Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Extreme neerslag voorspellen met behulp van k-nearest forest neighbors

(Engelse titel: Forecasting Extreme Precipitation Using k-nearest Forest Neighbors)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE en TECHNISCHE INFORMATICA

door

YINGHAO DAI

Delft, Nederland
Juli 2018

This page intentionally left blank

BSc verslag TECHNISCHE WISKUNDE

"Extreme neerslag voorspellen met behulp van k-nearest forest neighbors"

(Engelse titel: "Forecasting Extreme Precipitation Using k-nearest Forest Neighbors")

YINGHAO DAI

Technische Universiteit Delft

Begeleiders

Dr. J. Cai

J.J. Velthoen MSc

Overige commissieleden

Drs. E.M. van Elderen

Juli, 2018

Delft

This page intentionally left blank

Abstract

Precipitation has high spatial and temporal uncertainty, which makes it challenging to predict. We focus specifically on extreme amounts of precipitation. The Royal Dutch Meteorological Institute (KNMI) uses a numerical model, approximating the solutions to partial differential equations, to forecast precipitation and other metrics about the weather. These forecasts have systematic errors, due to the model's high sensitivity to input parameters. These errors can be corrected with statistical methods, by looking at the relation between the predicted and actual precipitation. We use a non-parametric regression set-up to estimate the conditional expectation of the weather given the forecasts of the numerical weather prediction model of the KNMI. Specifically, we focus on predicting the maximum precipitation in a three by three kilometers area in the Netherlands. There are several existing methods for solving non-parametric regression problems; in this thesis we will focus on k-nearest neighbors and random forests. A simulation study shows, however, that both these methods are not capable of dealing with more complex regression problems, such as forecasting extreme precipitation. Therefore, we are proposing a newly developed method, called k-nearest forest neighbors, which is a generalization of the random forests approach. This new method performs significantly better on the simulated data, compared to k-nearest neighbors and random forests. When applying the methods on a precipitation data set obtained from the KNMI, it also turns out that the method we developed has more predictive power than the numerical weather model and the existing non-parametric regression approaches.

Preface

You are reading the report "Forecasting Extreme Precipitation Using k-nearest Forest Neighbors", which has been written as part of my double bachelor in Applied Mathematics and Computer Science at the Delft University of Technology. I have been working on the project between April and July 2018.

One of my supervisors, Jasper Velthoen, is conducting a PhD research on weather forecasts, in cooperation with the Royal Dutch Meteorological Institute, and under supervision of my other supervisor, Juan-Juan Cai. Jasper formulated a short description of my project before the start, and I hope that the results I obtained will be valuable for his research.

I would like to thank Juan and Jasper for their excellent supervision and inspiring feedback throughout the whole project. After each meeting, with no exception, I walked out the room with a feeling of having become so much wiser in the past hour. Especially since we had no clear goal in mind when starting the project (we would "see how far we get" instead), their continuous guidance was essential to me. Apart from the weekly meetings, they were both willing to answer my questions whenever I dropped by their offices. It is needless to say that this support helped me a lot in completing the project.

I also want to thank Emiel van Elderen, for his great guidance in the colloquium part of this project, and for taking place in my thesis committee.

I hope you will enjoy reading this report. When doing so, please keep in mind that it would have been less enjoyable without my supervisors' useful feedback.

Yinghao Dai

Delft, July 2018

Contents

List of used symbols and abbreviations

symbol or abbreviation	**meaning**
d	number of covariates
D	number of most predictive covariates of the precipitation data set
$\mathbb{E}$	expectation of a random variable
k	parameter of k-nearest neighbors
K	parameter of k-nearest forest neighbors
KNMI	Royal Netherlands Meteorological Institute
m	number of samples
M	number of trees in a forest
MISE	mean integrated squared error
MSE	mean squared error
n	sample size
NWP	numerical weather prediction
$\mathbb{R}$	set of real numbers
RMSE	root mean squared error
s	parameter of random forests, same as `min.node.size`

Chapter 1

Introduction

In order to contribute to the citizens' safety, the Royal Netherlands Meteorological Institute (Koninklijk Nederlands Meteorologisch Instituut, or KNMI), and other companies, issue warnings whenever the weather will cause severe risks, i.e. in case of extreme weather events with high probability. Examples of risky events could be flooding, damage to buildings and infrastructure, or landslides, caused by heavy precipitation [1]. These warnings need to be issued in advance, to give people the opportunity to prepare themselves for the dangerous weather. However, such extreme precipitation events are hard to predict, due to the large spatial and temporal uncertainties concerned with precipitation. It goes without saying that being able to make accurate predictions regarding extreme weather, can be vital and life-saving.

The KNMI uses a numerical weather prediction (NWP) model, called HARMONIE. The model is based on partial differential equations describing the flows in the atmosphere, and approximates the solutions to these equations numerically. Systematic biases in the numerical forecasts arise and can be attributed to two causes. Firstly, the forecasts rely heavily on an initial condition which can not be measured exactly; and secondly, the numerical model is in itself an approximation to the real world and therefore creating systematic errors in the modeling process. Especially for extreme precipitation events, this forms a huge problem in forecasting [2]. Therefore, statistical post-processing techniques are used, where the NWP forecasts are used as predictor variables to estimate statistics of the conditional distribution of precipitation given the numerical forecasts. In this thesis, we focus on estimating the expectation of the conditional distribution. In other words, this means looking at the statistical relation between the precipitation forecast and the actual precipitation observations, in expectation.

Since the HARMONIE model does not only give forecasts on the precipitation, but also about atmospheric stability, airflows, precipitable water, and numerous other weather metrics that might be related to precipitation, we can use all these variables to predict the actual amount of precipitation. Our problem can then be stated as follows: given all these data, we want to estimate the conditional expectation of the amount of precipitation given all HARMONIE forecasts, such that when we only know the forecasts from the HARMONIE model, we can predict the expected amount of precipitation as accurately as possible. The KNMI has provided us with a data set, containing these forecasts together with the eventual precipitation observations, on which our models can be trained or validated.

In Chapter 2, we will explain this data set in more detail, and show some interesting properties of the data. We will see that our problem is an example of a non-parametric regression problem, which will be explained in Chapter 3. Existing methods of dealing with non-parametric regression will be introduced and explained in Chapter 4. We will evaluate the performance of these methods by doing a simulation study in Chapter 5. In order to address the problems arising with these approaches, we will introduce a new method, and show its performance in Chapter 6. In Chapter 7, we will apply all methods to real-world data of precipitation in the Netherlands. Finally, we will conclude the thesis in Chapter 8 and discuss areas for further research in Chapter 9.

Chapter 2

Precipitation data

2.1 KNMI data set

In the precipitation data set that the KNMI provided us, the Netherlands has been divided into twelve rectangular regions of roughly equal size, and a day has been split into four time windows of six hours, namely 0:00-6:00, 6:00-12:00, 12:00-18:00 and 18:00-24:00 UTC. For each region and each time window, we know the *hourly maximum precipitation* that was observed from radar data, within a three by three kilometers grid box in the region.[1] This hourly maximum precipitation is the most important metric for issuing, for instance, a code yellow warning.[2] Hence, this is the metric that we aim to predict.

In order to do so, we use other metrics from the HARMONIE model, which gives predictions on 2.5 by 2.5 kilometers grid boxes [3]. As mentioned in the introduction, the data set contains forecasts about atmosphere stability, airflows, precipitable water, and other weather metrics that might be related to precipitation. Moreover, we know the numerical prediction for the hourly maximum precipitation. In total, there are 42 measurements, of which we only know their hourly extreme values (i.e. minimum and maximum), leading to 84 numbers per region per time window. We have got these forecast data for different lead times: a *lead time* is defined as the time difference between the beginning of the interval for which the forecast is valid and the time at which the forecast is made. The available lead times are 6, 12, 18, 24, 30, 36 and 42 hours in advance. Using these data, we will see from how many hours in advance (which lead time) extreme precipitation can be predicted with enough accuracy.

2.2 Properties of the data

In order to gain insight on how the precipitation "behaves", e.g. whether there is more precipitation in certain months or regions than in others, we will make violin plots of the observed hourly precipitation values. A violin plot is similar to a box plot, with the only difference that there is a rotated kernel density plot on each side [4]. Hence, the violin plot is more informative than a box plot, since it shows the distribution of the data instead of only summary statistics such as the mean and some interquartile ranges. We can see in Figure 2.1(a) that there seems to be less extreme precipitation in coastal areas, namely regions 1, 2, 4, 7 and 10.[3] Since our intuition tells us that the total amount of precipitation should be larger in coastal areas, probably there are longer periods of rain along the coastline, with less extreme precipitation. Also, we see in

[1] We only have information about the warm seasons (April 15 until October 15) of the years 2010, 2011 and 2013, since these are the most interesting periods for extreme precipitation. For each year, there are approximately 8500 observations available.

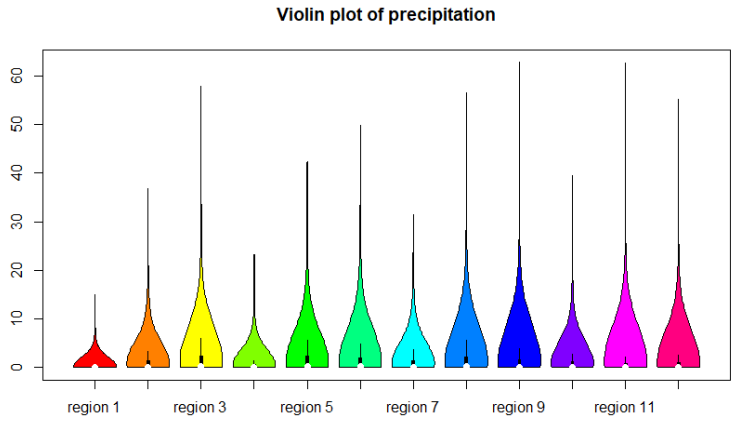
[2] Code yellow can be issued in several situations, ranging from extreme temperatures to heavy wind. We will be focusing on the criterion for precipitation: code yellow is issued when the local precipitation, i.e. within a three by three kilometers grid box, exceeds 30 millimeters per hour.

[3] For the reader's information: the regions are numbered from west to east (three regions per row), and then from north to south (four regions per column). This means that regions 1, 2 and 3 form the northern coastline, and regions 1, 4, 7 and 10 are along the western coast of the Netherlands.

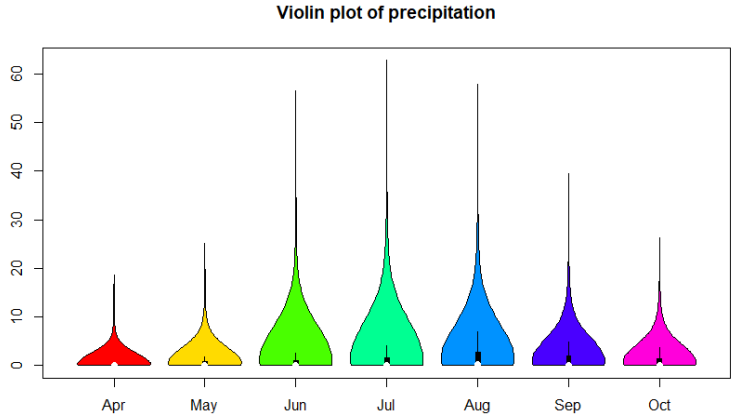
Figure 2.1(b) that there seems to be more extreme precipitation in summer months (June, July and August).

An average Dutch summer day starts with good weather in the morning, after which it will be hot in the afternoon, and heavily raining by the end of the day. It is thus interesting to look at the influence of the different time windows on the precipitation. As a reminder, the time windows are 0:00-6:00, 6:00-12:00, 12:00-18:00 and 18:00-24:00 UTC, corresponding to 2:00-8:00, 8:00-14:00, 14:00-20:00 and 20:00-2:00 local time. Figure 2.1(c) indeed tells us that there is more extreme precipitation at the end of the day.

