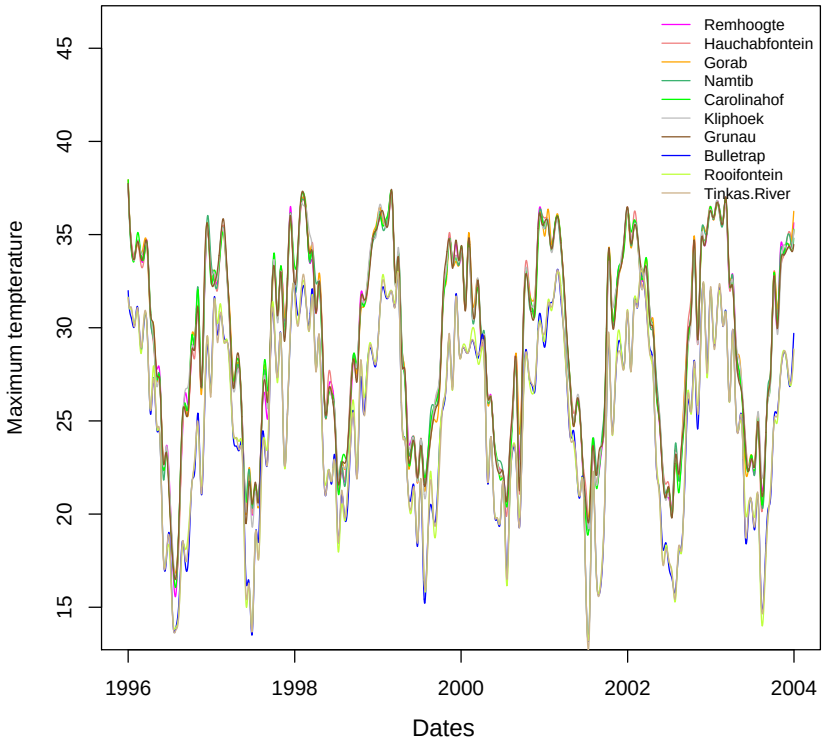


Figure 6.29: Penalized Rainfall Beta Function with Confidence Intervals

In Figure 6.30,represents the difference in the predicted value of the number of branches off the main stem for each unit difference in maximum temperature if minimum temper-

ature remains constant. This means that if maximum temperature differed by one unit and minimum temperature did not differ, the number of branches off the main stem will be most influenced when maximum temperature is relatively high in summer and winter months. This corresponds with a upward slope between 1999 and 2001 where the maximum temperatures experienced in winter and summer were relatively higher. This means that when considering the effect of both maximum temperature the locations that experience relatively warmer weather conditions around both winter and summer will be the ones most affected.



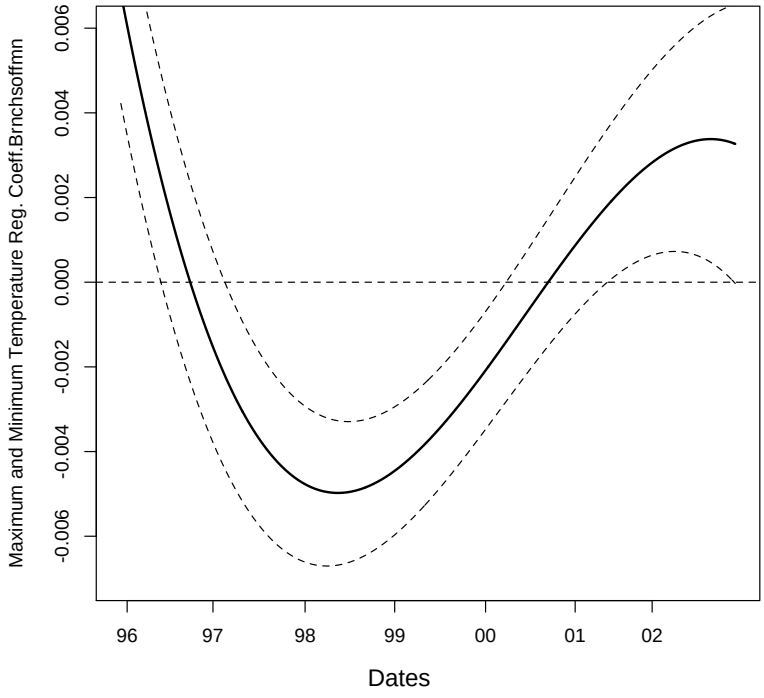


Figure 6.30: Maximum and Minimum Temperature : All Locations and Penalized Maximum and MinimumTemperature Beta Function for Number of branches off the Main Stem with Confidence Intervals

In Figure 6.31 it appears that there is no important relationship between the number of branches off the main stem and temperature, when considering the effect of minimum temperature when maximum temperature is kept constant because the confidence intervals across the entire function contain zero.

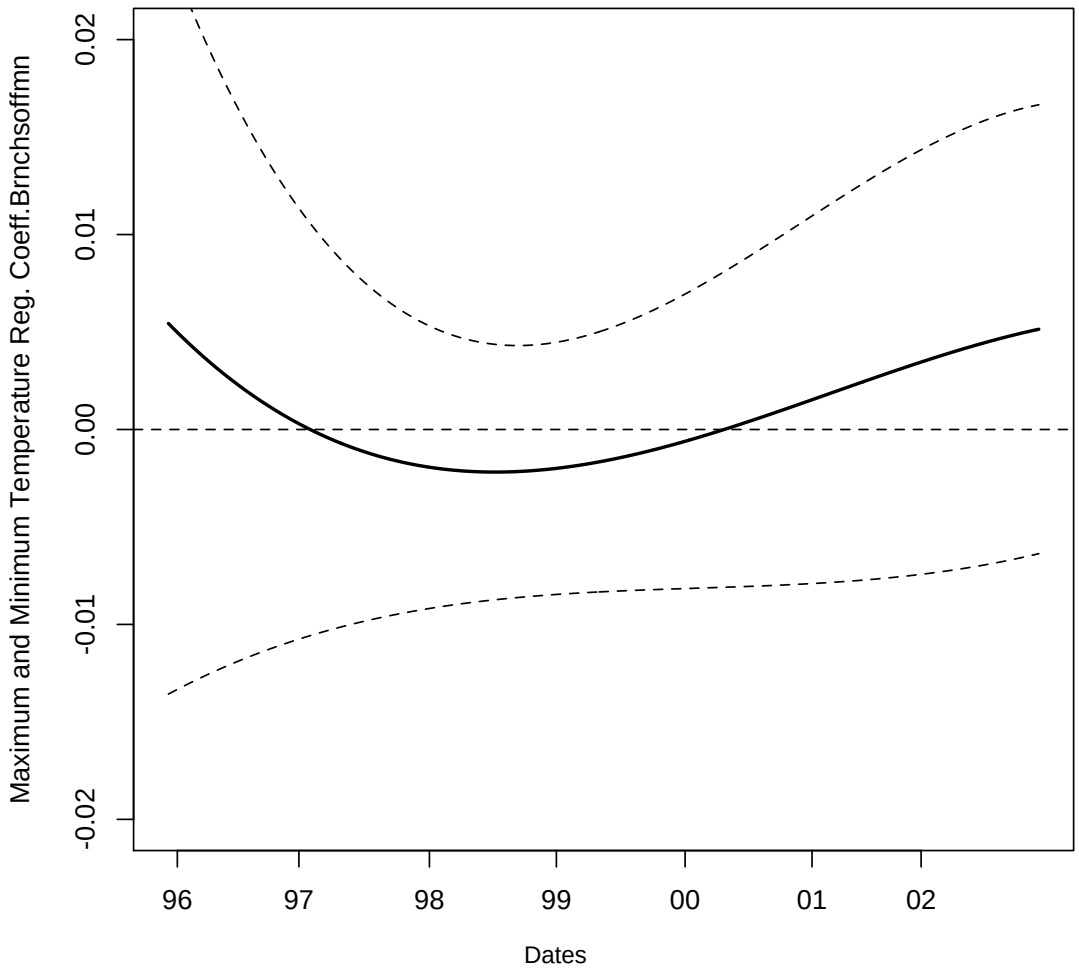


Figure 6.31: Maximum and Minimum Temperature Beta Function

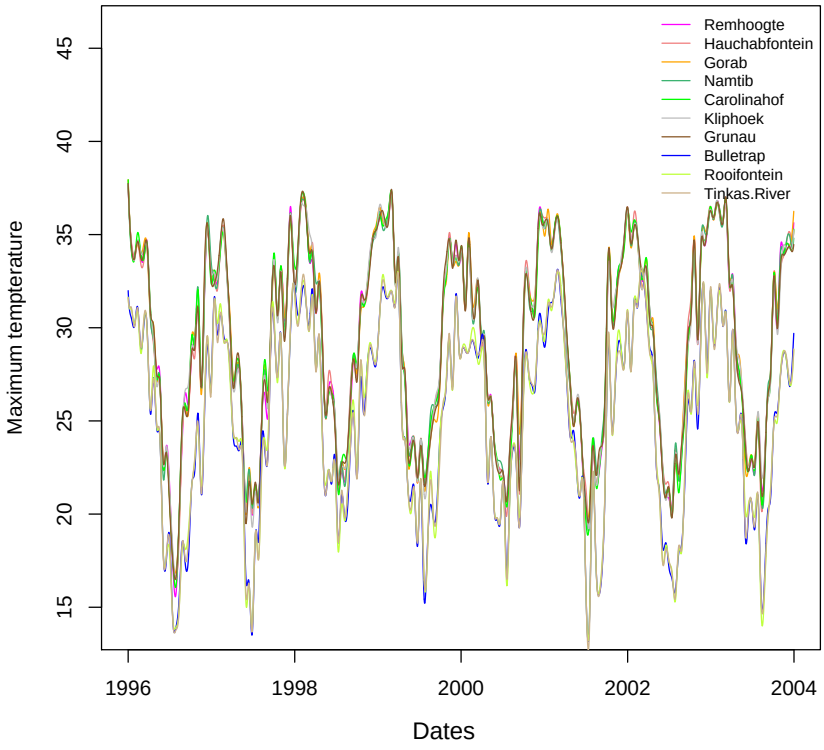
6.7 Number of Dichotomous Branches

In Figure 6.32, we see that the downward trend in the coefficient function between 1996 and 1998 agrees with lower than usual maximum winter temperatures in 1996 and 1997 and slightly lower than usual maximum summer temperatures in 1997, at least for some stations. Clearly the lower maximum temperatures had a detrimental effect on the number of dichotomous branches.

The effect of the maximum temperature on the number of dichotomous events is fairly immediate as high maximum temperatures where observed in the summers of 1998 and 1999 with exceptionally high maximum winter temperatures in 1998, and the regression coefficient function turns upward in 1998. Relatively high summer maximum temperatures correspond with the upward trend in the regression coefficient starting 1998. Lower maximum summer temperatures in the 2000 summer and exceptionally lower maximum winter temperatures in 2001, especially for Bulletrap, Rooifontein and Tinkas River corresponds with the second downwards slope of the regression coefficient function between 2000 and 2001. Another upward slope corresponds to the high maximum summer temperatures in 2002 and 2003. It should be noted that the downward slope could be accentuated by fairly low maximum winter temperatures in Bulletrap, Rooifontein and Tinkas River and instability due to nearing the end of the time series.

In Figure 6.32, we see that a predictor for a large number of dichotomous branches off the main stem is a relatively high maximum temperature around the months of August and September. The 95% confidence intervals in the earlier parts of the year around winter and summer contain zero which suggests that the influence of maximum temperature on canopy diameter in those periods is irrelevant. We see a peak in early spring and a valley towards early summer. This pattern suggests a contrast between spring and summer with more emphasis on spring. This pattern favours locations that are comparatively warm in late august/early spring and cool in summer.

The squared multiple correlation for this model is 0.94. The F-ratio is 4.20 with 7 and 2 degrees of freedom



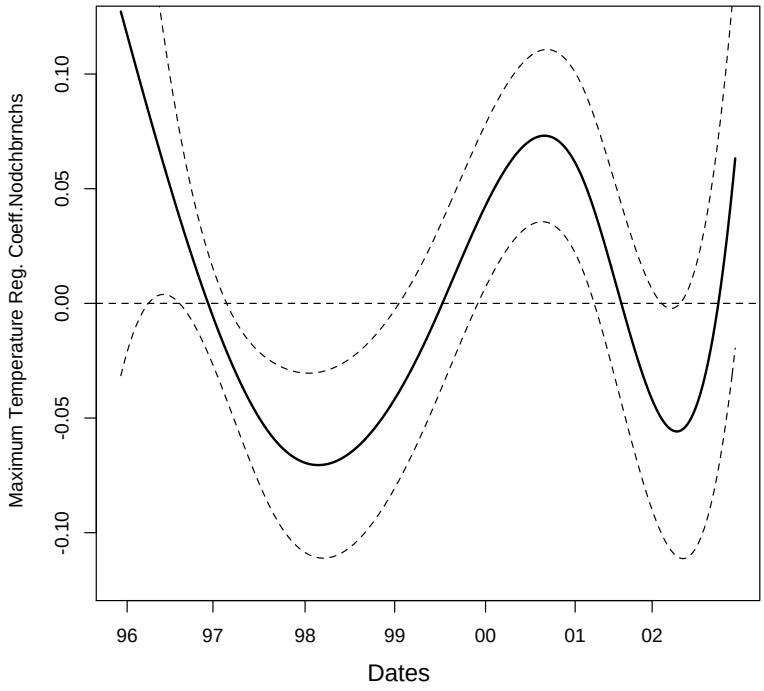


Figure 6.32: Maximum Temperature : All Locations and Penalized Maximum Temperature Beta Function for Number of Dichot Events with Confidence Intervals

In Figure 6.33 it appears that there is no important relationship between minimum temperature and number of dichotomous branches as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 0.30 and a squared multiple correlation of 0.51.