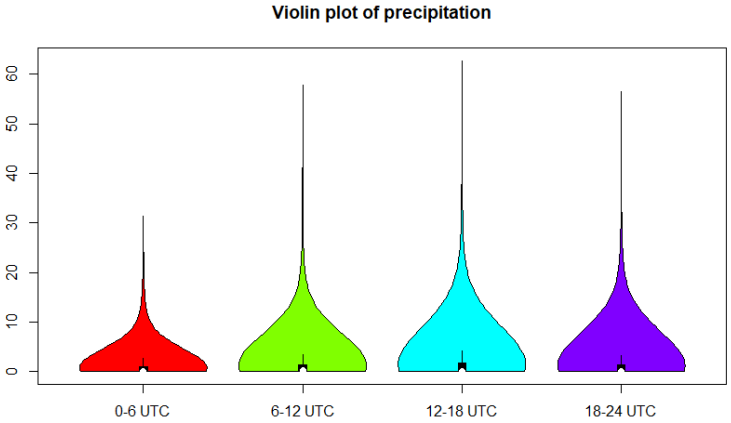
There could, of course, possibly be regions where the peak in the precipitation occurs in the early morning; or there could be months in which there is more extreme precipitation in coastal areas than in the rest of the country. Therefore, we also made violin plots for the twelve regions, split by month; violin plots for the seven months, split by region; violin plots for the twelve regions, split by time window; and violin plots for the four time windows, split by region. These are not really relevant for the thesis; interesting readers can find these plots in Figures A.1, A.2, A.3 and A.4 of the appendix.



(a) Twelve regions



(b) Seven months



(c) Four time windows

Figure 2.1: Violin plots of the observed hourly precipitation for all twelve regions, seven months and four time windows.

Chapter 3

Regression

In statistical modeling, we often want to estimate the relationship between variables, such as the numerically predicted and observed precipitation, or a person's foot length and body height. For example, we have observations $Y_1, \ldots, Y_n \in \mathbb{R}$, which are and independent identically distributed random variables. The expectation $\mathbb{E}(Y)$ of these random variables, where Y is a random variable with the same distribution as $Y_1, \ldots, Y_n$, can then be estimated by the average $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} Y_i$. Sometimes, for each $i = 1, \ldots, n$, we observe a d-dimensional vector of variables $\mathbf{X}_i = (X_{i1}, \ldots, X_{id})$, which contains information about Y_i. In other words, there is a regression function $f : \mathbb{R}^d \to \mathbb{R}$, that we do not know, satisfying

$$Y_i = f(\mathbf{X}_i) + \varepsilon_i$$

for each $i = 1, \ldots, n$, where ε_i is the error term or noise, a random variable with expectation 0, for instance $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. We refer to Y_i as the response variable, and to $X_{i1}, \ldots, X_{id}$ as *covariates*. We have got data points $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, where $\mathbf{X}_1, \ldots, \mathbf{X}_n \in \mathbb{R}^d$ and $Y_1, \ldots, Y_n \in \mathbb{R}$. Sometimes, we also refer to $\mathbf{X}_1, \ldots, \mathbf{X}_n$ as data points; the context will make clear which of the two is meant. The number of data points, n, will be called the *sample size*. Then the problem is as follows: given a new point $\mathbf{x}^* \in \mathbb{R}^d$, what is the expected corresponding value y^*? In other words, what is the conditional mean function $\mathbb{E}(Y|\mathbf{X} = \mathbf{x}) = f(\mathbf{x})$? The problem of estimating such a function is known as a *regression* problem.

If we assume that $f(\mathbf{x})$ (with $\mathbf{x} = (x_1, \ldots, x_d)$) is linear in the covariates $x_1, \ldots, x_d$, i.e. $f(x_1, \ldots, x_d) = \alpha + \beta_1 x_1 + \ldots + \beta_d x_d$, we have a *linear regression* problem. In this case, the method of least squares [5] can be used to estimate the parameter vector $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$, such that an estimator for the conditional mean function is given by $\hat{f}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & \cdots & x_d \end{bmatrix} \hat{\boldsymbol{\beta}}$, where $\mathbf{x} = (x_1, \ldots, x_d)$.

However, it is quite restrictive to assume that f is a linear function. If we drop this assumption, we are dealing with non-parametric regression [6]. Two of the methods of estimating the conditional mean function, are k-nearest neighbors, and random forests. The k-nearest neighbors method estimates $f(\mathbf{x})$ by taking an average over the k values of Y_i for which the Euclidean distance between $\mathbf{X}_i$ and $\mathbf{x}$ is the smallest. The random forests approach does something similar, taking an average over the Y_i values of the closest $\mathbf{X}_i$ points, using a data-driven distance measure, instead of the Euclidean one. Both methods will be explained in more detail in the next chapter.

There are numerous applications of non-parametric regression, mainly for prediction and forecasting. In Chapter 7, we will deal with applying non-parametric regression methods to the prediction of extreme precipitation in the Netherlands.

Chapter 4

Existing methods

Instead of predicting precipitation, we will illustrate the methods via a simple and more easily understandable example. Consider the problem of predicting a person's body height, using their foot and hair length. Using the observed foot length, hair length, and body height for n persons, we want to predict the body height of a new person, while knowing only their foot and hair length. In this case, we have a two-dimensional regression function $f : \mathbb{R}^2 \to \mathbb{R}$, satisfying

$$Y = f(X_1, X_2) + \varepsilon,$$

where X_1 and X_2 are covariates of the foot length and hair length, respectively. Moreover, ε is the noise term, reflecting the fact that people with exactly the same hair and foot lengths, do not necessarily have exactly the same body height. Y is the body height, depending on the foot length and hair length, with some noise. We have n people of whom we know foot length, hair length and body height: $(X_{11}, X_{21}, Y_1), \ldots, (X_{1n}, X_{2n}, Y_n)$. Now the problem is as follows: given a new person with foot length x_1^* and hair length x_2^*, what is their body height y^*? In other words, we are searching for the conditional mean function $\mathbb{E}[Y|(X_1, X_2) = (x_1, x_2)] = f(x_1, x_2)$.

Since we do not know how the body height depends on the hair and foot length, we have a *non-parametric* regression problem. In this case, the k-nearest neighbors method or the random forests approach can be used. Both methods are of interest in this thesis and will be discussed below.

4.1 k-nearest neighbors

Continuing with the example above, Figure 4.1 shows the n observations of foot length, hair length and body height, $\mathbf{X}_1, \ldots, \mathbf{X}_n$, which have been plotted in black. We aim to predict the body height of the red point $\mathbf{x}^*$, given only the foot and hair length. The k-nearest neighbors method has one parameter, denoted by k, and predicts, as the name already suggests, based on the k closest points (i.e. nearest neighbors) of the red point [7]. Closeness is in this case defined by the distance, which means we need a distance metric: usually the Euclidean distance $||\cdot||_2$ is used. Let $S_k(\mathbf{x}^*) \subseteq \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$ be the set of k nearest neighbors of $\mathbf{x}^*$. If we take $k = 4$ in our example, then this set consists of the people with a body height of 167, 188, 190 and 202 centimeters, respectively. The prediction for the body height of the red point can then be computed by the average of those four body heights, which is 187 centimeters.

In order to explain the rationale behind taking the average, we will first define a measure for the error of the prediction, namely the mean squared error (MSE). For a certain prediction $\hat{y}^*$, the mean squared error of data points $\mathbf{X}_i \in S_k(\mathbf{x}^*)$ equals

$$\frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} (\hat{y}^* - Y_i)^2. \tag{4.1}$$

Of course, we want to choose $\hat{y}^*$ in such a way that this error is minimized, which means that for

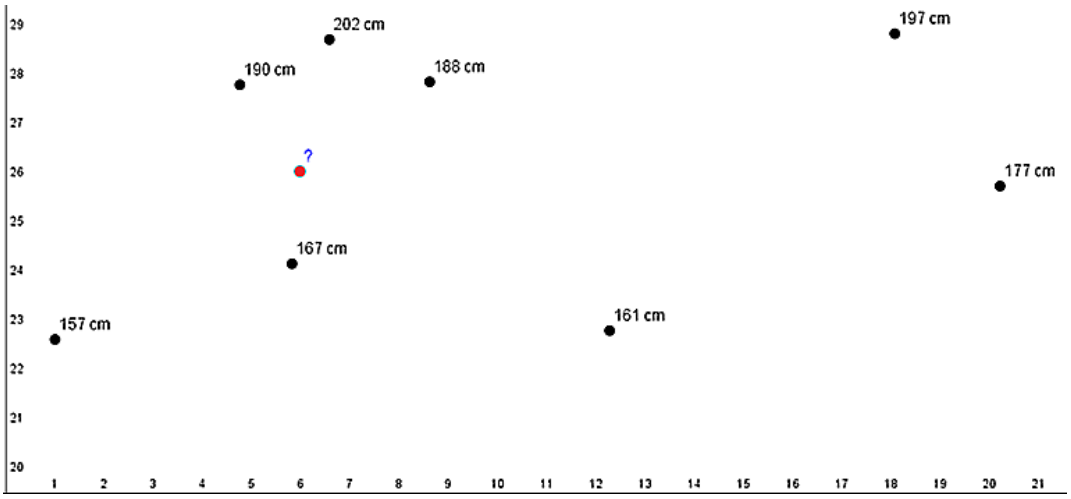


Figure 4.1: On the horizontal axis is the hair length in centimeters; on the vertical axis the foot length in centimeters. For the black points, the known body height is shown; the body height of the red point needs to be predicted.

the ultimate prediction, our error is

$$\min_{y^*} \frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} (y^* - Y_i)^2,$$

and

$$\hat{y}^* = \arg\min_{y^*} \frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} (y^* - Y_i)^2.$$

We can find $\hat{y}^*$ by differentiation with respect to y^*. Linearity of the differential operator yields

$$\frac{\mathrm{d}}{\mathrm{d}y^*} \left(\frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} (y^* - Y_i)^2 \right) = \frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} 2(y^* - Y_i) = \frac{2}{k} \left(k y^* - \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} Y_i \right).$$

This evaluates to zero for

$$y^* = \frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} Y_i, \tag{4.2}$$

and for that value of y^*, we have

$$\frac{\mathrm{d}^2}{\mathrm{d}(y^*)^2} \left(\sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} (y^* - Y_i)^2 \right) = 2k > 0. \tag{4.3}$$

Hence, we indeed have a minimum in $y^* = \frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} Y_i$, which means

$$\hat{y}^* = \frac{1}{k} \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} Y_i =: \mathrm{avg}_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} Y_i, \tag{4.4}$$

the average of the k-nearest neighbors[1].

A generalization of the method is to use different weights for the average, depending on the distance between the red point $\mathbf{x}^*$ and the covariate $\mathbf{X}_i$:

$$\hat{y}^* = \sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} w_i(\mathbf{x}^*) Y_i. \tag{4.5}$$

[1]We do not use the notation $\bar{Y}$ for the average, since we want to make clear over which points the average is being taken.

Note that in any case, we rely on the assumption that people with similar hair and foot lengths, have similar body heights as well. Moreover, note that the weights need to sum to one:

$$\sum_{\mathbf{X}_i \in S_k(\mathbf{x}^*)} w_i(\mathbf{x}^*) = 1.$$

4.2 Random forests

It is probably clear that there is a (strong) correlation between humans' foot lengths and body heights. Nevertheless, we do not expect a dependence between hair lengths and body heights. This is where the term *redundancy* comes in; the hair length is in this case a so-called redundant variable, as there is no reason to assume that people with similar hair lengths, have similar body heights as well. People with similar foot lengths, on the other hand, are expected to have similar body heights. Hence, in Figure 4.1, the black point with a body height of 177 centimeters, gives a much better indication for the body height of the red point, than for example the point with a height of 202 centimeters; yet the latter point is closer to the red point than the former. Exactly this problem, the random forest approach is able to deal with. It defines the distance between points differently, and in such a way that the "177 cm" point is indeed closer to the red point than the "202 cm" point. Note that there are several versions of the random forests approach, of which classical random forests [8] are best-known. We are, however, using the Generalized Random Forests approach, which has been published very recently [9].

This approach works by constructing several so-called *regression trees*, an example of such a tree has been given in Figure 4.2 (with a great amount of fantasy, one can see this as a tree growing upside-down, with the root on top and the leaves at the bottom). The first step in constructing a tree, is taking a random sample (without replacement) of the n data points, usually of size $n/2$, let's call this sample $S \subseteq \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$. This set S will be further split randomly into two sets of roughly equal size, namely S_t and S_w. The (approximately $n/4$) data points in S_t will be used to construct the tree, those in S_w will be used to determine the weights; it will become clear what that means. Splitting S into S_t and S_w causes the tree to be what Wager and Athey call "honest" [9].

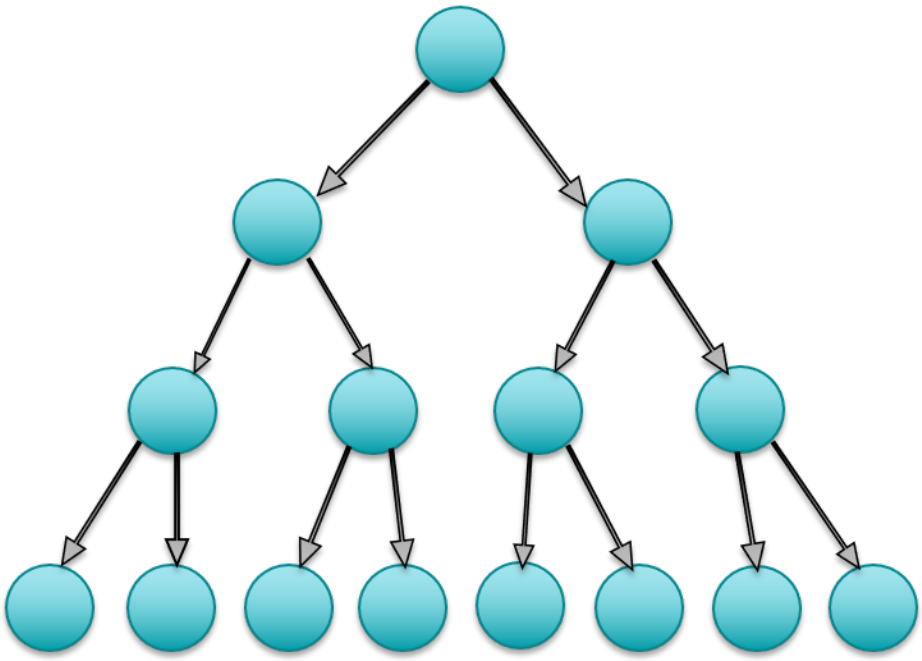


Figure 4.2: Example of a regression tree.

Before looking into how trees are constructed, note that, analogous to equation (4.1), the

mean squared error of data points $\mathbf{X}_1, \ldots, \mathbf{X}_p$ for a certain prediction $\hat{y}^*$ equals

$$\frac{1}{p}\sum_{i=1}^{p}(\hat{y}^* - Y_i)^2.$$

In the same way as done in equations (4.1) to (4.4), we find that we have a minimum in $y^* = \frac{1}{p}\sum_{i=1}^{p} Y_i$, which means

$$\hat{y}^* = \frac{1}{p}\sum_{i=1}^{p} Y_i =: \operatorname{avg}_{i=1}^{p} Y_i,$$

the average of the data points. The mean squared error is then equal to

$$\min_{y^*}\sum_{i=1}^{p}(y^* - Y_i)^2 = \sum_{i=1}^{p}[(\operatorname{avg}_{i=1}^{p} Y_i) - Y_i]^2.$$

For constructing the regression tree, the points in S_t will be split recursively, into so-called nodes.

The first node will consist of all points in S_t. Then we will split this node along one of the d dimensions. In our example, we either put people with short feet in one group and those with long feet in another, or we put all people with short hair into one group and those with long hair in another. In other words, we choose $j \in \{1, \ldots, d\}$ and $a \in \mathbb{R}$ and put all $\mathbf{X}_i = (X_i^1, \ldots, X_i^d) \in S_t$ with $X_i^j \leq a$ in one set, and those with $X_i^j > a$ in another, making two nodes out of each single node. j and a are chosen in such a way that the total MSE, the sum of the MSEs in both resulting nodes, is minimized. Let $\mathbf{X}_1, \ldots, \mathbf{X}_p$ again be the data points in our node, where $\mathbf{X}_i = (X_i^1, \ldots, X_i^d)$, and let $p_1(j,a) = \#\{i : X_i^j \leq a\}$ and $p_2(j,a) = \#\{i : X_i^j > a\}$. Then we need to choose $j \in \{1, \ldots, d\}$ and $a \in \mathbb{R}$ such that the MSE

$$\frac{1}{p_1(j,a)}\sum_{i:X_i^j \leq a}[(\operatorname{avg}_{i:X_i^j \leq a} Y_i) - Y_i]^2 + \frac{1}{p_2(j,a)}\sum_{i:X_i^j > a}[(\operatorname{avg}_{i:X_i^j > a} Y_i) - Y_i]^2$$

is minimized. In other words, we let

$$(j,a) = \arg\min_{(\tilde{j},\tilde{a})}\left(\frac{1}{p_1(\tilde{j},\tilde{a})}\sum_{i:X_i^{\tilde{j}} \leq \tilde{a}}[(\operatorname{avg}_{i:X_i^{\tilde{j}} \leq \tilde{a}} Y_i) - Y_i]^2 + \frac{1}{p_2(\tilde{j},\tilde{a})}\sum_{i:X_i^{\tilde{j}} > \tilde{a}}[(\operatorname{avg}_{i:X_i^{\tilde{j}} > \tilde{a}} Y_i) - Y_i]^2\right)$$

Note that since we have finitely many data points, we in fact have finitely many choices for a resulting in different outcomes. Now we have split our node containing data points $\{\mathbf{X}_1, \ldots, \mathbf{X}_p\}$ into two smaller nodes, containing $\{\mathbf{X}_i : X_i^j \leq a\}$ and $\{\mathbf{X}_i : X_i^j > a\}$, respectively. Then we repeat this process for each of the resulting nodes, giving us four even smaller nodes. We can split again and again, ending up with more and smaller nodes, until we reach the stop condition, which is a parameter called `min.node.size` or simply s, specifying the minimum number of points from S_t that need to be in a node. When the stop condition has been reached, we stop splitting, ending up with nodes (leaves) such as in Figure 4.3.

After having built the tree with the data points from S_t, we determine in which leaf every data point from S_w ends up. For a fixed $\mathbf{x}^*$, the tree obtains a prediction of y^* by averaging all observations $Y \in S_w$ that are in the same leaf.

One tree does not constitute a forest yet, so we start again by taking a new, independent random sample of the data points $S \subseteq \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$, which will be, with very high probability, different from our previous sample. Again, we split S into two sets S_t and S_w of roughly equal size, build the tree with points from S_t and determine another prediction for y^* based on this tree and the points from S_w. In total, many trees are being built, each having their own prediction for y^*.

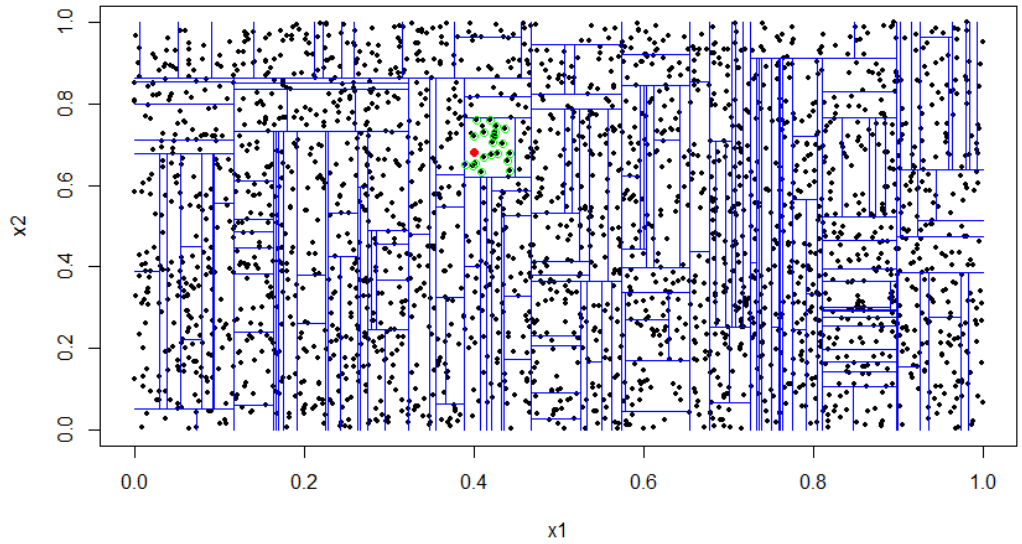


Figure 4.3: Example of how the leaves in a tree could look like. The axes are (in this case two) covariates. In this example, the tree prediction of the red point's y-value is the average of the green-circled data points.

Now let M be the number of trees, and y_j^* for $j = 1, \ldots, M$ the prediction of tree j, then we have

$$\hat{y}_j^* = \frac{\sum_{i=1}^{n} L_j(\mathbf{x}^*, \mathbf{X}_i) Y_i}{\sum_{k=1}^{n} L_j(\mathbf{x}^*, \mathbf{X}_k)},$$

where $L_j(\mathbf{x}^*, \mathbf{X}_i) = 1$ if $\mathbf{x}^*$ and $\mathbf{X}_i$ are in the same leaf of tree j; and $L_j(\mathbf{x}, \mathbf{y}) = 0$ otherwise. Note that $L_j(\mathbf{x}, \mathbf{y}) = 0$ for all $\mathbf{X}_i \notin S_w$, since for a fixed tree, the leaf nodes only contain observations from S_w. Then the prediction of the random forest is given by

$$\hat{y}^* = \frac{1}{M} \sum_{j=1}^{M} y_j^*,$$

the average of all tree predictions. Note that we can also write

$$\hat{y}^* = \frac{1}{M} \sum_{i=1}^{n} \sum_{j=1}^{M} \frac{L_j(\mathbf{x}^*, \mathbf{X}_i) Y_i}{\sum_{k=1}^{n} L_j(\mathbf{x}^*, \mathbf{X}_k)} = \sum_{i=1}^{n} w_i(\mathbf{x}^*) Y_i \tag{4.6}$$

for $w_i(\mathbf{x}^*) = \frac{1}{M} \sum_{j=1}^{M} \frac{L_j(\mathbf{x}^*, \mathbf{X}_i)}{\sum_{k=1}^{n} L_j(\mathbf{x}^*, \mathbf{X}_k)}$. Since $w_i(\mathbf{x}^*) \geq 0$ for all $i = 1, \ldots, n$ and all test points $\mathbf{x}^*$, and since $\sum_{i=1}^{n} w_i(\mathbf{x}^*) = 1$, it follows that $\hat{y}^*$ is a weighted average of $Y_1, \ldots, Y_n$, and the weight for Y_i, when predicting y^*, is given by $w_i(\mathbf{x}^*)$. Thus, data points ending up in the same leaf as $\mathbf{x}^*$ more often, get more weight when predicting y^*. Equation (4.6) shows a relationship between k-nearest neighbors and random forests (compare this to equation (4.5)), since they both predict by using a weighted average of the y-values of the data points and can thus be considered *weighted neighborhoods schemes* [10]. An example of a weight assignment when using random forests, is

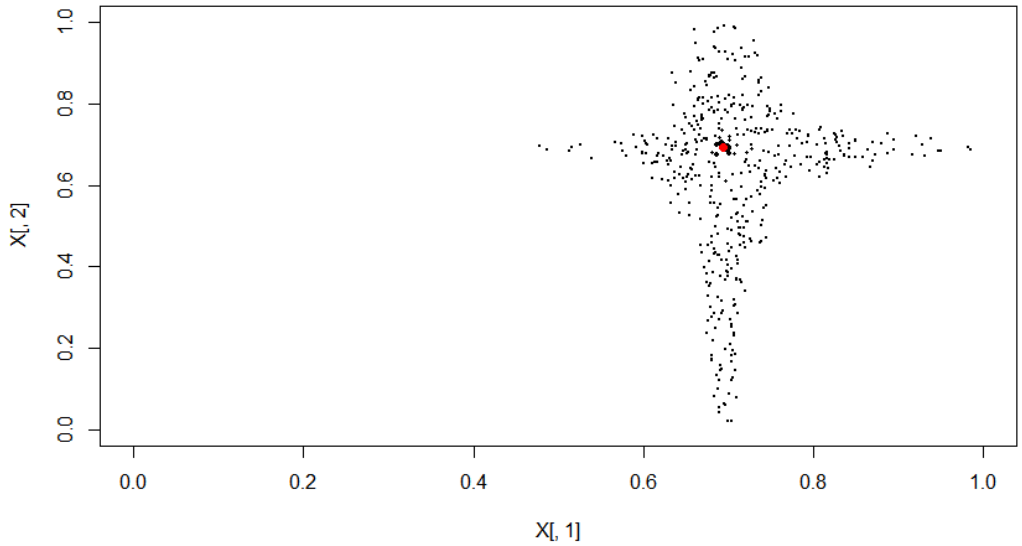


Figure 4.4: The red point is the (fixed) test point $\mathbf{x}^*$, and the black points are all data points $\mathbf{X}_i$ having nonzero weight $w_i(\mathbf{x}^*)$ when predicting y^*. The greater $w_i(\mathbf{x}^*)$, the bigger $\mathbf{X}_i$ has been depicted.

given in Figure 4.4, where the test point is colored red, and the black points are its neighborhood. We can see that in general, data points with roughly the same x_1-value as the test point, get larger weights than data points having comparable x_2-value. Hence, probably the first covariate has more predictive power than the second. The first covariate could be, for instance, the foot length in our example, whereas the second covariate is the hair length. This example illustrates that the shape of the neighborhood gives information about the importance of the covariates [10].