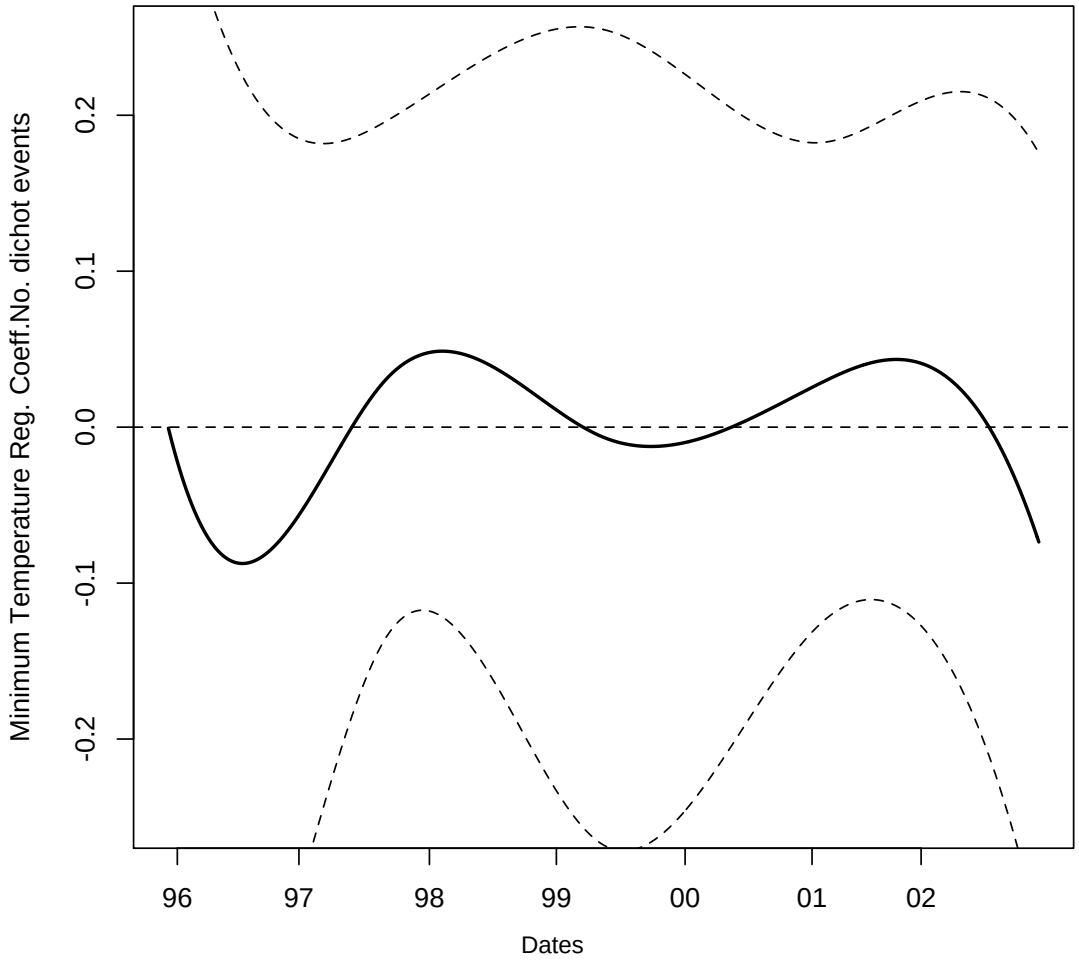
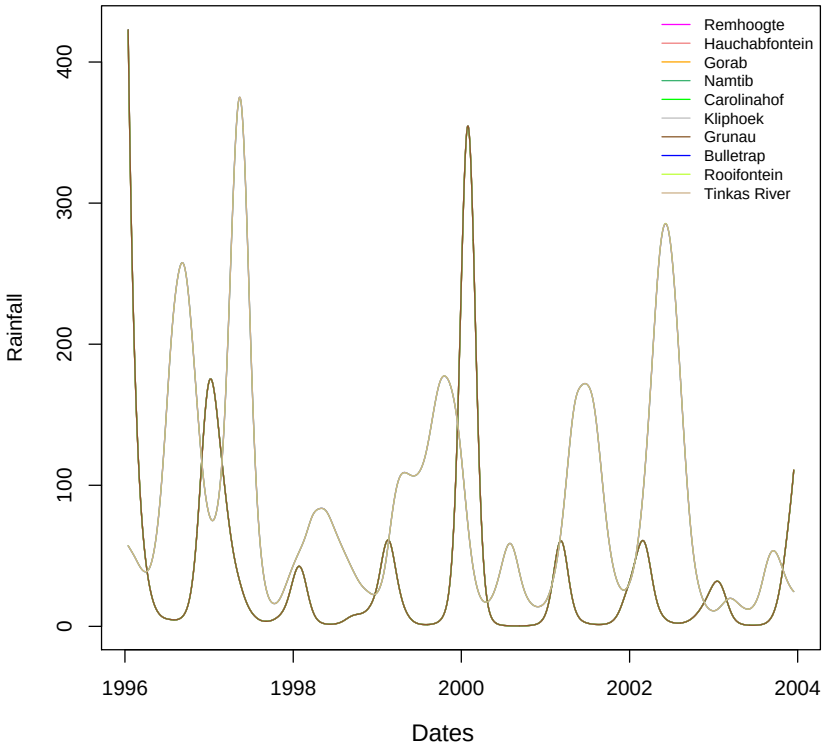


Figure 6.33: Penalized Minimum Temperature Beta Function with Confidence Intervals

Figure6.34 shows a slight upward slope of the regression coefficient function between 1996

and 1997 followed by a gradual downward slope between 1997 and 2001. This corresponds to relatively high levels of rainfall particularly in in Bulletrap, Tinkas River and Rooifontein. Another upward slope appears between 2001 and 2002 corresponding to relatively higher levels of rainfall particularly in the same regions. The sharp drop in the slope of the regression coefficient could be due to relatively low levels of rainfall, exaggerated by an end of series effect.

In Figure 6.34 it appears that there is no important relationship between rainfall and number of branches off the main stem as the confidence intervals across the entire function contain zero. The poor quality of the model is supported by an F-ratio of 1.29 and a squared multiple correlation of 0.80.



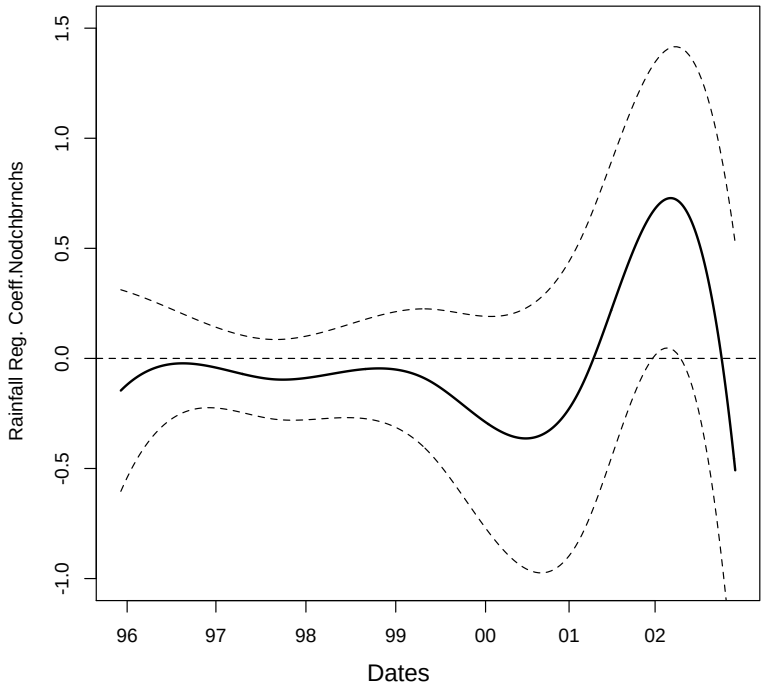
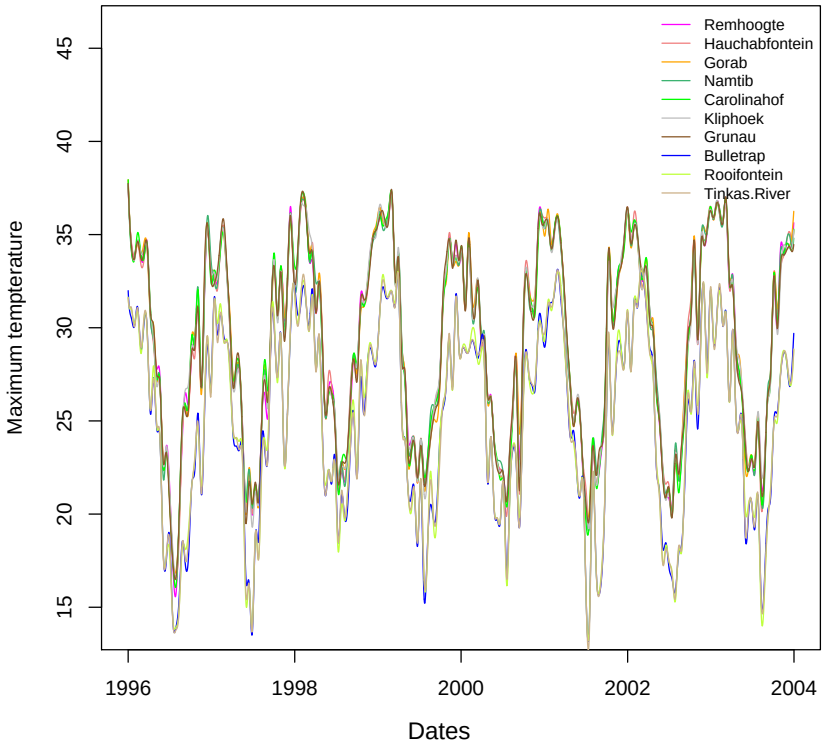


Figure 6.34: Rainfall: All Locations and Penalized Rainfall Beta Function for Number of Dichot Events with Confidence Intervals

In Figure 6.35,represents the difference in the predicted value of the number of dichotomous events for each unit difference in maximum temperature if minimum temperature remains constant. This means that if maximum temperature differed by one unit and minimum temperature did not differ, the number of dichotomous events will be most influenced when maximum temperature is relatively high in summer and winter months. This corresponds with a upward slope between 1999 and 2001 where the maximum temperatures experienced in winter and summer were relatively higher. This means that when considering the effect of both maximum temperature the locations that experience relatively warmer weather conditions around both winter and summer will be the ones most affected.



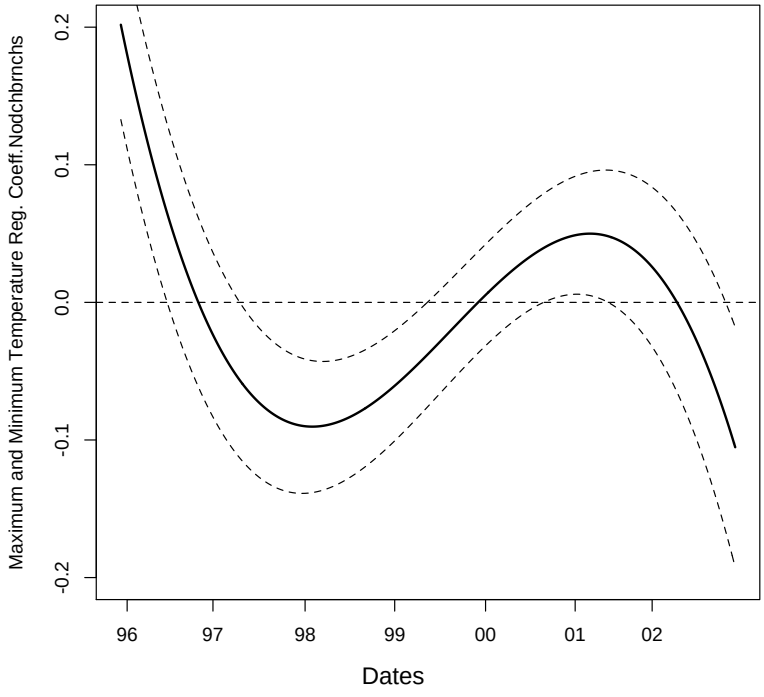


Figure 6.35: Maximum and Minimum Temperature : All Locations and Penalized Maximum and MinimumTemperature Beta Function for Number of Dichotomous Events with Confidence Intervals

In Figure 6.36 it appears that there is no important relationship between the number of dichotomous events and temperature, when considering the effect of minimum temperature when maximum temperature is kept constant because the confidence intervals across the entire function contain zero.

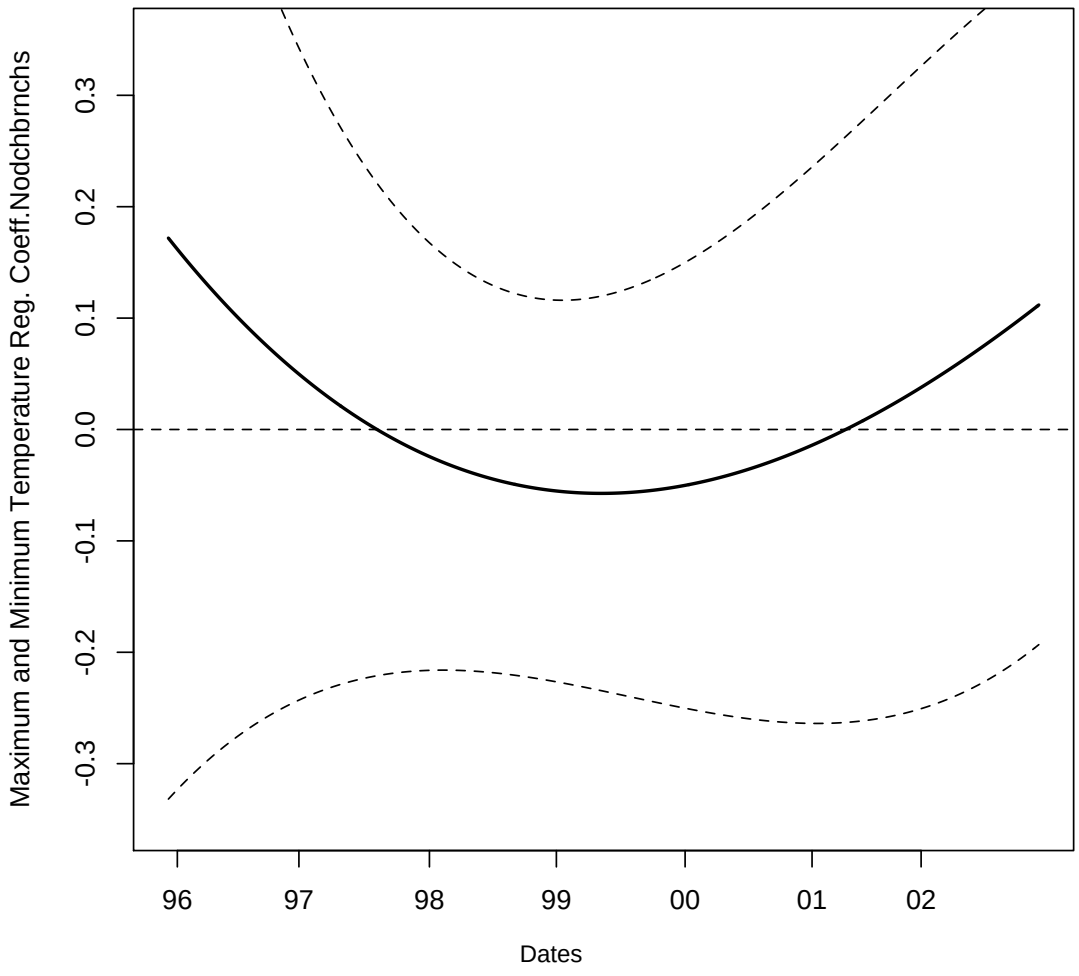


Figure 6.36: Maximum and Minimum Temperature Beta Function

It its quite clear in this section that maximum temperature has the biggest effect on the

structural form of the trees. Furthermore, the most affected structural variables were canopy diameter, total height, basal circumference and the number of dichotomous events.

Finally we present our concluding remarks in Chapter 7.

Chapter 7

Conclusion

The aim of this dissertation was to perform a more robust analysis around the hypotheses researchers had, regarding the structural form of the Quiver trees in geographically separated populations. Researchers believed that taller trunks with fewer branches characterize the Quiver tree populations found in the hot dry northern regions. Shorter trunks that are rounder for better seed production characterize the Quiver trees found in the Southern regions.

Previously structural and health variables were analyzed to investigate differences between 14 locations.First Fisher's linear discriminant analysis was used to identify which variables differed significantly between locations. Then multiple linear regression and Generalized additive modeling was used to model the effect of longitude, latitude, slope and aspect on each of these variables.Finally Partial least squares regression was used to relate the differences in structure and health to climate variables (Mzimela, 2013).

It was found that all seven structural variables needed to be included in the discrimination between locations. In general the hypotheses of the biologists were confirmed in this analysis, with taller trees in the north and shorter trees with fewer branches in the south. The longitude-latitude coordinates differed significantly for all seven variables. The slope only effects height at first branch and number of branches off the main stem, while aspect has a significant effect on total height, canopy diameter, basal circumference, circumference at first branch and number of dichotomous events (Mzimela, 2013).

Thus, the question we tried to answer is whether or not there are significant differences between Quiver trees in geographically separated locations using the more robust approach of functional data analysis.

Maximum temperature was the only climate variable that had a clear impact on the struc-

tural form of the trees. Between mid 1999 to mid 2000 relatively high values of maximum temperature in summer and winter corresponded to significant increases in canopy diameter, total height, basal circumference, circumference at first branch and the number of dichotomous branches, which suggests that these structural variables favour relatively warmer conditions. This means that canopy diameter, total height, basal circumference, circumference at first branch and the number of dichotomous events will be larger for trees in warmer locations. Furthermore we saw that the locations that were impacted the most were Bulletrap, Tinkas River and Rooifontein. These locations are the locations with the lowest elevation and those closest to the coast line.

The lack of an effect of rainfall on the Quiver tree structure might be due to the limited data that was available. In this analysis no clear effect of rainfall could be identified, but complete rainfall data over a longer period of time, including severe droughts and exceptionally wet seasons might still show an effect on tree structure.

The results suggest that Quiver trees in warmer locations are more likely to be taller and possibly have more branches if it can be assumed that the larger number of dichotomous events implies this. The results further suggest that maximum temperature is the climate variable with the most effect on the structural form of the trees. We conclude that the researchers finds were supported by our analysis.

Chapter 8

References

Araki, Y., Konishi, S., Kawano, S. and Matsui, H. (2009). Functional Regression Modeling via Regularized Gaussian Basis Expansions. Annals of the Institute of Statistical Mathematics 61.4 811-833

Attenborough. D. (1995). The Private Life of Plants: A Natural History of Plant Behavior. BBC Books: London.

Cardot, H., Ferraty, F. and Sarda, P. (1999). Functional linear model, Statistics and Probability Letters. pp.11:22.

Cardot, H., Ferraty, F. and Sarda, P. (2003). Spline Estimators for The Functional Linear Model. Statistica Sinica, pp.571-591.

GHCND.National Centers for Environmental Information, (2016). Climate Data Online. [online] Available at: http://www.ncdc.noaa.gov/cdo-web/search [Accessed 26 Jun. 2015].

Gower. J.C., Lubbe. S. and Le Roux. N.J. (2011). Understanding Biplots. Wiley: Chichester.

Gower and Hand (1996). Biplots. Chapman and Hall/CRC Press: London.

Haitovsky,Y.(1969).Multicollinearity in Regression Analysis: Comment.The Review of Economics and Statistics.Vol. 51, pp. 486-489. Online

Hastie, T. and Mallows, C. (1993). A discussion of "A Statistical View of Some Chemometrics Regression Tools" by I.E Frank and J.H Friedman. Technometrics 35, pp.140:143.

Hastie. T.J and Tibshirani. R.J (1990). Generalized Additive Models. Chapman and Hall: New York.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). The elements of statistical learning. New York: Springer.

IUCN.(2009). Climate Change and Specie Report: Quiver Trees and Climate Change. pp 20-23. https://www.cmsdata.iucn.org/downloads/fact sheet red list quivertree v2.pdf

Jack. S (2012). Personal communication.

James, G. M. (2002). Generalized linear models with functional predictor variables. J. Roy. Statist. Soc. Ser. B 64, pp.411:432.

Konishi, S. and Kitagawa, G. (2007). Information criteria and statistical modeling. New York: Springer.

Marx, B. and Eilers, P. (1996). Generalized linear regression on sampled signals with penalized likelihood. International Workshop on Statistical Modeling.

Marx, B. and Eilers, P. (1999). Generalized linear regression on sampled signals and curves: a P-spline approach. Technometrics 41, pp.1:13.

Muller, H. G. and Stadtmuller, U. (2005). Generalized functional linear models. Ann. Statist. 33, pp.774:805.

Mzimela, N. (2013). Differences in the structural form and health of Aloe Dichotoma in relation to climate and distribution. Unpublished BComHons project, Department of Statistical Sciences, UCT.

Ramsay, J. O. and Dalzell, C. J. (1991). Some tools for functional data analysis. Journal of the Royal Statistical Society: Series B 53 pp.539:572.

Ramsay, J. and Silverman, B. (1997). Functional data analysis. New York: Springer.

Ramsay, J. and Silverman, B. (2002). Applied functional data analysis. New York: Springer.

Ramsay, J., Hooker, G. and Graves, S. (2009). Functional data analysis with R and MATLAB. New York: Springer.