Chapter 5

Performance evaluation using simulation

5.1 Bias-variance trade-off

A measure for the performance of a method, is the error of the predictions. This can be represented by the mean squared error, as already mentioned in the previous chapter. The MSE is the expectation of the square of the difference between the true and predicted value. In other words, let $\hat{f}(\mathbf{x}^*)$ be the prediction for the underlying true value y^*, then the MSE is given by

$$\text{MSE}(\hat{f}(\mathbf{x}^*)) = \mathbb{E}[(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}^*))^2] = \text{var}(\hat{f}(\mathbf{x}^*)) + \text{bias}^2(\hat{f}(\mathbf{x}^*)),$$

and can be decomposed in a variance part $\text{var}(\hat{f}(\mathbf{x}^*)) = \mathbb{E}[(\hat{f}(\mathbf{x}^*) - \mathbb{E}(\hat{f}(\mathbf{x}^*)))^2]$ and a bias part $\text{bias}(\hat{f}(\mathbf{x}^*)) = \mathbb{E}(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}^*))$. Thus, the variance is the variability of the predictions, and the bias is the extent to which the expected prediction differs from the actual value; both the variance and bias cause error. If we want to minimize the MSE, we are dealing with the bias-variance trade-off (or bias-variance dilemma) [11]: finding a balance between bias and variance to minimize the mean squared error. When using simulations, we typically take m random independent samples $\hat{f}_1, \ldots, \hat{f}_m$ from the distribution of $\hat{f}$, and approximate the MSE, variance and bias as follows:

$$\begin{aligned}
\text{MSE}(\hat{f}(\mathbf{x}^*)) &\approx \frac{1}{m}\sum_{i=1}^{m}(\hat{f}_i(\mathbf{x}^*) - f(\mathbf{x}^*))^2; \\
\text{variance}(\hat{f}_i(\mathbf{x}^*)) &\approx \frac{1}{m}\sum_{i=1}^{m}\left(\hat{f}_i(\mathbf{x}^*) - \frac{1}{m}\sum_{j=1}^{m}(\hat{f}_j(\mathbf{x}^*))\right)^2; \\
\text{bias}(\hat{f}(\mathbf{x}^*)) &\approx \frac{1}{m}\sum_{i=1}^{m}(\hat{f}_i(\mathbf{x}^*) - f(\mathbf{x}^*)).
\end{aligned}$$

Suppose we set $k = n$ in the k-nearest neighbors method, then the prediction of any y^* will be the average of the y-values of the n nearest neighbors, which is the average of all y-values. In other words, every point gets the same prediction, so there is a small variance, but a very large bias (so still a big error). On the other hand, if we choose $k = 1$, then we have a small bias but a large variance (so a big error as well). As already mentioned, the goal is to find a balance between variance and bias (and determining the k-value as such), in order to minimize the MSE.

For s, the parameter that determines the minimal number of data points in a leaf (with random forests), we have something similar: for great values of s, we have a small variance and a large bias; for small values of this parameter, we have a small bias and a large variance. When determining the optimal parameter choice, we should thus again find a balance between bias and variance.

The number of trees M in a forest when using random forests, has influence on the MSE as well. A single tree is only trained on a single part of the training set, which causes the high

variance; by averaging multiple trees that are trained on different parts of the training set, the variance is reduced [12]. Therefore, M will in any case be chosen large enough, such that the variance caused by the number of trees is sufficiently small. This causes a small increase in the bias, but overall the MSE is decreased.

5.2 Simulation study

In order to measure the performance of the different methods, we need to test them within several fixed settings. As such, we generate data from a regression model $Y = f(\mathbf{X}) + \varepsilon$, where $\mathbf{X} \in \mathbb{R}^d$ with $d = 1, 2, \ldots, 10$ and $f : \mathbb{R}^d \to \mathbb{R}$ is one of:

- $f_1(x_1, \ldots, x_d) = \sin(2\pi x_1 - \pi)$: a function that essentially depends only on x_1, which means it will have redundancy for $d > 1$;
- $f_2(x_1, \ldots, x_d) = \prod_{i=1}^{d} \sin(2\pi x_i - \pi)$: a function that essentially depends on all covariates, with no redundancy.

We draw $n \in \{100, 300, 1000, 3000, 10000\}$ independent and identically distributed samples of $\mathbf{X} \sim \text{Unif}[0, 1]$ and $\varepsilon \sim \mathcal{N}(0, 1)$, namely $\mathbf{X}_1, \ldots, \mathbf{X}_n$ and $\varepsilon_1, \ldots, \varepsilon_n$. Then we will compute $Y_i = f(\mathbf{X}_i) + \varepsilon_i$ for $i = 1, \ldots, n$, and apply our methods, the k-nearest neighbors method and the random forest approach, to the generated samples $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, in order to estimate f. In other words, we choose a set of *test points* within the domain of f and estimate $f(\mathbf{x})$ for all test points $\mathbf{x}$. The set of test points is chosen as $\{(a, a, \ldots, a) \in \mathbb{R}^d : a = 0, 0.005, 0.01, \ldots, 0.995, 1\}$.

For the simulations, we use the statistical environment `R`, version 3.4.1. The software is freely available at `www.r-project.org`. The simulations are performed using a notebook with 8 GB Random Access Memory and an Intel i7-6700HQ Central Processing Unit, with a clock rate of 2.59 GHz.

5.3 Approximating mean squared error

In simulations as well as in practical situations, we need to determine optimal parameter values for the different methods. When doing simulations, we can use our knowledge of the function f in order to compute the errors of the predictions, and we can choose the parameter that leads to the smallest error. However, the magnitude of the MSE depends on the test point; for certain test points we can have a more accurate prediction than for others. It can thus be that one parameter value is a better choice for certain test points, but a worse choice for others. We would like to have a single number, capturing the overall error for all test points. If the test points are from a continuous domain, we can use the mean integrated squared error (MISE), which is given by integrating the pointwise MSE over the test domain D:

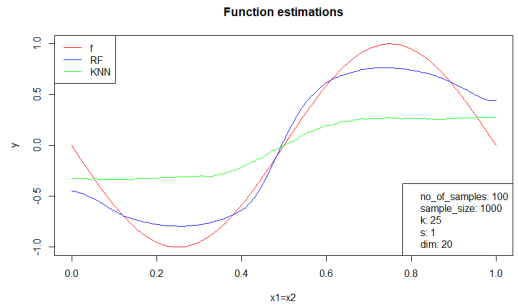
$$\begin{aligned} \text{MISE}(\hat{f}) &= \int_D \text{MSE}(\hat{f}(\mathbf{x}^*))\mathrm{d}\mathbf{x}^* = \mathbb{E}\left(\int_D (\hat{f}(x^*) - f(x^*))^2 \mathrm{d}\mathbf{x}^*\right) \\ &\approx \frac{1}{m}\sum_{i=1}^{m} \int_D (\hat{f}_i(x^*) - f(x^*))^2 \mathrm{d}\mathbf{x}^*, \end{aligned}$$

where $\hat{f}_1, \ldots, \hat{f}_m$ are independent samples from the distribution of $\hat{f}$, and the expectation is approximated by taking average over all m samples. Using the mean integrated squared error, we can objectively compare different parameter values, and pick the one with the smallest error.

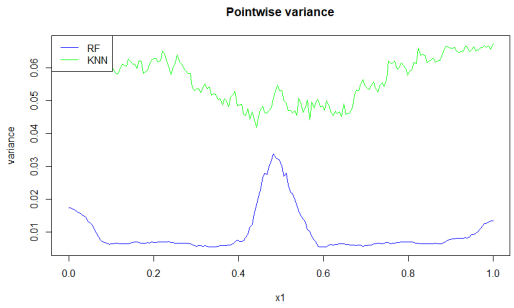
In real-world situations, when we do not have the knowledge of the function f, the optimal parameter value can be determined using predictive analytics. An example of such a technique is l-fold cross-validation, which will be discussed in Section 7.3.

5.4 Simulation results

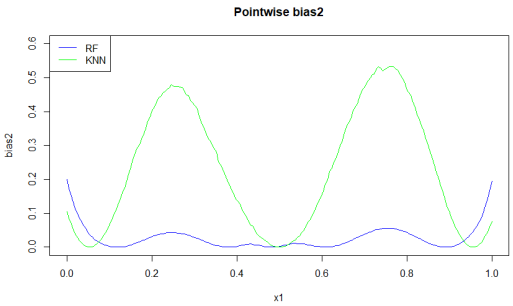
As could perhaps already be expected, the k-nearest neighbors method performs poorly when the number of dimensions d is large, especially when there is much redundancy. For example, when $d = 20$ and $f = f_1$, k-nearest neighbors is being outperformed by random forests, as can be seen in Figure 5.1. Obviously, since f_1 essentially depends only on x_1, the function and its estimations



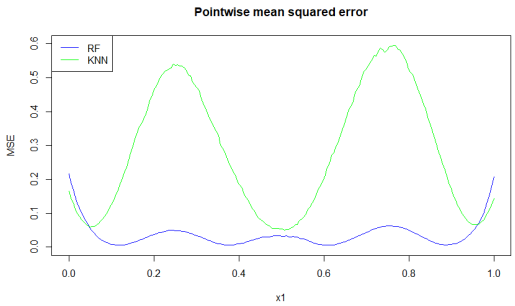
(a) Estimations



(b) Pointwise variances



(c) Pointwise squared biases

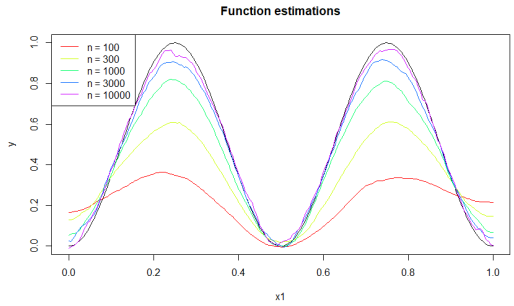


(d) Pointwise MSEs

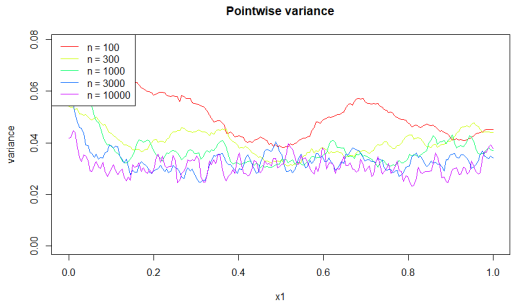
Figure 5.1: Estimations and errors when using k-nearest neighbors and random forests with $d = 2$, $f = f_1$ and $n = 1000$. The parameters k and s have been chosen optimally.

have been projected on the x_1-axis. Since we have an even higher number of dimensions in the precipitation data set and more redundancy, the k-nearest neighbors method will not be the solution.

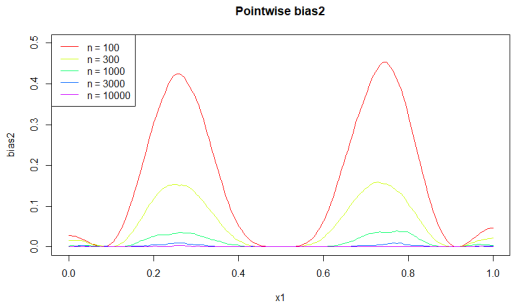
When using random forests, the bias is highly dominating the variance, even when choosing the smallest possible parameter value $s = 1$. This effect gets more severe as the dimension increases, but is already clearly visible for $d = 2$, $f = f_2$, see Figure 5.2. The function values are drawn and tested for points on the line $x_1 = x_2$, which has been projected onto the x_1-axis in the plots. It is proven that the bias vanishes as $n \to \infty$ [13], so it is perhaps not a surprise that the bias stops dominating for sufficiently large n. However, when slightly increasing the dimension to only $d = 5$, we can see in Figure 5.3 that even $n = 300000$ is not enough to bring the bias down to a reasonable level. Again, function values are drawn and tested for points on the line $x_1 = x_2 = \ldots = x_5$, which has been projected onto the x_1-axis in the plots. The sample sizes needed to stop the bias from dominating, are unfeasibly high. Regarding the fact that in our precipitation data set, the dimension is even much higher ($d \geq 84$), whereas the sample size is around $n = 15000$, random forests will not work either.