Chapter 9

Appendix

```
\begin{lstlisting}[breaklines]

library(MASS)
data<-read.csv("~/Documents/semester 2 hons/honours project/datasets/master data.csv",heade
dt<-read.csv("~/Documents/Possible data set- Namibia & SA.csv")


struc.vars<-cbind(data[,1:2],data[,13:19])
health.vars<-cbind(data[,1:2],data[,20:25],data[,33:35])

require(rgl)
require(RColorBrewer)
library(RgoogleMaps)

da.func<-function(xx)
{

#  print(ncol(xx))
  if (nrow(xx)==0)
    { return(-99)}
  i<-2

  while(i<=ncol(xx))
  {
    out<-table(xx[,i])
    if (dim(out)==1)
```

```
      return(-99)
    i<-i+1
  }

  {
    colnames(xx)<-c("Location",paste("x",1:(ncol(xx)-1),sep=""))
    my.text<-"out<-lda(Location~."

    for (i in 2:ncol(xx))
      my.text<-paste(my.text,colnames(xx)[i],sep="+")
    my.text<-paste(my.text,",data=xx,CV=T)")


    eval(parse(text=my.text))
    err<-sum(out$class!=xx$Location)/nrow(xx)

    return(err)


  }
}


combin <- function (r, n, vek = 1:n)
{
  if (r <= 0)
    NULL
  else if (r >= n)
    vek[1:n]
  else rbind(cbind(vek[1], Recall(r - 1, n - 1, vek[-1])),
             Recall(r, n - 1, vek[-1]))
}


err.vec <- vector("list",11)
best.combin<-vector("list",11)
for (r in 1:11)
{ mat<- combin(r,11)
  err.vec[[r]]<-rep(NA,nrow(mat))
  for (i in 1:nrow(mat))
  {
```

```
    flush.console()
    print (c(r,i))
    currentdata <- health.vars[,c(2,mat[i,]+2)]
    currentdata <- na.omit(currentdata)
    err.vec[[r]][i]<-da.func(currentdata)
  }
  err.vec[[r]][err.vec[[r]]==-99] <- NA
  min.err<-order(err.vec[[r]])[1]
  min.mat <- mat[err.vec[[r]]==err.vec[[r]][min.err],]
  best.combin[[r]]<-na.omit(min.mat)
}
print(best.combin)
subset.func(struc.vars)
err.vec <- subset.func(struc.vars)

min.err.rate.per.num.vars <- sapply(err.vec,min,na.rm=T)

min.err.rate.per.num.vars

levels(factor(paste(data$Location,data$Longitude,data$Latitude)))


#_________________confusion matrix_______________________________

#struc variables
struc.lda=lda(Location~.,data=struc.vars,CV=T)
my.data=na.omit(cbind(Location=data$Location,struc.vars))

my.data$Location=relevel(my.data$Location,"Gannabos")
my.data$Location=relevel(my.data$Location,"Rooifontein")
my.data$Location=relevel(my.data$Location,"Bulletrap")
my.data$Location=relevel(my.data$Location,"Grunau")
my.data$Location=relevel(my.data$Location,"Kliphoek")
my.data$Location=relevel(my.data$Location,"Carolinahof")
my.data$Location=relevel(my.data$Location,"Namtib")
my.data$Location=relevel(my.data$Location,"Gorab")
my.data$Location=relevel(my.data$Location,"Hauchabfontein")
my.data$Location=relevel(my.data$Location,"Remhoogte")
my.data$Location=relevel(my.data$Location,"Tinkas River")
my.data$Location=relevel(my.data$Location,"Spitzkoppe")
my.data$Location=relevel(my.data$Location,"Omaruru River")
```

```
my.data$Location=relevel(my.data$Location,"Brandberg")

class1=struc.lda$class
class1=relevel(class1,"Gannabos")
class1=relevel(class1,"Rooifontein")
class1=relevel(class1,"Bulletrap")
class1=relevel(class1,"Grunau")
class1=relevel(class1,"Kliphoek")
class1=relevel(class1,"Carolinahof")
class1=relevel(class1,"Namtib")
class1=relevel(class1,"Gorab")
class1=relevel(class1,"Hauchabfontein")
class1=relevel(class1,"Remhoogte")
class1=relevel(class1,"Tinkas River")
class1=relevel(class1,"Spitzkoppe")
class1=relevel(class1,"Omaruru River")
class1=relevel(class1,"Brandberg")
table(class1,my.data$Location)

#health Variables
data.lda=lda(Location~.,data=data,CV=T)
my.data=na.omit(cbind(Location=data$Location,data))

my.data$Location=relevel(my.data$Location,"Gannabos")
my.data$Location=relevel(my.data$Location,"Rooifontein")
my.data$Location=relevel(my.data$Location,"Bulletrap")
my.data$Location=relevel(my.data$Location,"Grunau")
my.data$Location=relevel(my.data$Location,"Kliphoek")
my.data$Location=relevel(my.data$Location,"Carolinahof")
my.data$Location=relevel(my.data$Location,"Namtib")
my.data$Location=relevel(my.data$Location,"Gorab")
my.data$Location=relevel(my.data$Location,"Hauchabfontein")
my.data$Location=relevel(my.data$Location,"Remhoogte")
my.data$Location=relevel(my.data$Location,"Tinkas River")
my.data$Location=relevel(my.data$Location,"Spitzkoppe")
my.data$Location=relevel(my.data$Location,"Omaruru River")
my.data$Location=relevel(my.data$Location,"Brandberg")

class1=data.lda$class
class1=relevel(class1,"Gannabos")
class1=relevel(class1,"Rooifontein")
```

```
class1=relevel(class1,"Bulletrap")
class1=relevel(class1,"Grunau")
class1=relevel(class1,"Kliphoek")
class1=relevel(class1,"Carolinahof")
class1=relevel(class1,"Namtib")
class1=relevel(class1,"Gorab")
class1=relevel(class1,"Hauchabfontein")
class1=relevel(class1,"Remhoogte")
class1=relevel(class1,"Tinkas River")
class1=relevel(class1,"Spitzkoppe")
class1=relevel(class1,"Omaruru River")
class1=relevel(class1,"Brandberg")
table(class1,my.data$Location)


#_____________________________descriptive analysis_____________________________________
levels(struc.vars$Location)
struc.vars$Location=relevel(struc.vars$Location,"Gannabos")
struc.vars$Location=relevel(struc.vars$Location,"Rooifontein")
struc.vars$Location=relevel(struc.vars$Location,"Bulletrap")
struc.vars$Location=relevel(struc.vars$Location,"Grunau")
struc.vars$Location=relevel(struc.vars$Location,"Kliphoek")
struc.vars$Location=relevel(struc.vars$Location,"Carolinahof")
struc.vars$Location=relevel(struc.vars$Location,"Namtib")
struc.vars$Location=relevel(struc.vars$Location,"Gorab")
struc.vars$Location=relevel(struc.vars$Location,"Hauchabfontein")
struc.vars$Location=relevel(struc.vars$Location,"Remhoogte")
struc.vars$Location=relevel(struc.vars$Location,"Tinkas River")
struc.vars$Location=relevel(struc.vars$Location,"Spitzkoppe")
struc.vars$Location=relevel(struc.vars$Location,"Omaruru River")
struc.vars$Location=relevel(struc.vars$Location,"Brandberg")

boxplot(struc.vars$TotHt~struc.vars$Location, main= "Total Height",las=2,cex.axis=0.7)
boxplot(struc.vars$Ht1stBrnch~struc.vars$Location, main= "Height at first branch",las=2,cex
boxplot(struc.vars$CnpyDm~struc.vars$Location, main= "Canopy diameter",las=2,cex.axis=0.7)
boxplot(struc.vars$BslCrc~struc.vars$Location, main= "Basal Circumference",las=2,cex.axis=0
boxplot(struc.vars$Crc1stBrnch~struc.vars$Location, main= "Circumference at first branch",l
table(struc.vars$NoDchBrnchs,group=struc.vars$Location)
barplot(table(struc.vars$BrnchsOffMn,group=struc.vars$Location),las=2,cex.axis=0.7,col=my.c
barplot(table(struc.vars$NoDchBrnchs,group=struc.vars$Location),las=2,cex.axis=0.7,col=my.c
boxplot(struc.vars$NoDchBrnchs~struc.vars$Location, main= "No dichotomous Branches",las=2,c
```

```
legend("topleft",legend=c(1,2,3,4,5),fill=my.col,bg=NULL,bty="n",cex=0.7)
legend("topright",legend=c(7,8,9,10,11,12,13,14,15),fill=my.col,bg=NULL,bty="n",cex=0.7)

health.vars$Location=relevel(health.vars$Location,"Gannabos")
health.vars$Location=relevel(health.vars$Location,"Rooifontein")
health.vars$Location=relevel(health.vars$Location,"Bulletrap")
health.vars$Location=relevel(health.vars$Location,"Grunau")
health.vars$Location=relevel(health.vars$Location,"Kliphoek")
health.vars$Location=relevel(health.vars$Location,"Carolinahof")
health.vars$Location=relevel(health.vars$Location,"Namtib")
health.vars$Location=relevel(health.vars$Location,"Gorab")
health.vars$Location=relevel(health.vars$Location,"Hauchabfontein")
health.vars$Location=relevel(health.vars$Location,"Remhoogte")
health.vars$Location=relevel(health.vars$Location,"Tinkas River")
health.vars$Location=relevel(health.vars$Location,"Spitzkoppe")
health.vars$Location=relevel(health.vars$Location,"Omaruru River")
health.vars$Location=relevel(health.vars$Location,"Brandberg")



boxplot(health.vars$NoLvHds~health.vars$Location, main= "Number of live heads ",las=2,cex.a
boxplot(health.vars$NoDdHds~health.vars$Location, main= "Number of dead heads",las=2,cex.ax
boxplot(health.vars$BrknHds~health.vars$Location, main= "Number of broken heads",las=2,cex.
boxplot(health.vars$PrpDdHds~health.vars$Location, main= "Proportion of dead heads",las=2,c
boxplot(health.vars$PrpBrknHds~health.vars$Location, main= "Proportion of broken heads",las
barplot(table(health.vars$StmCnd,group=health.vars$Location),las=2,cex.axis=0.7,col=c("blue
legend("topleft",legend=c(1,2,3),fill=c("blue","green","white"),bg=NULL,bty="n")
barplot(table(health.vars$LfCnd,group=health.vars$Location),las=2,cex.axis=0.7,col=c("blue"
legend("")


chisq.test(table(health.vars$StmCnd,group=health.vars$Location))
chisq.test(table(health.vars$LfCnd,group=health.vars$Location))
my.table2=table(group=health.vars$Location,health.vars$LfCnd)
my.table=table(group=health.vars$Location,health.vars$StmCnd)
chisq.test(my.table[,1:2])
my.table2[,2]=my.table2[,2]+my.table2[,3]
my.table2=my.table2[,-3]
chisq.test(my.table2)
```

```
table(group=health.vars$Location,health.vars$LfCnd)
#*****************************descriptive statistics*************************************
library(psych)
describeBy(struc.vars$TotHt,group=struc.vars$Location)
describeBy(struc.vars$Ht1stBrnch,group=struc.vars$Location)
describeBy(struc.vars$CnpyDm,group=struc.vars$Location)
describeBy(struc.vars$BslCrc,group=struc.vars$Location)
describeBy(struc.vars$Crc1stBrnch,group=struc.vars$Location)
describeBy(struc.vars$BrnchsOffMn,group=struc.vars$Location)
describeBy(struc.vars$NoDchBrnchs,group=struc.vars$Location)

describeBy(health.vars$NoLvHds,group=health.vars$Location)
describeBy(health.vars$NoDdHds,group=health.vars$Location)
describeBy(health.vars$BrknHds,group=health.vars$Location)
describeBy(health.vars$PrpDdHds,group=health.vars$Location)
describeBy(health.vars$PrpBrknHds,group=health.vars$Location)
describeBy(health.vars$StmCnd,group=health.vars$Location)
describeBy(health.vars$LfCnd,group=health.vars$Location)


#_____________________________________surface area drawings_____________________
tree.diag<-function(x,y,kk,gg,tt,bb,hh)
{
  polygon(x=c(x,x+gg,x+gg-((gg-kk)/2),x+(gg-kk)/2),y=c(y,y,y+tt,y+tt),col="brown")
  lines(x=c(x+gg/2+kk/2,x+gg/2+bb/2),y=c(y+tt,y+tt+hh-bb/2),col="dark green")
  lines(x=c(x+gg/2-kk/2,x+gg/2-bb/2),y=c(y+tt,y+tt+hh-bb/2),col="dark green")
  theta<-seq(0,pi,length=100)

  x.vec<-(bb/2*cos(theta))
  y.vec<-(bb/2*sin(theta))
  lines(x+gg/2+x.vec,y+tt+hh-bb/2+y.vec,col="dark green")

  theta.1<-seq(0,pi/3,length=100)

  x.vec.1<-(bb/2*cos(theta.1))
  y.vec.1<-(bb/2*sin(theta.1))
```

```
  theta.2<-seq(0,pi/3,length=100)

  x.vec.2<-(-bb/2*cos(theta.2))
  y.vec.2<-(-bb/2*sin(theta.2))
  #polygon(x=c(x+gg/2+x.vec.1,x+gg/2,x+gg/2+bb/2),y=c(y+tt+hh/2+y.vec.1,y+tt+hh-bb/2,y+tt+h

  #polygon(x=c(x+gg/2+x.vec.2,x+gg/2,x+gg/2+bb/2),y=c(y+tt+hh/2-y.vec.2,y+tt+hh-bb/2,y+tt+h


  lines(x+gg/2,y+tt+hh-bb/2,col="red")
  if(!is.na())
  {

  }
}
par(mfrow = c(1, 1))

legend("bottom", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n",horiz=
plot(c(0,500),c(0,500),type="n")
#north
tree.diag(100,0,18.93,48.13,160.9,167.95,164.17)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,22.71,57.73,178.22,167.09,174.03)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,19.63,47.63,192.26,178.39,149.53)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,36.57,66.29,178.9,225.33,231.34)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,42.72,67.65,188.77,301.5,257.85)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,31.36,56.88,152.41,234.09,200.12)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,46.57,76.93,181.82,336.5,271.12)
plot(c(0,500),c(0,500),type="n")

#south
tree.diag(100,0,30.59,56.67,144.56,274.43,248.34)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,26.69,51.13,160.28,204.31,147.14)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,20.02,43.30,116.55,191.62,140.89)
```

```
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,44.90,68.32,126.67,254.69,204.88)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,27.61,51.84,144.75,201.27,157.86)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,14.37,30.25,153.15,128.16,66.83)
plot(c(0,500),c(0,500),type="n")
tree.diag(100,0,17.03,39.15,152.05,150.5,81.26)
```

```
#_____________________________________2D Plots of long/lat_____________________________
```

```
B=cbind(data$Longitude,data$Latitude)/100000
```

```
my.scale <- function(z, z.range)
{
  ((z-z.range[1])/(z.range[2]-z.range[1]))/2
}
my.circle <- function(x,y,r,...)
{
  theta<-seq(from=-pi,to=pi,length=100)
  #print(c(x,y,r))
  #print(x+r*cos(theta))
  #lines(x+r*cos(theta),y+r*sin(theta),...)
  PlotOnStaticMap(mymap,lon=x+r*cos(theta),lat=y+r*sin(theta),FUN=lines,add=T,...)
  #PlotOnStaticMap(mymap,lat=x,lon=y,FUN=points,add=T,...)

}
my.col=c(brewer.pal(9,"Set1")[-6],brewer.pal(7,"Set3")[-4])

long.lat.plot <- function(x, y, z, var.name="")
{
  #plot(x,y,type="n",xlab="longitude",ylab="lattitude",main=var.name,asp=1,xlim=c(14,21))
  #legend("topleft",levels(data$Location),col=my.col,lty=1)
```

```
  n <- length(z)
  for(i in 1:n)
  { r <- my.scale(z[i], range(z,na.rm=T))
    my.circle(x[i],y[i],r,col=my.col[dt$Location[i]==levels(dt$Location)])
  }
}

mymap<-PlotOnStaticMap(lat=B[,2],lon=B[,1],type="n")

long.lat.plot(B[,1],B[,2],data$TotHt,var.name="Total Height")

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$Ht1stBrnch,var.name="Height at first Branch")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$CnpyDm,var.name="Canopy Diameter")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$BslCrc,var.name="Basal Circumference")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$Crc1stBrnch,var.name="Circumference at first Branch")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$BrnchsOffMn,var.name="Branches off main stem")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$NoDchBrnchs,var.name="Number of Dichotomous events")
legend("", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")

long.lat.plot(B[,1],B[,2],data$NoLvHds,var.name="Number of Live Heads")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$NoDdHds,var.name="Number of dead heads")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$BrknHds,var.name="Broken heads")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$PrpDdHds,var.name="Proportion Dead Heads")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$PrpBrknHds,var.name="Proportion Broken Heads")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$StmCnd,var.name="Stem Condition")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
long.lat.plot(B[,1],B[,2],data$LfCnd,var.name="Leaf Condition")
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#_____________________________________regression_____________________________________
```

```
B=cbind(data$Longitude,data$Latitude)/100000