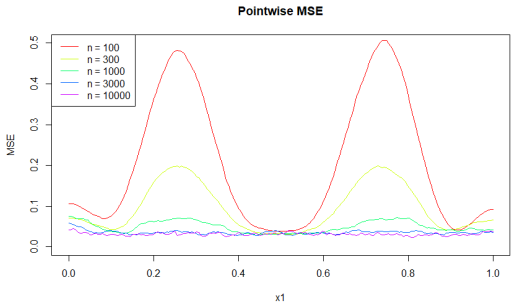
(a) Estimations



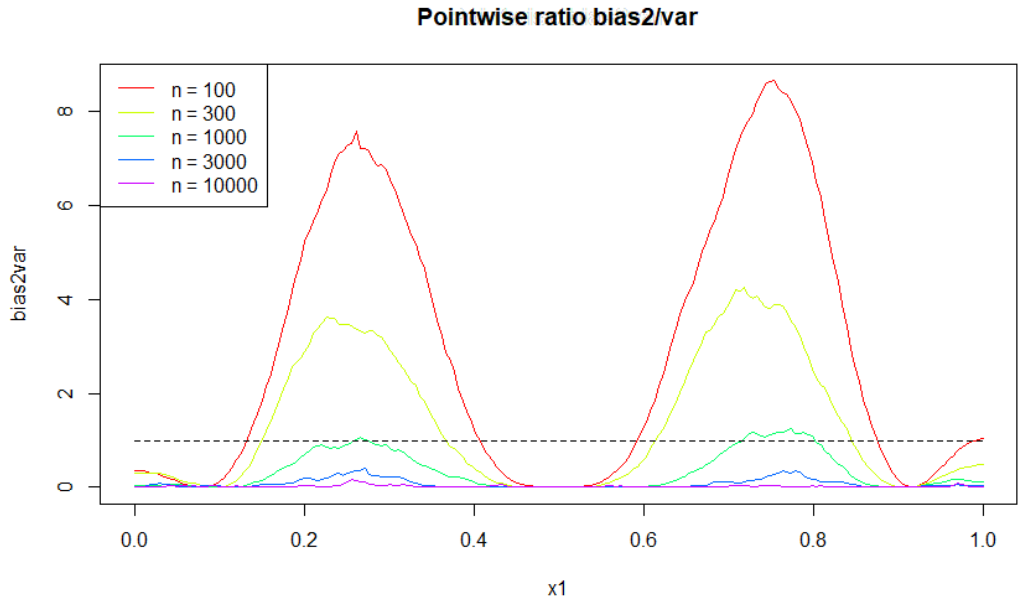
(b) Pointwise variances



(c) Pointwise squared biases

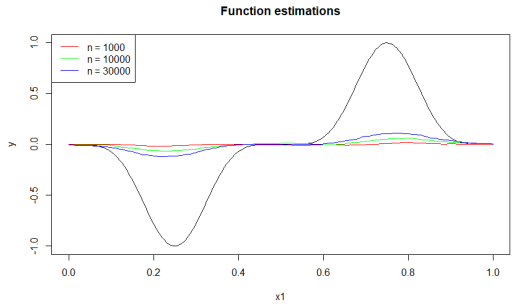


(d) Pointwise MSEs

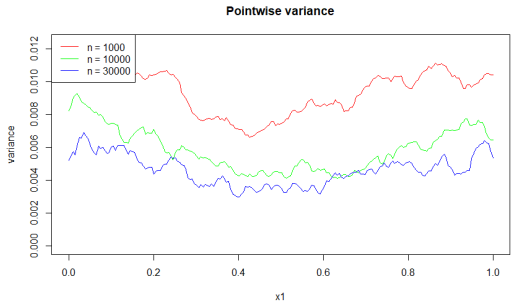


(e) Ratio between squared biases and variances

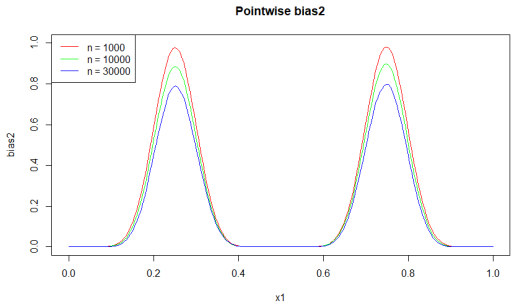
Figure 5.2: Estimations and errors when using random forests with $d = 2$, $f = f_2$, $s = 1$ and $n = 100, 300, 1000, 3000, 10000$. The bias is dominating over the variance for small n.



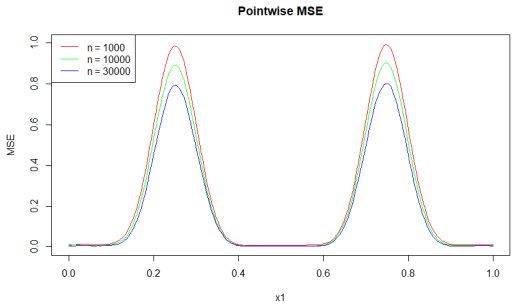
(a) Estimations



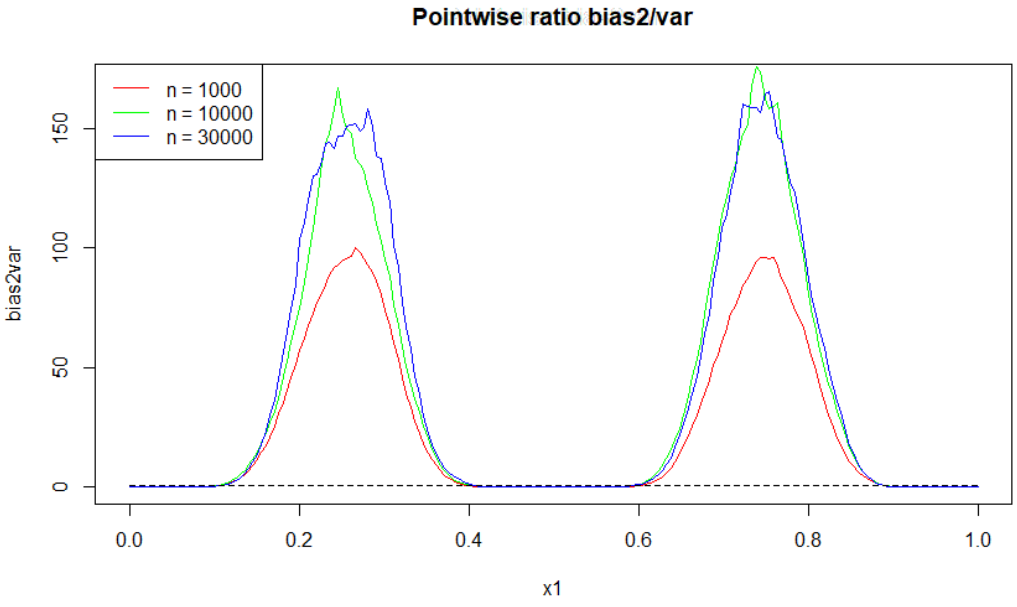
(b) Pointwise variances



(c) Pointwise squared biases



(d) Pointwise MSEs



(e) Ratio between squared biases and variances

Figure 5.3: Estimations and errors when using random forests with $d = 5$, $f = f_2$, $s = 1$ and $n = 1000, 10000, 30000$. The bias is dominating over the variance for all reasonable values of n.

Chapter 6

Proposed method: k-nearest forest neighbors

We saw in the previous chapter that both existing methods, k-nearest neighbors and random forests, are not good enough to be applied to the precipitation data set, as they both have their own drawbacks. Of course both methods do have advantages as well, so we can combine the best of their worlds into a new method, called k-nearest forest neighbors. In this chapter, we will explain this new method, and evaluate its performance again by simulation.

6.1 Explanation

In Chapter 4, we explained the random forests approach. For a certain test point, the prediction of its y-values is done as follows. The forest consists of several trees, and for each of them, we see in which leaf the test point ends up. The prediction for the tree is then the average of the data points in the leaf; the ultimate prediction is the average of the prediction of all trees in the forest. Certain data points end up in the same leaf as the test point more often than others, and have therefore more influence on the prediction for the test point's y-value. We could also say that these data points have more *weight*. Often, there are many data points with nonzero weight, causing a dominating bias and worse predictions. This is illustrated in the four left plots of Figure 6.1, where a simulation has been done using random forests with $n = 5000$, $d = 2$ and $f = f_1$. We see that even for the smallest possible parameter value $s = 1$, there is a big amount of data points having non-zero weight, and this gets even worse for larger values of s. This is exactly the problem that our proposed method, k-nearest forest neighbors, attempts to fix.

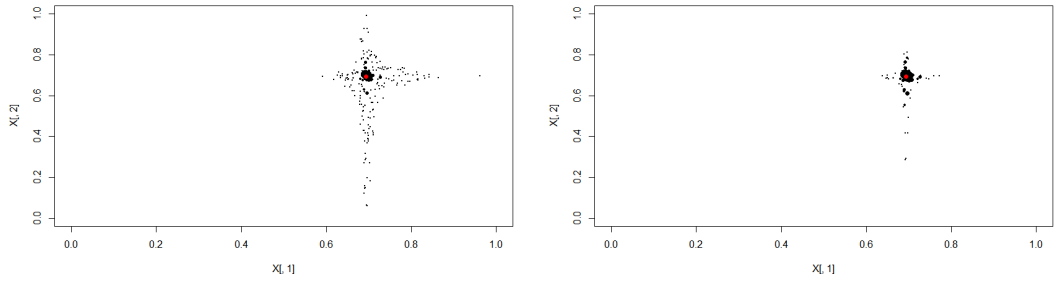
According to equation (4.6), we have $\hat{y}^* = \sum_{i=1}^{n} w_i(\mathbf{x}^*)Y_i$, where $w_i(\mathbf{x}^*)$ is the weight of data point $\mathbf{X}_i$ when predicting y^*. Now the intuition behind k-nearest forest neighbors is simple. For each test point, we take the K [1] data points that have the highest weight, and set the others to zero. Thus, let $K \in \mathbb{Z}$ with $K \leq n$ be a new parameter and let W_K be the set consisting of the K data points that have the largest weight. Now we assign new weights $w_i^{(K)}(\mathbf{x}^*)$ to each data point $\mathbf{x}_i$ as follows:

$$w_i^{(K)}(\mathbf{x}^*) = \begin{cases} w_i(\mathbf{x}^*) \Big/ \sum_{\mathbf{x}_j \in W_K} w_j(\mathbf{x}^*), & \text{if } \mathbf{x}_i \in W_K; \\ 0, & \text{otherwise.} \end{cases}$$

In other words, we take the K largest weights and scale them so that their sum equals 1, and set all other weights to zero. This reduces the number of data points with nonzero weight, and thus reduces the bias as well. Now our prediction is computed using the new weights,

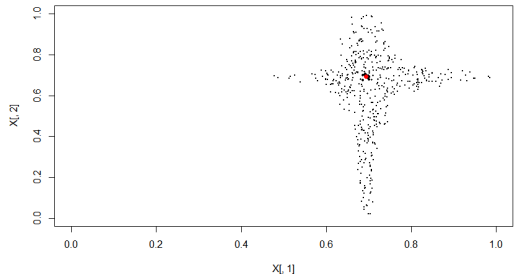
$$\hat{y} = \sum_{i=1}^{n} w_i^{(K)}(\mathbf{x}^*)Y_i = \sum_{\mathbf{x}_i \in W_K} w_i^{(K)}(\mathbf{x}^*)Y_i,$$

[1] We will use a capital K instead of k, in order not to get confused with the k-nearest neighbors method.

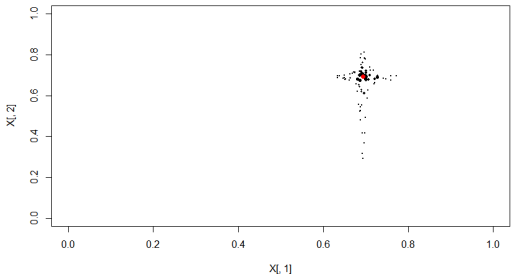


(a) $s = 1$

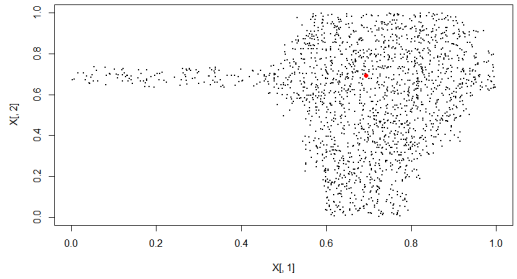
(b) $s = 1$, $K = 50$



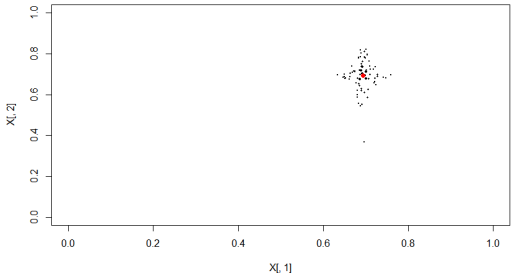
(c) $s = 10$



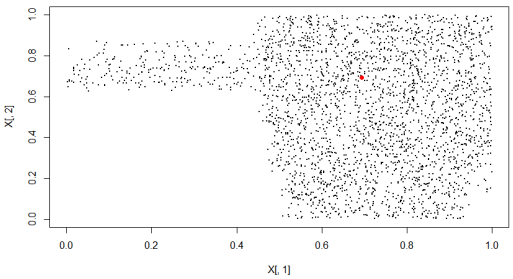
(d) $s = 10$, $K = 70$



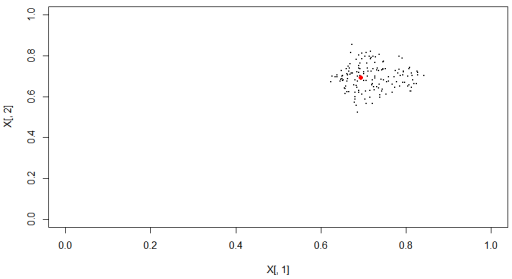
(e) $s = 100$



(f) $s = 100$, $K = 70$



(g) $s = 300$



(h) $s = 300$, $K = 150$

Figure 6.1: The red point is the (fixed) test point $\mathbf{x}^*$. For the plots at the left, the black points are all data points $\mathbf{X}_i$ having nonzero weight $w_i(\mathbf{x}^*)$ when predicting y^* using random forests, with different values of s; for the plots at the right, the black points are the data points having non-zero weight $w_i^K(\mathbf{x}^*)$ when predicting y^* using k-nearest forest neighbors, with parameter K.

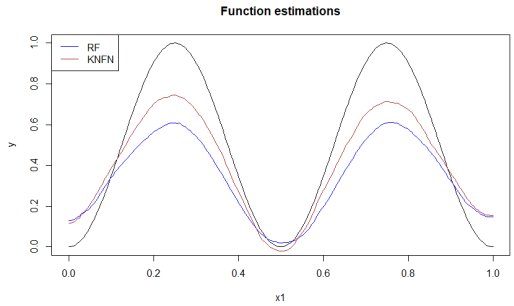
as a weighted average over the y-values of the K elements in W_K. In Figure 6.1 (the four plots at the right), we can see that using k-nearest forest neighbors indeed helps reducing the number of data points having non-zero weight. We will see in the next section whether this new method also helps reducing the bias.

6.2 Performance evaluation using simulation

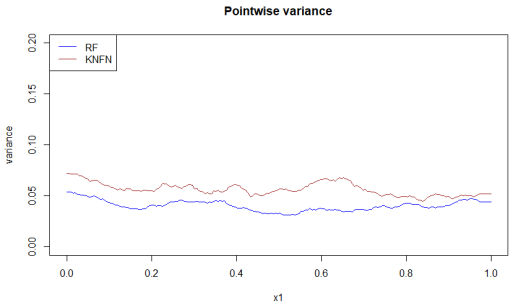
In Figure 5.2, we saw the results when using random forests for $d = 2$ and $f = f_2$. When using the k-nearest forest neighbors method on the exact same data set, the results improve for every value of n: the bias heavily decreases whereas the variance increases insignificantly, causing a significant drop in the ratio $\frac{\text{bias}^2}{\text{variance}}$. We will now focus on comparing $n = 300$, for the results with other sample sizes, the reader is referred to Figure B.1 in the appendix. The results obtained using the two methods with $n = 300$, are given in Figure 6.2. We see that the bias has been significantly decreased by using the k-nearest forest neighbors method, while the variance has hardly increased. This is especially visible in parts (b), (c) and (e) the figure. Therefore, the sample size needed for the bias to be sufficiently small, decreases when using k-nearest forest neighbors.

We also saw that random forest performed poorly for $d = 5$ and $f = f_2$, for reasonable sample sizes. When applying the k-nearest forest neighbors approach to the same data set, we again see an improvement for each value of n: again we can see the decreasing bias, whereas the variance does not increase significantly. The estimations thus improve, but are unfortunately still poor. We will show the results for $n = 10000$ in Figure 6.3; the results obtained for other sample sizes can be found in Figure B.2 of the appendix. Again, we see that the sample size needed for the bias to be sufficiently small, decreases when using k-nearest forest neighbors.

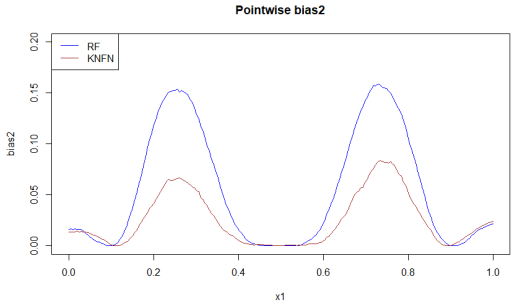
We will not show here that the k-nearest forest neighbors method also outperforms k-nearest neighbors, interested readers are referred to Figure B.3 of the appendix, where the k-nearest forest neighbors estimation has been added to Figure 5.1. To see whether the k-nearest forest neighbors method really works, however, we need to apply it to the precipitation data set, which will be done in the next chapter.



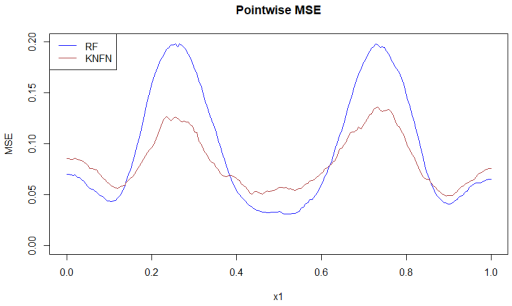
(a) Estimations



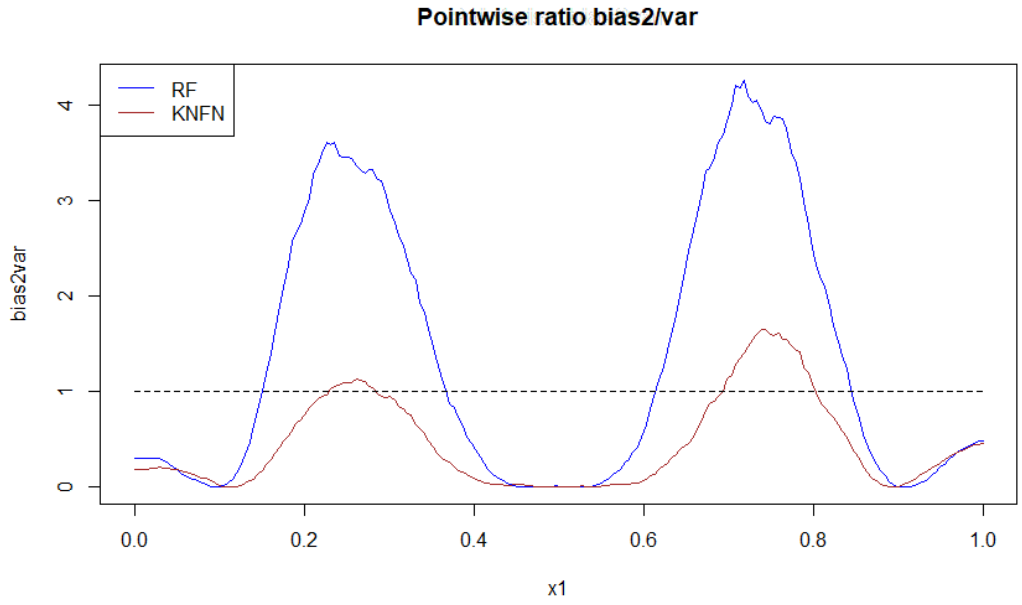
(b) Pointwise variances



(c) Pointwise squared biases

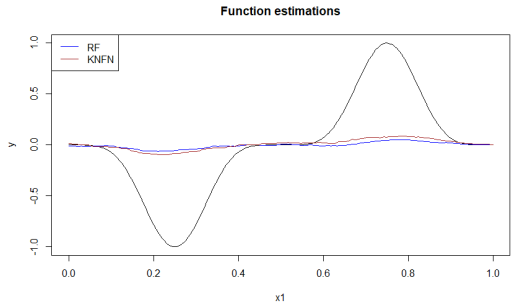


(d) Pointwise MSEs

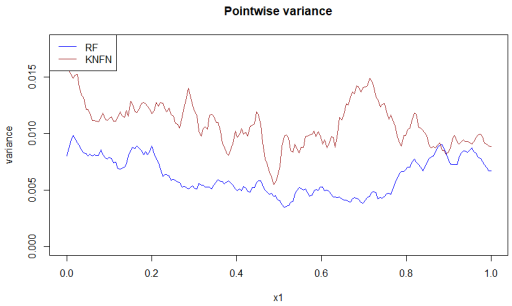


(e) Ratio between squared biases and variances

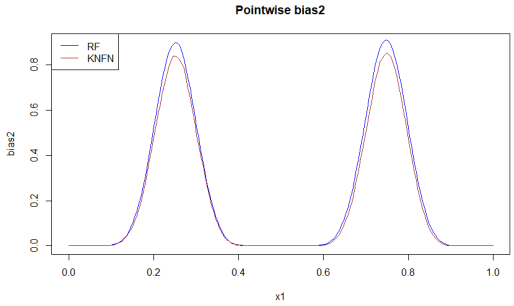
Figure 6.2: Estimations and errors when using random forest and k-nearest forest neighbors with $d = 2$, $f = f_2$ and $n = 300$. The bias domination has been greatly reduced by using k-nearest forest neighbors.



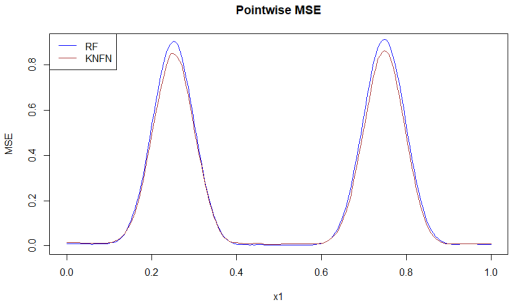
(a) Estimations



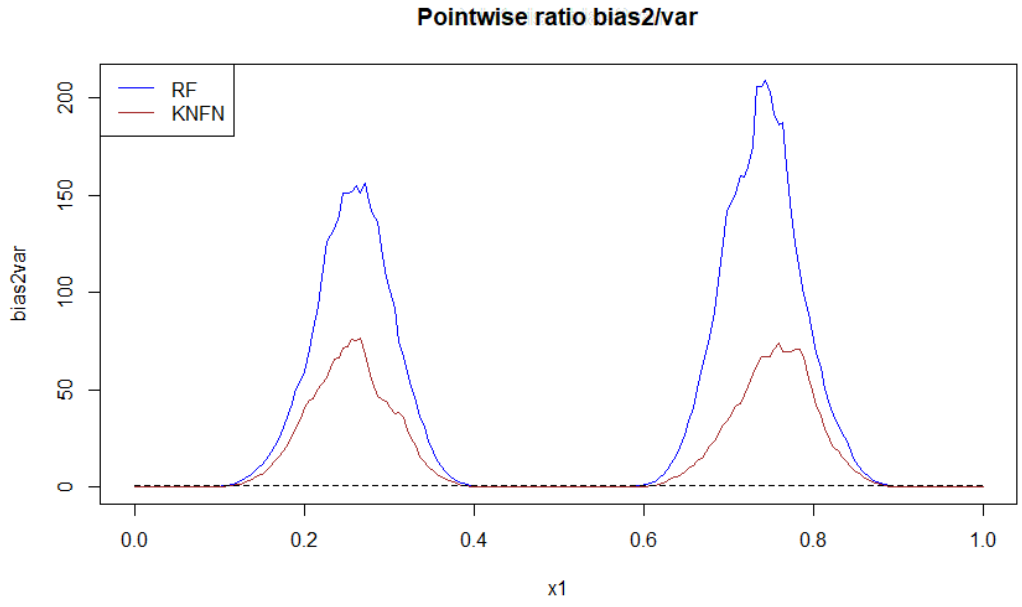
(b) Pointwise variances



(c) Pointwise squared biases



(d) Pointwise MSEs



(e) Ratio between squared biases and variances

Figure 6.3: Estimations and errors when using random forest and k-nearest forest neighbors with $d = 5$, $f = f_2$ and $n = 10000$. The bias domination has been greatly reduced by using k-nearest forest neighbors, but the estimations are still poor.