total.height<-lm(data$TotHt~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(total.height )

anova(total.height)



Height.1stBranch<- lm( data$Ht1stBrnch~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Height.1stBranch)
anova(Height.1stBranch)
plot(Height.1stBranch)



Canopy.diameter<- lm( data$CnpyDm~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Canopy.diameter)
anova(Canopy.diameter)
plot(Canopy.diameter)


Basal.circ <- lm( data$BslCrc~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Basal.circ)
anova(Basal.circ)
plot(Basal.circ)



Circ.1stBranch<- lm( data$Crc1stBrnch~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Circ.1stBranch)
anova(Circ.1stBranch)
plot(Circ.1stBranch)
```

```
Branchesoff.main<- lm( data$BrnchsOffMn~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Branchesoff.main)
anova(Branchesoff.main)
plot(Branchesoff.main)



Dichot.branches <- lm( data$NoDchBrnchs~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Dichot.branches)
anova(Dichot.branches)
plot(Dichot.branches)



Live.heads <- lm(data$NoLvHds~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Live.heads)
anova(Live.heads)
plot(Live.heads)



Dead.heads <- lm( data$NoDdHds~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Dead.heads)
anova(Dead.heads)
plot(Dead.heads)



Broken.heads<- lm( data$BrknHds~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Broken.heads)
anova(Broken.heads)
plot(Broken.heads)



Prop.dead.heads<- lm( data$PrpDdHds~B[,1]+B[,2]+data$Aspect+data$Slope)
```

```
summary(Prop.dead.heads)
anova(Prop.dead.heads)
plot(Prop.dead.heads)
```

```
Prop.broken.heads<- lm( data$PrpBrknHds~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Prop.broken.heads)
anova(Prop.broken.heads)
plot(Prop.broken.heads)
```

```
Stem.condition <- lm( data$StmCnd~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Stem.condition)
anova(Stem.condition)
plot(Stem.condition)
```

```
Leaf.condition<- lm( data$LfCnd~B[,1]+B[,2]+data$Aspect+data$Slope)
summary(Leaf.condition)
anova(Leaf.condition)
plot(Leaf.condition)
```

```
#_____________________________________GAM_____________________________________
#Total height
my.data=data.frame(total.height=data$TotHt,long=B[,1],lat=B[,2],aspect=data$Aspect,slope=da
total.height=gam(total.height~s(long*lat)+aspect+s(slope),data=my.data)
summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$TotHt,data$Location)
```

```
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(total.height,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")

#Height of first branch
my.data=data.frame(Height.1stBranch=data$Ht1stBrnch,long=B[,1],lat=B[,2],aspect=data$Aspect
Height.1stBranch=gam(Height.1stBranch~s(long*lat)+aspect+s(slope),data=my.data)
summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$Ht1stBrnch,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Height.1stBranch,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Canopy diameter
my.data=data.frame(Canopy.diameter=data$CnpyDm,long=B[,1],lat=B[,2],aspect=data$Aspect,slop
Canopy.diameter=gam(Canopy.diameter~s(long*lat)+aspect+s(slope),data=my.data)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$CnpyDm,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Canopy.diameter,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Basal circumference
my.data=data.frame(Basal.circ=data$BslCrc,long=B[,1],lat=B[,2],aspect=data$Aspect,slope=dat
Basal.circ=gam(Basal.circ~s(long*lat)+aspect+s(slope),data=my.data)

summary(total.height )
```

```
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$BslCrc,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Basal.circ,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Circumference at first branch
my.data=data.frame(Circ.1stBranch=data$Crc1stBrnch,long=B[,1],lat=B[,2],aspect=data$Aspect,
Circ.1stBranch=gam(Circ.1stBranch~s(long*lat)+aspect+s(slope),data=my.data)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$Crc1stBrnch,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Circ.1stBranch,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Brnaches off main stem
my.data=data.frame(Branchesoff.main=data$BrnchsOffMn,long=B[,1],lat=B[,2],aspect=data$Aspec
Branchesoff.main=gam(Branchesoff.main~s(long*lat)+aspect+s(slope),data=my.data,family=poiss

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$BrnchsOffMn,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Branchesoff.main,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Number of dichotomous
my.data=data.frame(Dichot.branches=data$NoDchBrnchs,long=B[,1],lat=B[,2],aspect=data$Aspect
Dichot.branches=gam(Dichot.branches~s(long*lat)+aspect+s(slope),data=my.data,family=poisson
```

```
summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$NoDchBrnchs,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Dichot.branches,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Number of live heads
my.data=data.frame(Live.heads=data$NoLvHds,long=B[,1],lat=B[,2],aspect=data$Aspect,slope=da
Live.heads=gam(Live.heads~s(long*lat)+aspect+s(slope),data=my.data,family=poisson)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$NoLvHds,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Live.heads,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Number dead heads
my.data=data.frame(Dead.heads=data$NoDdHds,long=B[,1],lat=B[,2],aspect=data$Aspect,slope=da
Dead.heads=gam(Dead.heads~s(long*lat)+aspect+s(slope),data=my.data,family=poisson)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$NoDdHds,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Dead.heads,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Broken heads
my.data=data.frame(Broken.heads=data$BrknHds,long=B[,1],lat=B[,2],aspect=data$Aspect,slope=
Broken.heads=gam(Broken.heads~s(long*lat)+aspect+s(slope),data=my.data,family=poisson)
```

```
summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$BrknHds,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Broken.heads,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Proportion of dead heads
my.data=data.frame(Prop.dead.heads=data$PrpDdHds,long=B[,1],lat=B[,2],aspect=data$Aspect,sl
Prop.dead.heads=gam(Prop.dead.heads~s(long*lat)+aspect+s(slope),data=my.data)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$PrpDdHds,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Prop.dead.heads,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Proportion of broken heads
my.data=data.frame(Prop.broken.heads=data$PrpBrknHds,long=B[,1],lat=B[,2],aspect=data$Aspec
Prop.broken.heads=gam(Prop.broken.heads~s(long*lat)+aspect+s(slope),data=my.data)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$PrpBrknHds,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Prop.broken.heads,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Stem condition
my.data=data.frame(Stem.condition=data$StmCnd,long=B[,1],lat=B[,2],aspect=data$Aspect,slope
```

```
Stem.condition=gam(Stem.condition~s(long*lat)+aspect+s(slope),data=my.data,family=poisson)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$StmCnd,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Stem.condition,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
#Leaf condition
my.data=data.frame(Leaf.condition=data$LfCnd,long=B[,1],lat=B[,2],aspect=data$Aspect,slope=
Leaf.condition=gam(Leaf.condition~s(long*lat)+aspect+s(slope),data=my.data,family=poisson)

summary(total.height )
mat=cbind(data$Longitude/100000,data$Latitude/100000,data$LfCnd,data$Location)
mat=na.omit(mat)
loc=factor(levels(data$Location)[mat[,4]])
s3d=scatterplot3d(x=mat[,1],y=mat[,2],z=mat[,3]-mean(mat[,3]),theta=0,phi=0)
plot(Leaf.condition,ask=T,theta=0,phi=0)
for (i in 1:length(levels(loc)))
  s3d$points3d(x=mat[loc==levels(loc)[i],1],y=mat[loc==levels(loc)[i],2],z=mat[loc==levels(

legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,cex=0.53,bty="n")
```

```
#_____________________________________reg 3d plot_____________________________________
reg.3d.plot<-function(x,y,z,beta0,beta1,beta2,variable.name="")
{ require(rgl)
  require(RColorBrewer)

  my.col=c(brewer.pal(9,"Set1")[-6],brewer.pal(7,"Set3")[-4])
```

```
  x.range=max(x,na.rm=T)-min(x,na.rm=T)
  y.range=max(y,na.rm=T)-min(y,na.rm=T)
  z.range=max(z,na.rm=T)-min(z,na.rm=T)

  open3d()
  aspect3d(x=1/(max(x.range,y.range)),y=1/(max(x.range,y.range)),z=1/(z.range))



for (i in 1:length(levels(data$Location)))
{
  x.y.z=cbind(x[data$Location==levels(data$Location)[i]],y[data$Location==levels(data$Locat


  x.y.z=na.omit(x.y.z)
  xx=x.y.z[,1]
  yy=x.y.z[,2]
  zz=x.y.z[,3]

  #spheres3d(x[data$Location==levels(data$Location)[i]],y[data$Location==levels(data$Locati
  if(length(zz)>0)
  spheres3d(xx,yy,zz,col=my.col[i],radius=0.01)
}


  x.vec=c(min(x),max(x))
  y.vec=c(min(y),max(y))
  x.y=expand.grid(x.vec,y.vec)
  z.hat=beta0+beta1*(x.y[,1])+beta2*(x.y[,2])
  z.hat=matrix(z.hat,ncol=2)
  persp3d(x.vec,y.vec,z.hat,col="blue",alpha=0.5,add=T)

  axes3d()
  title3d(xlab="Longitude",ylab="Latitude",zlab=variable.name)
  print(c(x.range,y.range,z.range))
}
```

```
reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$TotHt,589.637,21.803,24.097

snapshot3d("Total Height.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$Ht1stBrnch,119.120,27.349,1
snapshot3d("Height at first branch.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$CnpyDm,351.351,2.535,7.152,
snapshot3d("canopy diameter.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$BslCrc,114.703,36.003,20.97
snapshot3d("basal circumference.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$Crc1stBrnch,48.696,18.976,1
snapshot3d("circum at first branch.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$BrnchsOffMn,1.153,-0.016,-0
snapshot3d("branches off main.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$NoDchBrnchs,3.186,0.85,0.47
snapshot3d("dichot events.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$NoLvHds,-24.870,11.007,4.80
snapshot3d("live heads.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$NoDdHds,-77.988,13.413,5.22
snapshot3d("dead heads.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$BrknHds,18.447,0.951,1.077,
snapshot3d("broken heads.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$PrpDdHds,24.058,-1.560,-0.5
snapshot3d("prop dead heads.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$PrpBrknHds,25.989,-20.581,-
snapshot3d("prop broken heads.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$StmCnd,0.829,0.191,0.109,"s
snapshot3d("stem condition.png")

reg.3d.plot((data$Longitude)/100000,(data$Latitude)/100000,data$LfCnd,0.836,0.049,0.023,"Le
```

```
snapshot3d("leaf condition.png")



source("biplot.RFN")
X=cbind(data$Location,data$TotHt,data$Ht1stBrnch,data$CnpyDm,data$BslCrc,data$Crc1stBrnch,d
X=na.omit(X)
nonmissingLoc=X[,1]
X=X[,-1]
colnames(X)=c("TotHt","Ht1stBrnch","CnpyDm","BslCrc","Crc1stBrnch","BrnchsOffMn","NoDchBrnc


dyn.load("abagplot.f")

#_____________________________________climate data_____________________________________

#biovars for defn of vars
climate<-read.csv("~/Documents/semester 2 hons/honours project/datasets/Bioclim_nosipho.csv
temperature <- lm( climate$nsabio01~B[,1]+B[,2])
summary(temperature )
precipitation<- lm( climate$nsabio12~B[,1]+B[,2])
summary(precipitation)

#TotHt.clim<-(data$TotHt~nsabio01*nsabio02*nsabio03*nsabio04*nsabio05*nsabio06*nsabio07*nsa
step(TotHt.clim,scope=list(lower=~1,upper=~nsabio01*nsabio02*nsabio03*nsabio04*nsabio05*nsa
TotHt.clim<-lm(data$TotHt~nsabio01+nsabio02+nsabio03+nsabio04+nsabio05+nsabio06+nsabio07+ns
step(TotHt.clim)

step(TotHt.clim,scope=list(lower=~1,upper=~nsabio01*nsabio02*nsabio03*nsabio04*nsabio05*nsa

#_____________________________________pls_____________________________________

mod.VIP = function(X, Y, A, algorithm=NULL, ... )
{
  #algorithm = any of the plsr algorithm in the pls package
  options(digits=3)
  X.scal = scale(X, center=TRUE, scale=TRUE)  #scaled X
  Y.scal = scale(Y, center=TRUE, scale=TRUE)  #scaled Y

  #PLS and PLSR parameters
  Rmat = algorithm(X.scal,Y.scal,A)$projection  #(PxA) transformed X.weights matrix
```

```
  Tmat = algorithm(X.scal,Y.scal,A)$scores   #(NxA) X.scores matrix
  Bmat = algorithm(X.scal,Y.scal,A)$coefficients[,,A]  #(PxM) estimated PLSR coefficients m

  #VIP values
  P = ncol(X) #or nrow(Bmat)
  VIP = array(0, dim=P)  #(Px1) VIP values vector
  for (i in 1:P)
  {
    vip = sqrt( (P / (t(Bmat[i,]) %*% Bmat[i,] %*% sum(t(Tmat[,1:A]) %*% Tmat[,1:A]))) *
                  (t(Bmat[i,]) %*% Bmat[i,] %*% sum(t(Tmat[,1:A]) %*% Tmat[,1:A] %*% ((
                    Rmat[i,1:A])^2))) )
    VIP[i] = vip  #(Px1) VIP value vector
  }
  names(VIP) = rownames(Bmat)

  list(VIP.values=VIP, X.impor=which(VIP >= 0.8))


}

X=as.matrix(climate[,5:22])
Y1=cbind(data[,13:19])
Y2=cbind(data[,20:24],data[,33:34])

Y1=as.matrix(Y1)
mat=cbind(X,Y1)
mat=na.omit(mat)


X=mat[,1:ncol(X)]
Y1=mat[,-(1:ncol(X))]

Xstar=scale(X)
Ystar=scale(Y1)
my.plsr=simpls.fit(Y=Ystar,X=Xstar,ncomp=18)
plot(apply(residuals(my.plsr)^2,3,sum),type="b")
plot(mod.VIP(Xstar, Ystar, A=17, algorithm=simpls.fit)$VIP.values,type="b",xaxt="n",xlab=""
axis(side=1,at=1:18,colnames(X),las=2)
lines(c(0,19),rep(0.8,2),lty=2)

Xstar=scale(X[,c(1,2,4,10,11)])
```

```
Ystar=scale(Y1)

my.plsr2=simpls.fit(Y=Ystar,X=Xstar,ncomp=5)
plot(apply(residuals(my.plsr2),3,sum),type="b")
coeff=my.plsr2$coefficients
coeff[,,5]

X=as.matrix(climate[,5:22])
Y2=as.matrix(Y2)
mat=cbind(X,Y2)
mat=na.omit(mat)
X=mat[,1:ncol(X)]
Y2=mat[,-(1:ncol(X))]
Xstar=scale(X)
Ystar=scale(Y2)
my.plsr=simpls.fit(Y=Ystar,X=Xstar,ncomp=18)
plot(apply(residuals(my.plsr)^2,3,sum),type="b")
plot(mod.VIP(Xstar, Ystar, A=15, algorithm=simpls.fit)$VIP.values,type="b",xaxt="n",xlab=""
mod.VIP(Xstar, Ystar, A=15, algorithm=simpls.fit)$VIP.values
axis(side=1,at=1:18,colnames(X),las=2)
lines(c(0,19),rep(0.8,2),lty=2)

Xstar=scale(X[,c(1,3:6,10:12,15,17)])
Ystar=scale(Y2)

my.plsr2=simpls.fit(Y=Ystar,X=Xstar,ncomp=10)
plot(apply(residuals(my.plsr2),3,sum),type="b")
coeff=my.plsr2$coefficients
coeff[,,10]
for (i in 1:ncol(Y2))
{

  m0<-lm(Y1[,i]~nsabio04,data=as.data.frame(X))
  print(summary(m0))
}

mod.VIP(X, Y1, A=3, algorithm=oscorespls.fit)
```

```
betas=plsr$coefficients
betas

par(mfrow = c(1, 1))

#___________________________________Plotting locations_____________________________________
B=cbind(data$Longitude,data$Latitude)/100000

loc.col=rep(NA,nrow(data))
for (i in 1:nrow(data))
  loc.col[i]=my.col[data$Location[i]==levels(data$Location)]
library(RgoogleMaps)
map<-PlotOnStaticMap(lat=B[,2],lon=B[,1],cex=1.5,pch=20,col=loc.col)
legend("topleft", legend = levels(data$Location),fill=my.col,bg=NULL,bty="n")