Chapter 7

Application on precipitation prediction

7.1 Adding new covariates to the data set

The precipitation data set from the KNMI was explained in Chapter 2. Apart from the 84 covariates resulting from the HARMONIE model, containing forecasts of precipitation, atmospheric stability, airflows and precipitable water (among others), we saw in Section 2.2 that region, date and time can be correlated to the precipitation as well. Namely, it seems that there is more extreme precipitation in non-coastal areas, in summer months and by the end of the afternoon. We can therefore add information about the region, date and time to the 84 already existing covariates.

Region

Since the regions are in a two-dimensional plane, we need two numbers to reflect their distances correctly. In the current situation, region 9 is closer to region 12 than to region 10, whereas the numbers 9 and 10 are closer to each other than 9 and 12, which is undesirable. Hence, we will introduce two covariates:

$$\texttt{region\_x} = [(\texttt{region} - 1) \mod 3] + 1, \text{ and } \texttt{region\_y} = \lceil \texttt{region}/3 \rceil,$$

which means regions $1, 2, 3, 4, \ldots, 12$ become $(1,1), (2,1), (3,1), (1,2), \ldots, (3,4)$, respectively.

Date

For the date, we just let April 1st be day number 1, April 2nd day number 2, until October 31st, which is then day number 214. From the violin plots in Section 2.2, it followed that the weather in April was not comparable to the weather in October, so it makes no sense to define some periodic metric (where April 1st is close to October 31st). Note that the year is not being taken into account, which means April 3rd, 2011 and April 3rd, 2013 will both have $\texttt{date} = 3$. We also expect that the weathers on the same day in different years will have some correlation.

Time

For the time, it does make sense to have periodicity, since the time window 18:00-24:00 is close to the time window 0:00-6:00 of the next day. However, this is not as simple as it appeared at first sight. Choosing a single sinusoid results in $-1, 0, 1, 0$ for the time windows 0-6, 6-12, 12-18 and 18-24, respectively. This gives 6-12 and 18-24 the same metric, which is clearly not what we want. Shifting the sinusoid does not solve this problem.

Another option could be combining date and time, and adding time as a fraction to the date. April 1st 18:00-24:00 then becomes 1.75, and April 2nd 0:00-6:00 would be 2.00. However, the

time was introduced to reflect that, for instance, April 5th 12:00-18:00 and May 2nd 12:00-18:00 should be comparable, and using such a metric throws away the link between those two (as their distance becomes large). Keeping only the fraction, results in a large distance between the time windows 18-24 (0.75) and 0-6 (0.00), so this is not a good solution either.

The solution has been found in adding another metric for time. Besides $-1, 0, 1, 0$, we have $0, 1, 0, -1$ as well. This means the (Euclidean) distance between two neighboring time windows is $\sqrt{2}$, and the distance between two opposing time windows is 2. One could compare these metrics to the (x, y) coordinates of the points where the unit circle intersects the axes. Each of these four intersection points $(\pm 1, 0), (0, \pm 1)$ then represents a time window.

7.2 Selecting the most predictive covariates

After adding these covariates about region, date and time, we have a total of 89 covariates in the data set. We already saw that all methods have difficulties dealing with a large number of covariates; even the k-nearest forest neighbors method did not significantly improve upon using random forests, for sufficiently many covariates with a fixed sample size. Since the sample size in the data set is fixed, we need to limit the number of covariates. We will do so by selecting only $D << 89$ of them to train the models on, which means the precipitation predictions will only be based on these covariates. It goes without saying that we want to choose the D most predictive ones. When building a random forest, the data points are split across the most predictive covariates, which means the splits give an indication of a covariates' predictive value. A technique for ranking the covariates according to their importance, is described in [8], and implemented in the `variable_importance` function of the `grf` package [17]. This function has two parameters `decay.exponent` and `max.depth`: the latter specifies the maximum depth of the splits to consider, since deeper splits give less information about predictive value. Hence, deeper splits should also be given less weight in determining the variable importance, and this is exactly what the `decay.exponent` parameter is controlling. After having tried different values, it turned out that `decay.exponent` $= 2$ and `max.depth` $= 4$ gave the best results.

The covariate importances of all 89 covariates are given in Tables B.1 and B.2 in the appendix. The twenty most predictive covariates per lead time are given in Table 7.1. The physical meaning of the covariates is beyond the scope of this thesis. However, it is worth saying that the covariates ending on a `+` and `-` are hourly maxima and minima, respectively. The suffix _x`m` means that the measurement has been taken x meters above the ground, and _y`mb` means that the measurement was taken with an air pressure of y millibars, which is also an indication of the altitude, as air gets thinner further above the ground. Generally, a covariate having a person's name, such as `Boyden` or `Rackliff`, is about atmosphere stability. The covariate `Rain` is the HARMONIE model precipitation prediction.

We will now focus on lead time number 3, which is 18 hours in advance. The simple reason for this choice is that earlier lead times are hard to predict, whereas later lead times are more easily predictable; lead time number 3 is thus a good balance in terms of predictability. We can build a random forest using only the twenty most predictive covariates, and retrieve the variable importances again. This turns out to change the order of importance, as can be seen in table 7.2. We see that `Rain_0m.+` makes a giant leap forward, whereas `SWEAT_0m.+` suffers from a free fall. We will see that in general, taking the top-D from the second list (best of 20) gives better results than taking the top-D from the first one. It is also intuitive that a covariate such as `Rain_0m.+` has a high predictive value. Thus, our top-D will be taken from the second list, i.e. the "best D of 20".

7.3 Validation setting

In section 5.3, we discussed how to determine error measures, such as the MSE, when having knowledge of the underlying regression function f. When applying the methods to the precipitation data set, however, we do not have this knowledge. There are several ways of still approximating the errors, and one of those is l-fold cross-validation. Being able to approximate,

for example, the MSE, not only allows us to estimate a method's predictive accuracy, but also to select the optimal parameter value of a certain method, simply by selecting the parameter value leading to the smallest error.

For validation, we cannot use the same data points for estimating the function, as well as judging the predictive accuracy of the estimated function. Otherwise, we would favor strange functions that perfectly fit out training data. Hence, we partition the data set into two equally sized sets: one used for estimating (the training set), and the other for testing this estimation (the estimation set or test set), with which we estimate a measure of performance, for instance the MSE. The idea of *cross*-validation is that we could also use the second set for estimating and the first one for testing the estimation, and thus get another value for predictive performance. The performance is then calculated by averaging these two values.

This exactly describes 2-fold cross-validation. We could generalize this into l-fold cross-validation, by randomly partitioning the n data points into l sets, called *folds*, having roughly equal size n/l. Then we use all folds except the first one in order to make an estimation, and evaluate the predictive performance using the first fold. After this, we use all folds except the second one to make (another) estimation, and evaluate its performance using the second fold. We repeat this process for all l folds, and calculate the performance by averaging over all these l values [14]. The parameter value (e.g. s or k) for which we have the best performance (least error), is the value that we eventually choose. In practice, 10-fold cross-validation is often chosen [15].

Since we have comparable data about three years, it is natural to use 3-fold cross-validation in our case, where the folds are the different years. As an error measure, we use the root mean squared error (RMSE), which is the square root of the MSE. Since the root function is strictly increasing, this preserves the ordering in terms of predictive accuracy. The advantage of using RMSE, is that its unit is the same as the precipitation unit (millimeters per hour), which makes the numbers more easily interpretable. The (high-level) pseudo-code for the cross-validation is given in Algorithm 1, the R code documentation can be found in Appendix C. We can repeat

Algorithm 1 Pseudo-code for 3-fold cross-validation on precipitation data.

```
1: procedure CrossValidation(model, data, t (lead time number), top-D covariates for lead time number t)
2:     add new covariates (region, date, time) to data
3:     for i = 2010, 2011, 2013 do
4:         cov ← top-D covariates for lead time t
5:         training_data ← data of the covariates cov, all years except i
6:         train model on training_data
7:         for each data point p in year i do
8:             predict hrly.obs of p using model and compute MSE using its value in data
9:         e[i] ← average of the MSEs over all p
10:    MSE[t] ← average of e[i] over i = 2010, 2011, 2013
11:    return √MSE[t]
```

this procedure for different parameter values and pick the best one, i.e. the one with the smallest RMSE. This way, we can obtain an average root mean squared error for each method (k-nearest neighbors, random forest and k-nearest forest neighbors) and each lead time. We can then determine which of the methods performs best for a certain lead time, and see how early (in which lead time) extreme precipitation can be predicted accurately enough.

Number	LT1	LT2	LT3	LT4	LT5	LT6	LT7
1	Lifted_Om.-	VWND_700mb.-	Boyden_Om.+	Boyden_Om.+	Boyden_Om.+	Sur_CI_Om.-	Boyden_Om.+
2	Helicity_Om.-	LCL_Om.-	Lifted_Om.-	Rackliff_Om.-	Boyden_Om.-	Tot_1e_Om.-	Helicity_Om.+
3	LFC_Om.-	Helicity_Om.+	VWND_700mb.-	Helicity_Om.-	Rackliff_Om.-	Rackliff_Om.+	Tot_1e_Om.+
4	Lifted_Om.+	Lifted_Om.-	VWND_700mb.+	UWND_700mb.-	Lifted_Om.+	date	Helicity_Om.-
5	Boyden_Om.+	Boyden_Om.+	Rackliff_Om.-	Lifted_Om.+	LCL_Om.-	UWND_700mb.-	UWND_700mb.+
6	Rackliff_Om.-	SWEAT_Om.+	Lifted_Om.+	Sur_CI_Om.-	date	Lifted_Om.+	SWEAT_Om.+
7	Tot_1e_Om.+	Tot_1e_Om.-	SWEAT_Om.+	Lifted_Om.-	UWND_700mb.+	Tot_1e_Om.+	VWND_700mb.-
8	Tot_1e_Om.-	Helicity_Om.-	UWND_700mb.-	Boyden_Om.-	Helicity_Om.-	SWEAT_Om.+	LFC_Om.+
9	UWND_700mb.-	Sur_CI_Om.-	VWND_700mb.+	Rackliff_Om.+	Rackliff_Om.+	Boyden_Om.-	Lifted_Om.+
10	LCL_Om.-	LCL_Om.+	LCL_Om.-	Helicity_Om.+	Sur_CI_Om.-	Rackliff_Om.-	Sur_CI_Om.-
11	LCL_Om.+	UWND_700mb.-	UWND_700mb.+	VWND_700mb.-	SWEAT_Om.+	LCL_Om.-	Rackliff_Om.-
12	UWND_700mb.+	Boyden_Om.-	Helicity_Om.+	Rackliff_Om.+	VWND_700mb.-	UWND_700mb.+	Rackliff_Om.+
13	Sur_CI_Om.-	Rackliff_Om.-	Rain_Om.+	LFC_Om.-	Helicity_Om.+	VWND_700mb.-	UWND_700mb.-
14	SWEAT_sin_Om.+	SWEAT_sin_Om.+	Helicity_Om.-	SWEAT_Om.+	LFC_Om.+	Helicity_Om.-	Boyden_Om.-
15	SWEAT_Om.+	Rackliff_Om.+	Tot_1e_Om.+	SWEAT_sin_Om.+	UWND_700mb.-	Boyden_Om.+	Tot_1e_Om.-
16	Rackliff_Om.+	Tot_1e_Om.+	LFC_Om.-	Tot_1e_Om.+	Tot_1e_Om.+	LFC_Om.-	LFC_Om.-
17	Rain_Om.+	Lifted_Om.+	Boyden_Om.-	UWND_700mb.+	SWEAT_sin_Om.+	Helicity_Om.+	Lifted_Om.-
18	VWND_700mb.-	UWND_700mb.+	Tot_1e_Om.-	LCL_Om.-	LFC_Om.-	SWEAT_sin_Om.+	LCL_Om.-
19	Boyden_Om.-	Rain_Om.+	SWEAT_sin_Om.+	Tot_1e_Om.-	Tot_1e_Om.-	Rain_Om.+	Rain_Om.+
20	Helicity_Om.+	VWND_700mb.+	Rackliff_Om.+	Rain_Om.+	VWND_700mb.+	LFC_Om.+	Fateev_Om.+

Table 7.1: Top-20 of most predictive covariates per lead time.