#_________________________Biplot_____________________________________________
install.packages("UBbipl.dll", repos=NULL)
dyn.load("abagplot.dll")

CVAbiplot(X,indmat(nonmissingLoc))
CVAbiplot(X,indmat(nonmissingLoc),e.vects=c(2,3),alpha.bags=list(which=1:14))


-------------------------------------------------------------------------------------------------


library(fda)

sample.date<-dta$DATE
year=substring(sample.date,1,4)
month=substring(sample.date,5,6)
day=substring(sample.date,7,8)
my.date=paste(year,month,day,sep="/")
dates=as.Date(my.date,"%Y/%m/%d")
date.count=as.numeric(dates)

monthly.dates=unique(substring(dates,1,7))
year=substring(monthly.dates,1,4)
```

```
month=substring(monthly.dates,6,7)
day=15
my.date=paste(year,month,day,sep="/")
monthly.dates=as.Date(my.dat,"%Y/%m/%d")
month.count=as.numeric(monthly.dates)

quarterly.dates=rep(NA,length(monthly.dates))
quarterly.dates[month=="01"]="summer"
quarterly.dates[month=="02"]="summer"
quarterly.dates[month=="03"]="autumn"
quarterly.dates[month=="04"]="autumn"
quarterly.dates[month=="05"]="autumn"
quarterly.dates[month=="06"]="winter"
quarterly.dates[month=="07"]="winter"
quarterly.dates[month=="08"]="winter"
quarterly.dates[month=="09"]="spring"
quarterly.dates[month=="10"]="spring"
quarterly.dates[month=="11"]="spring"
quarterly.dates[month=="12"]="summer"
quarterly.dates=paste(quarterly.dates,ifelse(month=="12",as.numeric(year)+1,year))

quart.med<-tapply(month.count,quarterly.dates,median)
quarterly.levels <- levels(factor(quarterly.dates))
cbind(names(quart.med),quarterly.levels)
quarterly.count <- rep(NA,length(month.count))
for (i in 1:length(quarterly.levels))
  quarterly.count[quarterly.dates==quarterly.levels[i]]<-quart.med[i]

# --- extract climate data from stations 2, 5 and 6
sample2<-dta[118:3039,]
sample5<-dta[9752:13297,]
sample6<-dta[13298:20196,]
sample2[sample2=="-9999"]<-NA
sample5[sample5=="-9999"]<-NA
sample6[sample6=="-9999"]<-NA
sample2$TMAX[sample2$TMAX<10] <- NA  # one zero value probably missing
datesample2<-date.count[118:3039]
datesample5<-date.count[9752:13297]
datesample6<-date.count[13298:20196]
date2sample2=dates[118:3039]
date2sample5=dates[9752:13297]
```

```
date2sample6=dates[13298:20196]
sample2=cbind(datesample2,date2=date2sample2,sample2)
sample5=cbind(datesample5,date2=date2sample5,sample5)
sample6=cbind(datesample6,date2=date2sample6,sample6)
#remove 3 observations in 2014 far from other data points for sample 5
#sample5$TMAX[datesample5>15000]=NA
# --- move NA omit to individual variables, TMAX, TMIN, rain
# --- datesample2, 5 and 6 contains numeric date.count values
# --- date2sample2, 5 and 6 contains the dates

#plot (sample2[,1], sample2$TMAX, type="l")
#plot (sample2[,1], sample2$TMIN, type="l")
#plot (sample2[,1], sample2$PRCP, type="l")

#plot (sample5[,1], sample5$TMAX, type="l")
#plot (sample5[,1], sample5$TMIN, type="l")
#plot (sample5[,1], sample5$PRCP, type="l")

#plot (sample6[,1], sample6$TMAX, type="l")
#plot (sample6[,1], sample6$TMIN, type="l")
#plot (sample6[,1], sample6$PRCP, type="l")

# === Create samples for the different tree locations
# ::::::::::::::::::::::::::::::::::::::::::::::::::

sub.sample.size = 0.9
random.rain.error = 0.001

# --- save the data in a list so that for loops can be used rather than repeated code for e
data <- vector("list",10)
names (data) <- c("Remhoogte","Hauchabfontein","Gorab","Namtib","Carolinahof","Kliphoek","G

#  par(ask=T)

date.vec=sample2[,1]
n=length(date.vec)
for (i in 1:7)  #7 regions associated with climate station 2: "Remhoogte","Hauchabfontein",
{
  data[[i]] <- vector("list", 3)
  names(data[[i]]) <- c("max","min","rain")
```

```
  subdate <- sample(date.vec, size=n*sub.sample.size, replace=F)
  subdate <- unique(c(subdate,range(date.vec)))
  subdate <- sort(subdate)
  mat <- data.frame (date.count=subdate,
                     maxtemp = sample2$TMAX[match(subdate,date.vec)]/10,
                     actual.date = sample2$date2[match(subdate,date.vec)])
  data[[i]][[1]] <- na.omit(mat)
  #       plot (mat, type="l")

  subdate <- sample(date.vec, size=n*sub.sample.size, replace=F)
  subdate <- unique(c(subdate,range(date.vec)))
  subdate <- sort(subdate)
  mat <- data.frame (date.count=subdate,
                     mintemp = sample2$TMIN[match(subdate,date.vec)]/10,
                     actual.date = sample2$date2[match(subdate,date.vec)])
  data[[i]][[2]] <- na.omit(mat)
  #       plot (mat, type="l")

  daily.rain <- sample2$PRCP
  monthly.rain <- tapply(daily.rain, substring(sample2$date2,1,7), sum, na.rm=T)
  monthly.rain[monthly.rain<1] <- 0.5
  log.rain <- log(monthly.rain)
  my.date <- paste(names(monthly.rain),"15",sep="-")
  monthly.nums <- as.numeric(as.Date(my.date,"%Y-%m-%d"))
  mat <- data.frame (date.count=monthly.nums,
                     lograin = log.rain + rnorm(length(log.rain), 0, random.rain.error),
                     actual.date = as.Date(my.date,"%Y-%m-%d"))
  data[[i]][[3]] <- na.omit(mat)
        plot (mat, type="l")
}

date.vec=sample6[,1]
n=length(date.vec)
for (i in 8:9)  #2 regions associated with climate station 6: "Bulletrap", "Rooifontein"
{
  data[[i]] <- vector("list", 3)
  names(data[[i]]) <- c("max","min","rain")

  subdate <- sample(date.vec, size=n*sub.sample.size, replace=F)
  subdate <- unique(c(subdate,range(date.vec)))
  subdate <- sort(subdate)
```

```
  mat <- data.frame (date.count=subdate,
                     maxtemp = sample6$TMAX[match(subdate,date.vec)]/10,
                     actual.date = sample6$date2[match(subdate,date.vec)])
  data[[i]][[1]] <- na.omit(mat)
  #       plot (mat, type="l")

  subdate <- sample(date.vec, size=n*sub.sample.size, replace=F)
  subdate <- unique(c(subdate,range(date.vec)))
  subdate <- sort(subdate)
  mat <- data.frame (date.count=subdate,
                     mintemp = sample6$TMIN[match(subdate,date.vec)]/10,
                     actual.date = sample6$date2[match(subdate,date.vec)])
  data[[i]][[2]] <- na.omit(mat)
  #       plot (mat, type="l")

  daily.rain <- sample6$PRCP
  monthly.rain <- tapply(daily.rain, substring(sample6$date2,1,7), sum, na.rm=T)
  monthly.rain[monthly.rain<1] <- 0.5
  log.rain <- log(monthly.rain)
  my.date <- paste(names(monthly.rain),"15",sep="-")
  monthly.nums <- as.numeric(as.Date(my.date,"%Y-%m-%d"))
  mat <- data.frame (date.count=monthly.nums,
                     lograin = log.rain + rnorm(length(log.rain), 0, random.rain.error),
                     actual.date = as.Date(my.date,"%Y-%m-%d"))
  data[[i]][[3]] <- na.omit(mat)
  #       plot (mat, type="l")
}

# Tinkas River associated with station 5
# no sampling required since it is the only location

data[[10]] <- vector("list", 3)
names(data[[10]]) <- c("max","min","rain")

mat <- data.frame (date.count=sample6[,1],
                   maxtemp = sample6$TMAX/10,
                   actual.date = sample6$date2)
data[[10]][[1]] <- na.omit(mat)
#       plot (mat, type="l")

mat <- data.frame (date.count=sample6[,1],
```

```
                  mintemp = sample6$TMIN/10,
                  actual.date = sample6$date2)
data[[10]][[2]] <- na.omit(mat)
#       plot (mat, type="l")

daily.rain <- sample6$PRCP
monthly.rain <- tapply(daily.rain, substring(sample6$date2,1,7), sum, na.rm=T)
monthly.rain[monthly.rain<1] <- 0.5
log.rain <- log(monthly.rain)
my.date <- paste(names(monthly.rain),"15",sep="-")
monthly.nums <- as.numeric(as.Date(my.date,"%Y-%m-%d"))
mat <- data.frame (date.count=monthly.nums,
                  lograin = log.rain,
                  actual.date = as.Date(my.date,"%Y-%m-%d"))
data[[10]][[3]] <- na.omit(mat)
#       plot (mat, type="l")

#  par (ask=F)
par(mar=c(1,1,1,1)) #use when the following shows
#Error in plot.new() : figure margins too large

# --- Check skewness of rain data
par(mfrow=c(3,4))
lapply (data, function(short.list)
{ mat <- short.list[[3]]
  hist(mat[,2])
})
# par (mfrow=c(1,1))
#*********************************************
par(mar=c(1,1,1,1))
par(mfrow=c(3,4))
for (i in 1:10)
{
  hist(data[[i]][[3]][,2],main="",xlab="Rainfall",ylab="Frequency")
  title(main=names(data)[i], line = -1)
}
#*********************************************
# === Fit continuous functions
# ::::::::::::::::::::::::::::

# --- Set up large basis
```

```
tempbasis=create.bspline.basis(range(date.count),nbasis=500)
rainbasis=create.bspline.basis(range(month.count),nbasis=length(month.count))

# --- select best smoothing paramater based on GCV

fn.to.optim <- function (lambda, item)
{
  gcv <- rep(NA,10)
  if (item<3) fdParobj.lambda <- fdPar(tempbasis, 2, lambda) else fdParobj.lambda <- fdPar(
  for (i in 1:10)
    gcv[i] <- smooth.basis(data[[i]][[item]][,1],as.numeric(data[[i]][[item]][,2]),fdParobj
  mean(gcv, na.rm=T)
}

optimal.lambda.max <- optim (par=0.0003, fn=fn.to.optim, method="Brent", lower=1e-15,upper=
optimal.lambda.max
fdParobj.max <- fdPar(tempbasis, 2, optimal.lambda.max)

optimal.lambda.min <- optim (par=0.0003, fn=fn.to.optim, method="Brent", lower=1e-15,upper=
optimal.lambda.min
fdParobj.min <- fdPar(tempbasis, 2, optimal.lambda.min)

#not working
optimal.lambda.rain <- optim (par=0.03, fn=fn.to.optim, method="Brent", lower=0.001,upper=2
#  optimal.lambda.rain
optimal.lambda.rain <- 500
fdParobj.rain <- fdPar(rainbasis, 2, optimal.lambda.rain)

# --- create output matrices for fitted values from smooth functions
all.dates.temp <- seq(from=min(date.count),to=max(date.count))
act.dates.temp <- seq(from=min(dates),to=max(dates), by=1)
maxmat <-minmat <-matrix(nrow=length(all.dates.temp),ncol=10)

colnames(maxmat) <- colnames(minmat) <- c("Remhoogte","Hauchabfontein","Gorab","Namtib","Ca

all.dates.rain=seq(from=min(month.count),to=max(month.count))
act.dates.rain <- seq(from=min(monthly.dates),to=max(monthly.dates), by=1)
rainmat=matrix(nrow=length(all.dates.rain),ncol=10)
colnames(rainmat)=c("Remhoogte","Hauchabfontein","Gorab","Namtib","Carolinahof","Kliphoek",

# --- fit functions
```

```
# -----------------
#length(maxmat[,i])=2922
#length(all.dates.temp)=6940
#length(all.dates.rain)=2892
#length(rainmat[,i])=2892
for (i in 1:10)
{
  mat <- as.matrix (data[[i]][[1]][,1:2])
  maxfd <- smooth.basis(mat[,1],mat[,2],fdParobj.max)$fd
  maxmat[,i] <- eval.fd (all.dates.temp, maxfd)
  maxmat[all.dates.temp < min(data[[i]][[1]][,1]) | all.dates.temp > max(data[[i]][[1]][,1]

  mat <- as.matrix (data[[i]][[2]][,1:2])
  minfd <- smooth.basis(mat[,1],mat[,2],fdParobj.min)$fd
  minmat[,i] <- eval.fd (all.dates.temp, minfd)
  minmat[all.dates.temp < min(data[[i]][[2]][,1]) | all.dates.temp > max(data[[i]][[2]][,1]

  mat <- as.matrix (data[[i]][[3]][,1:2])
  rainfd <- smooth.basis(mat[,1],mat[,2],fdParobj.rain)$fd
  rainmat[,i] <- exp(eval.fd (all.dates.rain, rainfd))
  #plot(smooth.basis(mat[,1],mat[,2],fdParobj.rain)$fd)
  rainmat[all.dates.rain < min(data[[i]][[3]][,1]) | all.dates.rain > max(data[[i]][[3]][,1

  # --- create plots to evaluate fit

  #par(mfrow=c(1,3))
  dev.new()
  plot (data[[i]][[1]][,3], data[[i]][[1]][,2], col="red",xlab="Dates",ylab="Max Temp")
  lines (act.dates.temp, maxmat[,i], lwd=2, col="black")
  dev.new()
   plot(act.dates.temp, maxmat[,i], lwd=2, col="black",xlab="Dates",ylab="Max Temp",type="l

 plot (data[[i]][[2]][,3], data[[i]][[2]][,2], col="blue",xlab="Dates",ylab="Min Temp")
  lines (act.dates.temp, minmat[,i], lwd=2, col="black")

 plot (data[[i]][[3]][,3], exp(data[[i]][[3]][,2])/10, col="green",xlab="Dates",ylab="Rainf
  lines (act.dates.rain, rainmat[,i], lwd=2, col="black")
}

# --- compare fits from different locations with the same station
par(mar=c(1,1,1,1))
```

```
require(RColorBrewer)
col.vec <- c(brewer.pal (7, "Dark2"), brewer.pal(3, "Accent"))
dev.new()
par(mfrow=c(1,1))
plot (range(act.dates.temp),c(0,40),type="n",xlab="Dates",ylab="Max Temp",main="JGH Van Der
for (i in 1:7)
  lines (act.dates.temp, maxmat[,i], col=col.vec[i])
plot (range(act.dates.temp),c(0,40),type="n",xlab="Dates",ylab="Max Temp",main="Springbok")
for (i in 8:9)
  lines (act.dates.temp, maxmat[,i], col=col.vec[i])
plot (range(act.dates.temp),c(0,40),type="n",xlab="Dates",ylab="Max Temp",main="Gobabeb")
lines (act.dates.temp, maxmat[,10], col=col.vec[10])

plot (range(act.dates.temp),c(0,40),type="n",xlab="Dates",ylab="Min Temp",main="JGH Van Der
for (i in 1:7)
  lines (act.dates.temp, minmat[,i], col=col.vec[i])
plot (range(act.dates.temp),c(0,40),type="n",xlab="Dates",ylab="Min Temp",main="Springbok")
for (i in 8:9)
  lines (act.dates.temp, minmat[,i], col=col.vec[i])
plot (range(act.dates.temp),c(0,40),type="n",xlab="Dates",ylab="Min Temp",main="Gobabeb")
lines (act.dates.temp, minmat[,10], col=col.vec[10])

plot (range(act.dates.rain),c(0,1500),type="n",xlab="Dates",ylab="Rainfall",main="JGH Van D
for (i in 1:7)
  lines (act.dates.rain, rainmat[,i], col=col.vec[i])
plot (range(act.dates.rain),c(0,1500),type="n",xlab="Dates",ylab="Rainfall",main="Springbok
for (i in 8:9)
  lines (act.dates.rain, rainmat[,i], col=col.vec[i])
plot (range(act.dates.rain),c(0,1500),type="n",xlab="Dates",ylab="Rainfall",main="Gobabeb")
lines (act.dates.rain, rainmat[,10], col=col.vec[10])

# === Define response variables
#::::::::::::::::::::::::::::::

can.dm=with(tree.data,tapply(CnpyDm,Location,mean,na.rm=T))
total.height=with(tree.data,tapply(TotHt,Location,mean,na.rm=T))
height.branch=with(tree.data,tapply(Ht1stBrnch,Location,mean,na.rm=T))
basal.circum=with(tree.data,tapply(BslCrc,Location,mean,na.rm=T))
circum.branch=with(tree.data,tapply(Crc1stBrnch,Location,mean,na.rm=T))
branch.main=with(tree.data,tapply(BrnchsOffMn,Location,mean,na.rm=T))
dichot.branch=with(tree.data,tapply(NoDchBrnchs,Location,mean,na.rm=T))
```

```
candm=can.dm[c(11,7,5,9,3,8,6,2,12,14)]
totht=total.height[c(11,7,5,9,3,8,6,2,12,14)]
ht1stbrnch=height.branch[c(11,7,5,9,3,8,6,2,12,14)]
bslcrc=basal.circum[c(11,7,5,9,3,8,6,2,12,14)]
crc1stbrnch=circum.branch[c(11,7,5,9,3,8,6,2,12,14)]
brnchsoffmn=branch.main[c(11,7,5,9,3,8,6,2,12,14)]
nodchbrnchs=dichot.branch[c(11,7,5,9,3,8,6,2,12,14)]

# === Functional regression
#::::::::::::::::::::::::::

# --- remove NA's for fit of all locations together
maxmat=data.frame(all.dates.temp, act.dates.temp, maxmat)
maxmat=na.omit(maxmat)
all.dates.max=maxmat[,1]
act.dates.max <- maxmat[,2]
maxmat=as.matrix(maxmat[,-(1:2)])


head(maxmat)
location.cols <- c("magenta","lightcoral","orange","mediumseagreen","green","grey","tan4","
plot (range(act.dates.max),range(maxmat),type="n",xlab="Year",ylab="Maximum tempterature",y
for (i in 1:ncol(maxmat))
  lines (act.dates.max, maxmat[,i], col=location.cols[i])
# add a legend to the plot!!!!
legend('topright', colnames(maxmat) ,
   lty=1, col=location.cols, bty='n', cex=.75)


minmat=data.frame(all.dates.temp, act.dates.temp, minmat)
minmat=na.omit(minmat)
all.dates.min=minmat[,1]
act.dates.min <- minmat[,2]
minmat=as.matrix(minmat[,-(1:2)])

head(minmat)
location.cols <- c("magenta","lightcoral","orange","mediumseagreen","green","grey","tan4","
plot (range(act.dates.min),range(minmat),type="n",xlab="Year",ylab="Minimum tempterature",y
for (i in 1:ncol(minmat))
  lines (act.dates.min, minmat[,i], col=location.cols[i])
```

```
legend('topright', colnames(maxmat) ,
   lty=1, col=location.cols, bty='n', cex=.75)

rainmat=data.frame(all.dates.rain, act.dates.rain, rainmat)
rainmat=na.omit(rainmat)
all.dates.rain=rainmat[,1]
act.dates.rain <- rainmat[,2]
rainmat=as.matrix(rainmat[,-(1:2)])

head(rainmat)
location.cols <- c("magenta","lightcoral","orange","mediumseagreen","green","grey","tan4","
plot (range(act.dates.rain),range(rainmat),type="n",xlab="Year",ylab="Rainfall")
for (i in 1:ncol(rainmat))
  lines (act.dates.rain, rainmat[,i], col=location.cols[i])

legend('topright', colnames(maxmat) ,
   lty=1, col=location.cols, bty='n', cex=.75)




# --- create single fd object for all locations

#where/how do we include roughness penalty smoothing
maxbasis=create.bspline.basis(range(all.dates.max),nbasis=length(all.dates.max))
fdParobj=fdPar(maxbasis,2,1e-15)
maxSmooth=smooth.basis(all.dates.max,maxmat,fdParobj)
maxfd=maxSmooth$fd
maxlist<-vector("list",2)
maxlist[[1]]<-rep(1,10)
maxlist[[2]] <- maxfd


par(mfrow=c(1,1))
plot(maxfd)

minbasis=create.bspline.basis(range(all.dates.min),nbasis=length(all.dates.min))
fdParobj=fdPar(minbasis,2,1e-15)
minSmooth=smooth.basis(all.dates.min,minmat,fdParobj)
minfd=maxSmooth$fd
minlist<-vector("list",2)
```

```
minlist[[1]]<-rep(1,10)
minlist[[2]] <- minfd
plot(minfd)

rainbasis=create.bspline.basis(range(all.dates.rain),nbasis=length(all.dates.rain))
fdParobj=fdPar(rainbasis,2,1e-15)
rainSmooth=smooth.basis(all.dates.rain,rainmat,fdParobj)
rainfd=rainSmooth$fd
rainlist<-vector("list",2)
rainlist[[1]]<-rep(1,10)
rainlist[[2]]<-rainfd

plot(rainfd)

# --- perform regression

maxminlist <- maxrainlist <- minrainlist <- vector("list",3)
allthreelist <- vector("list",4)

# - intercepts
 maxminlist[[1]] <- maxrainlist[[1]] <- minrainlist[[1]] <- allthreelist[[1]] <- rep(1,10)

# - predictors

maxminlist[[2]] <- maxfd
maxminlist[[3]] <- minfd
maxrainlist[[2]] <- maxfd
maxrainlist[[3]] <- rainfd
minrainlist[[2]] <- minfd
minrainlist[[3]] <- rainfd

allthreelist[[2]] <- maxfd
allthreelist[[3]] <- minfd
allthreelist[[4]] <- rainfd

# - parameters
conbasis=create.constant.basis(range(all.dates))
beta.1predictor.basis=create.bspline.basis(range(all.dates),nbasis=7)
beta.2predictors.basis=create.bspline.basis(range(all.dates),nbasis=4)
beta.3predictors.basis=create.bspline.basis(range(all.dates),nbasis=2,norder=1)
```

```
betafdParmax1=fdPar(beta.1predictor.basis, 2, optimal.lambda.max)
betafdParmin1=fdPar(beta.1predictor.basis, 2, optimal.lambda.min)
betafdParrain1=fdPar(beta.1predictor.basis, 2, optimal.lambda.rain)


betafdParmax2=fdPar(beta.2predictors.basis, 2, optimal.lambda.max)
betafdParmin2=fdPar(beta.2predictors.basis, 2, optimal.lambda.min)
betafdParrain2=fdPar(beta.2predictors.basis, 2, optimal.lambda.rain)



betafdParmax3=fdPar(beta.3predictors.basis, 2, optimal.lambda.max)
betafdParmin3=fdPar(beta.3predictors.basis, 2, optimal.lambda.min)
betafdParrain3=fdPar(beta.3predictors.basis, 2, optimal.lambda.rain)



betamaxlist <- betaminlist <- betarainlist <-vector("list",2)
betamaxminlist <- betamaxrainlist <- betaminrainlist <- vector("list",3)
betaallthreelist <- vector("list",4)

betamaxlist[[1]] <- conbasis
betaminlist[[1]] <- conbasis
betarainlist[[1]] <- conbasis
betamaxlist[[2]] <- betafdParmax1
betaminlist[[2]] <- betafdParmin1
betarainlist[[2]] <- betafdParrain1

betamaxminlist[[1]] <- conbasis
betamaxrainlist[[1]] <- conbasis
betaminrainlist[[1]] <- conbasis
betamaxminlist[[2]] <- betafdParmax2
betamaxminlist[[3]] <- betafdParmin2
betamaxrainlist[[2]] <- betafdParmax2
betamaxrainlist[[3]] <- betafdParrain2
betaminrainlist[[2]] <- betafdParmin2
betaminrainlist[[3]] <- betafdParrain2

betaallthreelist[[1]] <- conbasis
betaallthreelist[[2]] <- betafdParmax3
```

```
betaallthreelist[[3]] <- betafdParmin3
betaallthreelist[[4]] <- betafdParrain3


#=============================================================================================

                                  Max Temp Regressions
#=============================================================================================


#==============================Candm max temp plots===================================
fRegress.max <- fRegress (as.vector(candm), maxlist, betamaxlist)
beta.est.list <- fRegress.max$betaestlist
candm.betamax.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
#plot (act.dates.max, candm.betamax.max, type="l", xlab="Day", ylab="beta for maxtemp",main


candmhat=fRegress.max$yhatfdobj
resid=as.numeric(candm)-as.numeric(candmhat)
SigmaE.=sum(resid^2)/(10-fRegress.max$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.max, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Maximum Temperature Reg. Coeff. Candm", lwd=2,xaxt='n',ylim=
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.max[31],act.dates.max[439],act.dates.max[877],act.dates.max[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)


TSS=sum((candm-mean(candm))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.max$df-1))/(SigmaE.)
Fratio
```

```
#16.81814
RSQ=(TSS-RSS)/TSS
RSQ
#0.9832953
F.res=Fperm.fd(as.vector(candm),maxlist,betamaxlist)
for (i in 1:10){
dev.new()
plot (act.dates.max, (as.vector(candm.betamax.max)*maxmat)[,i], type="l", xlab="Day", ylab=
}

for (i in 1:10){
dev.new()
plot (act.dates.max,maxmat[,1], type="l", xlab="Day", ylab="x(t) for maxtemp",main=colnames
}


#=================================Total Height max tmep==============================
fRegress.max <- fRegress (as.vector(totht), maxlist, betamaxlist)
beta.est.list <- fRegress.max$betaestlist
totht.betamax.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
#plot (act.dates.max, totht.betamax.max, type="l", xlab="Day", ylab="beta for maxtemp",main

totht hat=fRegress.max$yhatfdobj
resid=as.numeric(totht)-as.numeric(totht hat)
SigmaE.=sum(resid^2)/(10-fRegress.max$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.max, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Maximum Temperature Reg. Coeff. Totht", lwd=2,xaxt='n',ylim=
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.max[31],act.dates.max[439],act.dates.max[877],act.dates.max[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)
```

```
TSS=sum((totht-mean(totht))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.max$df-1))/(SigmaE.)
Fratio
#2.385547
RSQ=(TSS-RSS)/TSS
RSQ
#0.8930414



#====================================height at first branch max temp==================

fRegress.max <- fRegress (as.vector(ht1stbrnch), maxlist, betamaxlist)
beta.est.list <- fRegress.max$betaestlist
ht1stbrnch.betamax.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
#plot (act.dates.max, ht1stbrnch.betamax.max, type="l", xlab="Day", ylab="beta for maxtemp"


ht1stbrnchhat=fRegress.max$yhatfdobj
resid=as.numeric(ht1stbrnch)-as.numeric(ht1stbrnchhat)
SigmaE.=sum(resid^2)/(10-fRegress.max$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.max, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Maximum Temperature Reg. Coeff. Ht1stbrnch", lwd=2,xaxt='n',
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.max[31],act.dates.max[439],act.dates.max[877],act.dates.max[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((ht1stbrnch-mean(ht1stbrnch))^2)
```

```
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.max$df-1))/(SigmaE.)
Fratio
#1.366249
RSQ=(TSS-RSS)/TSS
RSQ
#0.8270455



#====================================Basal Circumference max temp====================

fRegress.max <- fRegress (as.vector(bslcrc), maxlist, betamaxlist)
beta.est.list <- fRegress.max$betaestlist
bslcrc.betamax.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
#plot (act.dates.max, bslcrc.betamax.max, type="l", xlab="Day", ylab="beta for maxtemp",mai

bslcrchat=fRegress.max$yhatfdobj
resid=as.numeric(bslcrc)-as.numeric(bslcrchat)
SigmaE.=sum(resid^2)/(10-fRegress.max$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.max, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Maximum Temperature Reg. Coeff.Bslcrc", lwd=2,xaxt='n',ylim=
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.max[31],act.dates.max[439],act.dates.max[877],act.dates.max[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((bslcrc-mean(bslcrc))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.max$df-1))/(SigmaE.)
Fratio
```

```
#10.91529
RSQ=(TSS-RSS)/TSS
RSQ
# 0.9744921

#================================Crc1stbrnch max temp==============================

fRegress.max <- fRegress (as.vector(crc1stbrnch), maxlist, betamaxlist)
beta.est.list <- fRegress.max$betaestlist
crc1stbrnch.betamax.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
#plot (act.dates.max, crc1stbrnch.betamax.max, type="l", xlab="Day", ylab="beta for maxtemp

crc1stbrnchhat=fRegress.max$yhatfdobj
resid=as.numeric(crc1stbrnch)-as.numeric(crc1stbrnchhat)
SigmaE.=sum(resid^2)/(10-fRegress.max$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.max, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Maximum Temperature Reg. Coeff.Crc1stbrnch", lwd=2,xaxt='n',
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.max[31],act.dates.max[439],act.dates.max[877],act.dates.max[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((crc1stbrnch-mean(crc1stbrnch))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.max$df-1))/(SigmaE.)
Fratio
#51.03259
RSQ=(TSS-RSS)/TSS
RSQ
#0.9944325
```

```
#=========================================brnchsoffmn max temp=======================
fRegress.max <- fRegress (as.vector(brnchsoffmn), maxlist, betamaxlist)
beta.est.list <- fRegress.max$betaestlist
brnchsoffmn.betamax.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
#plot (act.dates.max, brnchsoffmn.betamax.max, type="l", xlab="Day", ylab="beta for maxtemp

brnchsoffmnhat=fRegress.max$yhatfdobj
resid=as.numeric(brnchsoffmn)-as.numeric(brnchsoffmnhat)
SigmaE.=sum(resid^2)/(10-fRegress.max$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.max, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Maximum Temperature Reg. Coeff.Brnchsoffmn", lwd=2,xaxt='n',
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.max[31],act.dates.max[439],act.dates.max[877],act.dates.max[1303],act.dates.
axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((brnchsoffmn-mean(brnchsoffmn))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.max$df-1))/(SigmaE.)
Fratio
#0.8196203
RSQ=(TSS-RSS)/TSS
RSQ
#0.7415132


#=================================nodchbrnchs max temp============================
fRegress.max <- fRegress (as.vector(nodchbrnchs), maxlist, betamaxlist)
beta.est.list <- fRegress.max$betaestlist
nodchbrnchs.betamax.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
#plot (act.dates.max, nodchbrnchs.betamax.max, type="l", xlab="Day", ylab="beta for maxtemp

nodchbrnchshat=fRegress.max$yhatfdobj
```

```
resid=as.numeric(nodchbrnchs)-as.numeric(nodchbrnchshat)
SigmaE.=sum(resid^2)/(10-fRegress.max$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.max, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Maximum Temperature Reg. Coeff.Nodchbrnchs", lwd=2,xaxt='n',
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.max[31],act.dates.max[439],act.dates.max[877],act.dates.max[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((nodchbrnchs-mean(nodchbrnchs))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.max$df-1))/(SigmaE.)
Fratio
#4.197042
RSQ=(TSS-RSS)/TSS
RSQ
#0.9362637


#=============================================================================================

                                Min Temp Regressions
#=============================================================================================

#=========================================candm min temp=============================
fRegress.min <- fRegress (as.vector(candm), minlist, betaminlist)
beta.est.list <- fRegress.min$betaestlist
candm.betamin.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
#plot (act.dates.min, candm.betamin.min, type="l", xlab="Day", ylab="beta for mintemp",main

candmhat=fRegress.min$yhatfdobj
resid=as.numeric(candm)-as.numeric(candmhat)
```

```
SigmaE.=sum(resid^2)/(10-fRegress.min$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.min, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab="Minimum Temperature Reg. Coeff.Candm", lwd=2,xaxt='n',ylim=c(

lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.min[31],act.dates.min[439],act.dates.min[877],act.dates.min[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((candm-mean(candm))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.min$df-1))/(SigmaE.)
Fratio
#0.6096243
RSQ=(TSS-RSS)/TSS
RSQ
#0.6808868


#=========================================totht min temp==============================
fRegress.min <- fRegress (as.vector(totht), minlist, betaminlist)
beta.est.list <- fRegress.min$betaestlist
totht.betamin.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
#plot (act.dates.min, totht.betamin.min, type="l", xlab="Day", ylab="beta for mintemp",main

totht hat=fRegress.min$yhatfdobj
resid=as.numeric(totht)-as.numeric(totht hat)
SigmaE.=sum(resid^2)/(10-fRegress.min$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.min, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
```

```
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab="Minimum Temperature Reg. Coeff.Totht", lwd=2,xaxt='n',ylim=c(
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.min[31],act.dates.min[439],act.dates.min[877],act.dates.min[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((totht-mean(totht))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.min$df-1))/(SigmaE.)
Fratio
#0.2858727
RSQ=(TSS-RSS)/TSS
RSQ
#0.5001384


#=========================================ht1stbrnch min temp=========================
fRegress.min <- fRegress (as.vector(ht1stbrnch), minlist, betaminlist)
beta.est.list <- fRegress.min$betaestlist
ht1stbrnch.betamin.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
#plot (act.dates.min, ht1stbrnch.betamin.min, type="l", xlab="Day", ylab="beta for mintemp"

ht1stbrnchhat=fRegress.min$yhatfdobj
resid=as.numeric(ht1stbrnch)-as.numeric(ht1stbrnchhat)
SigmaE.=sum(resid^2)/(10-fRegress.min$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.min, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab="Minimum Temperature Reg. Coeff.Ht1stbrnch", lwd=2,xaxt='n',yl
lines(betafd+2*betastderrfd, lty=2, lwd=1)
```

```
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.min[31],act.dates.min[439],act.dates.min[877],act.dates.min[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((ht1stbrnch-mean(ht1stbrnch))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.min$df-1))/(SigmaE.)
Fratio
#1.085929
RSQ=(TSS-RSS)/TSS
RSQ
#0.7916992


#=========================================bslcrc min temp==============================
fRegress.min <- fRegress (as.vector(bslcrc), minlist, betaminlist)
beta.est.list <- fRegress.min$betaestlist
bslcrc.betamin.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
#plot (act.dates.min, bslcrc.betamin.min, type="l", xlab="Day", ylab="beta for mintemp",mai

bslcrchat=fRegress.min$yhatfdobj
resid=as.numeric(bslcrc)-as.numeric(bslcrchat)
SigmaE.=sum(resid^2)/(10-fRegress.min$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.min, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab="Minimum Temperature Reg. Coeff.Bslcrc", lwd=2,xaxt='n',ylim=c
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.min[31],act.dates.min[439],act.dates.min[877],act.dates.min[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)
```

```
TSS=sum((bslcrc-mean(bslcrc))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.min$df-1))/(SigmaE.)
Fratio
#0.2909247
RSQ=(TSS-RSS)/TSS
RSQ
#0.8270455


#=========================================crc1stbrnch min temp=======================
fRegress.min <- fRegress (as.vector(crc1stbrnch), minlist, betaminlist)
beta.est.list <- fRegress.min$betaestlist
crc1stbrnch.betamin.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
#plot (act.dates.min, crc1stbrnch.betamin.min, type="l", xlab="Day", ylab="beta for mintemp

crc1stbrnchhat=fRegress.min$yhatfdobj
resid=as.numeric(crc1stbrnch)-as.numeric(crc1stbrnchhat)
SigmaE.=sum(resid^2)/(10-fRegress.min$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.min, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab="Minimum Temperature Reg. Coeff.Crc1stbrnch", lwd=2,xaxt='n',y
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.min[31],act.dates.min[439],act.dates.min[877],act.dates.min[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((crc1stbrnch-mean(crc1stbrnch))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.min$df-1))/(SigmaE.)
Fratio
#0.2454618
RSQ=(TSS-RSS)/TSS
```

```
RSQ
#0.46211

#=========================================brnchsoffmn min temp=======================
fRegress.min <- fRegress (as.vector(brnchsoffmn), minlist, betaminlist)
beta.est.list <- fRegress.min$betaestlist
brnchsoffmn.betamin.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
#plot (act.dates.min, brnchsoffmn.betamin.min, type="l", xlab="Day", ylab="beta for mintemp

brnchsoffmnhat=fRegress.min$yhatfdobj
resid=as.numeric(brnchsoffmn)-as.numeric(brnchsoffmnhat)
SigmaE.=sum(resid^2)/(10-fRegress.min$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.min, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab="Minimum Temperature Reg. Coeff.Brnchsoffmn", lwd=2,xaxt='n',y
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.min[31],act.dates.min[439],act.dates.min[877],act.dates.min[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((brnchsoffmn-mean(brnchsoffmn))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.min$df-1))/(SigmaE.)
Fratio
#0.5668043
RSQ=(TSS-RSS)/TSS
RSQ
#0.6648585


#=========================================nodchbrnchs min temp=======================
fRegress.min <- fRegress (as.vector(nodchbrnchs), minlist, betaminlist)
beta.est.list <- fRegress.min$betaestlist
```

```
nodchbrnchs.betamin.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
#plot (act.dates.min, nodchbrnchs.betamin.min, type="l", xlab="Day", ylab="beta for mintemp

nodchbrnchshat=fRegress.min$yhatfdobj
resid=as.numeric(nodchbrnchs)-as.numeric(nodchbrnchshat)
SigmaE.=sum(resid^2)/(10-fRegress.min$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.min, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]

dev.new()
plot(betafd, xlab="Day",ylab="Minimum Temperature Reg. Coeff.No. dichot events", lwd=2,xaxt
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.min[31],act.dates.min[439],act.dates.min[877],act.dates.min[1303],act.dates.

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)


TSS=sum((nodchbrnchs-mean(nodchbrnchs))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.min$df-1))/(SigmaE.)
Fratio
#0.2989781
RSQ=(TSS-RSS)/TSS
RSQ
#0.5113425



#==============================================================================

                              Rainfall Regressions
#==============================================================================
```

```
fRegress.rain <- fRegress (as.vector(candm), rainlist, betarainlist)
beta.est.list <- fRegress.rain$betaestlist
candm.betarain.rain <- eval.fd (all.dates.rain, beta.est.list[[2]]$fd)
#plot (act.dates.rain, candm.betarain.rain, type="l", xlab="Day", ylab="beta for rain",main


candmhat=fRegress.rain$yhatfdobj
resid=as.numeric(candm)-as.numeric(candmhat)
SigmaE.=sum(resid^2)/(10-fRegress.rain$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.rain, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Rainfall Reg. Coeff. Candm", lwd=2,xaxt='n',ylim=c(-22,24))
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.rain[17],act.dates.rain[439],act.dates.rain[867],act.dates.rain[1294],act.da

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((candm-mean(candm))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.rain$df-1))/(SigmaE.)
Fratio
#1.327307
RSQ=(TSS-RSS)/TSS
RSQ
#0.8221589



#==================================Total Height rain==============================
fRegress.rain <- fRegress (as.vector(totht), rainlist, betarainlist)
beta.est.list <- fRegress.rain$betaestlist
totht.betarain.rain <- eval.fd (all.dates.rain, beta.est.list[[2]]$fd)
#plot (act.dates.rain, totht.betarain.rain, type="l", xlab="Day", ylab="beta for Rainfall",
```

```
totht hat=fRegress.rain$yhatfdobj
resid=as.numeric(totht)-as.numeric(totht hat)
SigmaE.=sum(resid^2)/(10-fRegress.rain$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.rain, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Rainfall Reg. Coeff. Totht", lwd=2,xaxt='n',ylim=c(-45,45))
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.rain[17],act.dates.rain[439],act.dates.rain[867],act.dates.rain[1294],act.da

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((totht-mean(totht))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.rain$df-1))/(SigmaE.)
Fratio
#0.5741595
RSQ=(TSS-RSS)/TSS
RSQ
#0.6957029



#====================================height at first branch rain==================

fRegress.rain <- fRegress (as.vector(ht1stbrnch), rainlist, betarainlist)
beta.est.list <- fRegress.rain$betaestlist
ht1stbrnch.betarain.rain <- eval.fd (all.dates.rain, beta.est.list[[2]]$fd)
#plot (act.dates.rain, ht1stbrnch.betarain.rain, type="l", xlab="Day", ylab="beta for Rainf


ht1stbrnchhat=fRegress.rain$yhatfdobj
resid=as.numeric(ht1stbrnch)-as.numeric(ht1stbrnchhat)
```

```
SigmaE.=sum(resid^2)/(10-fRegress.rain$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.rain, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Rainfall Reg. Coeff. Ht1stbrnch", lwd=2,xaxt='n',ylim=c(-15,
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.rain[17],act.dates.rain[439],act.dates.rain[867],act.dates.rain[1294],act.da

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((ht1stbrnch-mean(ht1stbrnch))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.rain$df-1))/(SigmaE.)
Fratio
#0.840543
RSQ=(TSS-RSS)/TSS
RSQ
#0.770289


#====================================Basal Circumference rain====================

fRegress.rain <- fRegress (as.vector(bslcrc), rainlist, betarainlist)
beta.est.list <- fRegress.rain$betaestlist
bslcrc.betarain.rain <- eval.fd (all.dates.rain, beta.est.list[[2]]$fd)
#plot (act.dates.rain, bslcrc.betarain.rain, type="l", xlab="Day", ylab="beta for Rainfall"

bslcrchat=fRegress.rain$yhatfdobj
resid=as.numeric(bslcrc)-as.numeric(bslcrchat)
SigmaE.=sum(resid^2)/(10-fRegress.rain$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.rain, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
```

```
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Rainfall Reg. Coeff.Bslcrc", lwd=2,xaxt='n',ylim=c(-25,30))
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.rain[17],act.dates.rain[439],act.dates.rain[867],act.dates.rain[1294],act.da

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((bslcrc-mean(bslcrc))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.rain$df-1))/(SigmaE.)
Fratio
#0.4025269
RSQ=(TSS-RSS)/TSS
RSQ
# 0.5836798



#================================Crc1stbrnch rain================================

fRegress.rain <- fRegress (as.vector(crc1stbrnch), rainlist, betarainlist)
beta.est.list <- fRegress.rain$betaestlist
crc1stbrnch.betarain.rain <- eval.fd (all.dates.rain, beta.est.list[[2]]$fd)
#plot (act.dates.rain, crc1stbrnch.betarain.rain, type="l", xlab="Day", ylab="beta for Rain

crc1stbrnchhat=fRegress.rain$yhatfdobj
resid=as.numeric(crc1stbrnch)-as.numeric(crc1stbrnchhat)
SigmaE.=sum(resid^2)/(10-fRegress.rain$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.rain, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
```

```
plot(betafd, xlab="Day",ylab=" Rainfall Reg. Coeff.Crc1stbrnch", lwd=2,xaxt='n',ylim=c(-20,
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.rain[17],act.dates.rain[439],act.dates.rain[867],act.dates.rain[1294],act.da

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((crc1stbrnch-mean(crc1stbrnch))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.rain$df-1))/(SigmaE.)
Fratio
#0.6195843
RSQ=(TSS-RSS)/TSS
RSQ
#0.6833445



#=========================================brnchsoffmn rain=======================
fRegress.rain <- fRegress (as.vector(brnchsoffmn), rainlist, betarainlist)
beta.est.list <- fRegress.rain$betaestlist
brnchsoffmn.betarain.rain <- eval.fd (all.dates.rain, beta.est.list[[2]]$fd)
#plot (act.dates.rain, brnchsoffmn.betarain.rain, type="l", xlab="Day", ylab="beta for Rain

brnchsoffmnhat=fRegress.rain$yhatfdobj
resid=as.numeric(brnchsoffmn)-as.numeric(brnchsoffmnhat)
SigmaE.=sum(resid^2)/(10-fRegress.rain$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.rain, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Rainfall Reg. Coeff.Brnchsoffmn", lwd=2,xaxt='n',ylim=c(-0.0
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.rain[17],act.dates.rain[439],act.dates.rain[867],act.dates.rain[1294],act.da
```

```
axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((brnchsoffmn-mean(brnchsoffmn))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.rain$df-1))/(SigmaE.)
Fratio
# 4.768356
RSQ=(TSS-RSS)/TSS
RSQ
#0.9432081


#==================================nodchbrnchs rain==============================
fRegress.rain <- fRegress (as.vector(nodchbrnchs), rainlist, betarainlist)
beta.est.list <- fRegress.rain$betaestlist
nodchbrnchs.betarain.rain <- eval.fd (all.dates.rain, beta.est.list[[2]]$fd)
#plot (act.dates.rain, nodchbrnchs.betarain.rain, type="l", xlab="Day", ylab="beta for maxt

nodchbrnchshat=fRegress.rain$yhatfdobj
resid=as.numeric(nodchbrnchs)-as.numeric(nodchbrnchshat)
SigmaE.=sum(resid^2)/(10-fRegress.rain$df)
SigmaE=SigmaE.*diag(rep(1,10))
y2cMap = tempSmooth$y2cMap
stderrList = fRegress.stderr(fRegress.rain, y2cMap,SigmaE)
betafdPar       = beta.est.list[[2]]
betafd          = betafdPar$fd
betastderrList  = stderrList$betastderrlist
betastderrfd    = betastderrList[[2]]
dev.new()
plot(betafd, xlab="Day",ylab=" Rainfall Reg. Coeff.Nodchbrnchs", lwd=2,xaxt='n',ylim=c(-1,1
lines(betafd+2*betastderrfd, lty=2, lwd=1)
lines(betafd-2*betastderrfd, lty=2, lwd=1)

at=c(act.dates.rain[17],act.dates.rain[439],act.dates.rain[867],act.dates.rain[1294],act.da

axis(side=1, at=at, labels=strftime(at, format="%y"), las=1)

TSS=sum((nodchbrnchs-mean(nodchbrnchs))^2)
RSS=sum(resid^2)
Fratio=((TSS-RSS)/(fRegress.rain$df-1))/(SigmaE.)
```

```
Fratio
#1.28815
RSQ=(TSS-RSS)/TSS
RSQ
#0.8177382

#xi(t)
dev.new()
plot (act.dates.rain, (as.vector(nodchbrnchs.betarain.rain)*rainmat)[,1], type="l", xlab="D
dev.new()
plot (act.dates.rain,rainmat[,1], type="l", xlab="Day", ylab="beta for maxtemp",main="Numbe




#=====================================2 predictor Models============================


fRegress.maxmin <- fRegress (as.vector(candm), maxminlist, betamaxminlist)
beta.est.list <- fRegress.maxmin$betaestlist
candm.betamaxmin.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
dev.new()
plot (act.dates.max, candm.betamaxmin.max, type="l", xlab="Day", ylab="beta for maxmintemp
candm.betamaxmin.min <- eval.fd (all.dates.min, beta.est.list[[3]]$fd)
dev.new()
plot (act.dates.min, candm.betamaxmin.min, type="l", xlab="Day", ylab="beta for maxmintemp

fRegress.maxrain <- fRegress (as.vector(candm), maxrainlist, betamaxrainlist)
beta.est.list <- fRegress.maxrain$betaestlist
candm.betamaxrain.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
dev.new()
plot (act.dates.max, candm.betamaxrain.max, type="l", xlab="Day", ylab="beta for maxrain ma
candm.betamaxrain.rain <- eval.fd (all.dates.rain, beta.est.list[[3]]$fd)
dev.new()
plot (act.dates.rain, candm.betamaxrain.rain, type="l", xlab="Day", ylab="beta for maxrain


fRegress.minrain <- fRegress (as.vector(candm), minrainlist, betaminrainlist)
beta.est.list <- fRegress.minrain$betaestlist
candm.betaminrain.min <- eval.fd (all.dates.min, beta.est.list[[2]]$fd)
dev.new()
```

```
plot (act.dates.min, candm.betaminrain.min, type="l", xlab="Day", ylab="beta for minrain mi
candm.betaminrain.rain <- eval.fd (all.dates.rain, beta.est.list[[3]]$fd)
dev.new()
plot (act.dates.rain, candm.betaminrain.rain, type="l", xlab="Day", ylab="beta for minrain

# === does not work for 3 predictors
fRegress.allthree <- fRegress (as.vector(candm), allthreelist, betaallthreelist)
beta.est.list <- fRegress.allthree$betaestlist
candm.betaallthree.max <- eval.fd (all.dates.max, beta.est.list[[2]]$fd)
plot (act.dates.max, candm.betaallthree.max, type="l", xlab="Day", ylab="beta for maxtemp",
candm.betaallthree.min <- eval.fd (all.dates.min, beta.est.list[[3]]$fd)
plot (act.dates.min, candm.betaallthree.min, type="l", xlab="Day", ylab="beta for mintemp",
candm.betaallthree.rain <- eval.fd (all.dates.rain, beta.est.list[[4]]$fd)
plot (act.dates.rain, candm.betaallthree.rain, type="l", xlab="Day", ylab="beta for rain",m



\end{lstlisting}
```