Number	best of all	best of 20
1	Boyden_0m.+	Boyden_0m.+ (− 0)
2	Lifted_0m.-	Lifted_0m.- (− 0)
3	VWND_700mb.-	Rain_0m.+ (↑ 10)
4	VWND_700mb.+	VWND_700mb.- (↓ 1)
5	Rackliff_0m.-	VWND_700mb.+ (↓ 1)
6	Lifted_0m.+	Sur_CI_0m.- (↑ 3)
7	SWEAT_0m.+	Helicity_0m.- (↑ 7)
8	UWND_700mb.-	LCL_0m.- (↑ 2)
9	Sur_CI_0m.-	Helicity_0m.+ (↑ 3)
10	LCL_0m.-	UWND_700mb.+ (↑ 1)
11	UWND_700mb.+	LFC_0m.- (↑ 5)
12	Helicity_0m.+	Rackliff_0m.- (↓ 7)
13	Rain_0m.+	Lifted_0m.+ (↓ 7)
14	Helicity_0m.-	UWND_700mb.- (↓ 6)
15	Tot_1e_0m.+	Boyden_0m.- (↑ 2)
16	LFC_0m.-	Tot_1e_0m.+ (↓ 1)
17	Boyden_0m.-	Rackliff_0m.+ (↑ 3)
18	Tot_1e_0m.-	SWEAT_0m.+ (↓ 11)
19	SWEAT_sin_0m.+	Tot_1e_0m.- (↓ 1)
20	Rackliff_0m.+	SWEAT_sin_0m.+ (↓ 1)

Table 7.2: Top-20 of most predictive covariates, for lead time number 3.

7.4 Validation results

We computed the RMSE for different methods: the NWP prediction, k-nearest neighbors method, random forests approach, and k-nearest forest neighbors, with different parameter values. The results are given in Table 7.3.

D	method	best parameter	RMSE
–	NWP	–	3.006
89	knn	$k = 31$	3.072
89	rf	$s = 1$	2.891
89	knfn	$s = 1, K = 205$	2.888
20	knn	$k = 59$	3.291
20	rf	$s = 1$	2.903
20	knfn	$s = 1, K = 609$	2.885
5	knn	$k = 64$	2.957
5	rf	$s = 1$	2.931
5	knfn	$s = 1, K = 502$	2.918
3	knn	$k = 15$	3.118
3	rf	$s = 1$	2.957
3	knfn	$s = 1, K = 541$	2.927

Table 7.3: RMSEs after applying different methods on precipitation data set, for lead time number 3. The abbreviations knn, rf and knfn stand for k-nearest neighbors, random forests, and k-nearest forest neighbors, respectively.

As expected, we see that random forests and k-nearest forest neighbors perform better than k-nearest neighbors, regardless of the number of covariates. We can also see that in general, taking fewer covariates (smaller D) results in worse predictions, which is very intuitive since selecting few covariates means throwing away much information. There is one notable exception: when using k-nearest neighbors, we get significantly better results with $D = 5$ than $D = 20$. This can be explained as follows: we saw in Section 5.4 that k-nearest neighbors has a lot of trouble when there is a large number of covariates, due to the redundancy. Namely, it does not know which covariates are predictive, and which are not. In this case, however, the D most predictive covariates have already been selected by random forests, and this apparently helps the k-nearest neighbors method to perform better. When using $D = 3$, the performance gets worse again, as valuable information (the fourth and fifth covariate) has been thrown away.

It is also good to see that k-nearest forest neighbors and random forests give better predictions than the NWP model. Moreover, our newly developed method, k-nearest forest neighbors, has more predictive accuracy than the other approaches; the difference with random forests is, however, not significantly large. Analogous to the simulation study, we see that the improvement upon random forests gets more significant as the number of covariates decreases. We also saw in the simulation study that the improvement gets more significant when the sample size increases, which suggests that the insignificance of the improvement is likely due to the limited sample size in the precipitation data set as well.

Chapter 8

Conclusion

In this report, we started by doing a simulation study of two existing non-parametric regression methods, namely k-nearest neighbors and random forests. This study gave much insight into how these existing methods work. It turned out that the k-nearest neighbors method was not able to deal with the redundancy present in the simulated data, and was thus outperformed by random forests. The random forests approach itself had a problem as well: the dominating bias, causing a large mean squared error in the predictions.

We attempted to solve this problem by introducing a new approach called k-nearest forest neighbors. When applying this method to the simulated data, we indeed saw that it improved upon random forests. When using a small number of covariates, this improvement was more significant than in a model with many covariates.

Since the KNMI precipitation data set contained 89 covariates, we decided to select the most predictive D of them, in order to limit the number of covariates. The predictive value of a covariate was estimated by growing a forest on the precipitation data.

We saw that both k-nearest forest neighbors and random forests give better predictions on the precipitation data set than the NWP model, which means that regression is a good post-processing method for predicting extreme precipitation. Furthermore, the k-nearest forest neighbors method performed better than random forests, in terms of root mean squared error. The difference between those methods was, however, not really significant, partly due to the limited available sample size in the precipitation data set. Analogous to the simulated data, we saw the improvement getting more significant for smaller values of D. It might therefore be safe to conclude that the k-nearest forest neighbors method that we developed, is promising and has the potential of significantly improving upon random forests.

Chapter 9

Further research

The k-nearest forest neighbors method that we developed, still leaves room for improvement. Below are listed some possible areas for further research, which will hopefully lead to the k-nearest forest neighbors method ultimately outperforming random forests significantly.

Firstly, when doing a simulation study, more different cases can be considered. In this thesis, we assumed a uniform distribution of the covariates, and normality of the noise term. If different distributions are used, the performance of the methods can possibly change.

Besides, the validation of the method can be done more theoretically, instead of only using simulations. This kind of validation could give more insight into the mathematical side of the method. It could, for example, make clear that a slight modification of the method can improve its performance.

Furthermore, since the bias and variance are dependent on the test point $\mathbf{x}^*$, the optimal parameter value will depend on $\mathbf{x}^*$, too. Choosing a constant parameter value, as was done in this thesis, likely causes sub-optimality for certain test points. The mean squared error could thus be further reduced when choosing different parameter values for different test points.

Last but not least, since our goal is forecasting *extreme* precipitation, it would be interesting to see if the models can correctly predict whether the hourly precipitation will exceed P millimeters. Different values of P could then be chosen; for example, $P = 30$ would indicate whether the models justly issue a code yellow warning. The RMSEs that we have computed in this thesis, only tell us something about averaged errors, without giving information about the ability to correctly forecast extreme numbers.

Bibliography

[1] *Uitleg over KNMI Waarschuwingen.* `https://www.knmi.nl/kennis-en-datacentrum/uitleg/knmi-waarschuwingen` [Accessed on July 12, 2018].

[2] J.J. Velthoen. Non-parametric extreme quantile estimation for the common shaped tail model. MSc thesis, Delft University of Technology, Oct 2016.

[3] P. Baas and H.W. van den Brink. The added value of the high-resolution HARMONIE runs for deriving the HBCs. Technical report, Royal Netherlands Meteorological Institute, 2014.

[4] J.L. Hintze and R.D. Nelson. Violin plots: A box plot-density trace synergism. *The American Statistician*, 52(2):181–184, May 1998.

[5] A.M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes.* F. Didot, Paris, 1805.

[6] P.J. Green and B.W. Silverman. *Nonparametric Regression and Generalized Linear Models.* Chapman and Hall, New York, 1993.

[7] N.S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[8] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[9] S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 2017.

[10] Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. Technical Report 1055, University of Wisconsin, 2002.

[11] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, 2009.

[13] S. Wager. Asymptotic theory for random forests. Technical report, Stanford University, 2016.

[14] Y. Yang. Consistency of cross validation for comparing regression procedures. *The Annals of Statistics*, 35(6):2450–2473, 2007.

[15] G.J. McLachlan, K.A. Do, and C. Ambroise. *Analyzing Microarray Gene Expression Data.* John Wiley & Sons, 2004.

[16] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap.* Chapman & Hall/CRC, Boca Raton, 1993.

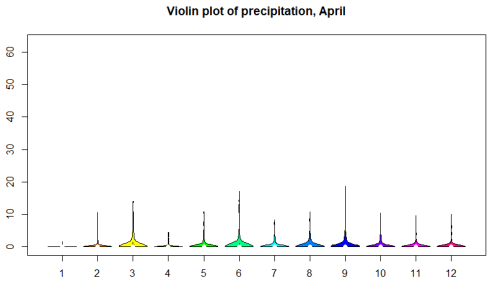
[17] J. Tibshirani, S. Athey, and S. Wager. *Package 'grf'*, May 2018.

Appendix A

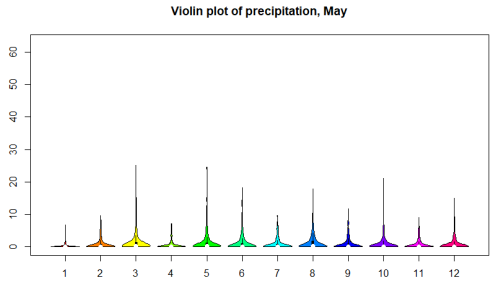
Additional data properties

As mentioned in Section 2.2, we would include violin plots for the twelve regions, split by month; violin plots for the seven months, split by region; violin plots for the twelve regions, split by time window; and violin plots for the four time windows, split by region. This is to gain more insight into the combinations of regions, months and time windows in which there is the most extreme precipitation. Judging from Figure A.1, it seems that in months like May and September, there is quite much precipitation in coastal areas such as regions 7 and 10. In summer months, however, there is relatively little precipitation. In Figure A.2, we can see that the peaks for the coastal areas are relatively late (after summer), whereas most areas have their peaks in summer, as expected. Furthermore, it is notable that there is a huge peak in September in region 10, as well as in August in region 3.

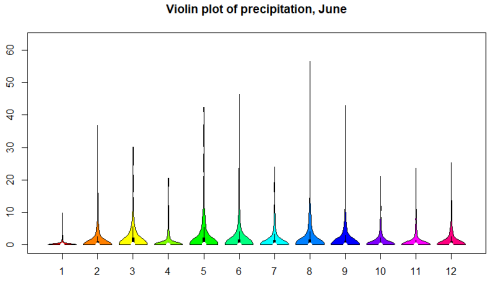
The violin plots for the twelve regions, split by time frame; and for the four time windows, split by region, are given in figures A.3 and A.4, respectively. It seems that there is the least extreme precipitation during the night (0-6 UTC), and as expected, the most extreme precipitation happens at the end of the day (12-18 UTC in the southeast, 18-24 UTC in the rest of the country). In region 3, however, it seems that the most extreme precipitation occurs in the morning, judging from A.3(b) and A.4(c). Nevertheless, generally the last two time frames of a day are most interesting in terms of extreme precipitation.



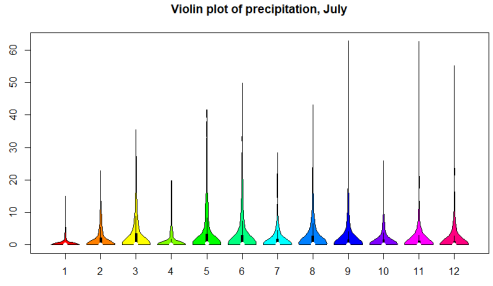
(a) April



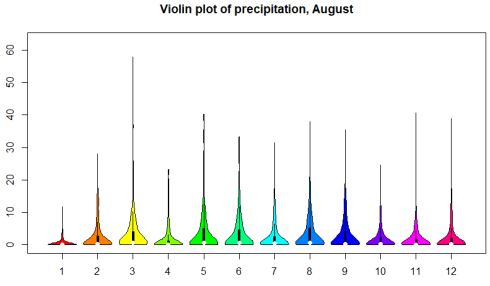
(b) May



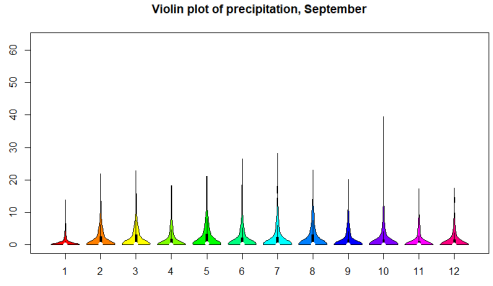
(c) June



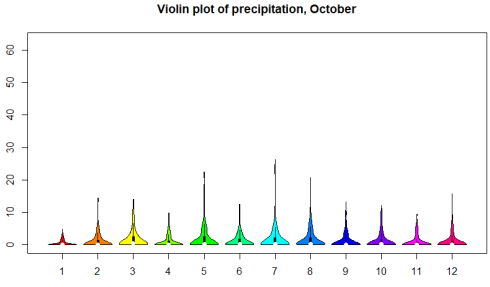
(d) July



(e) August

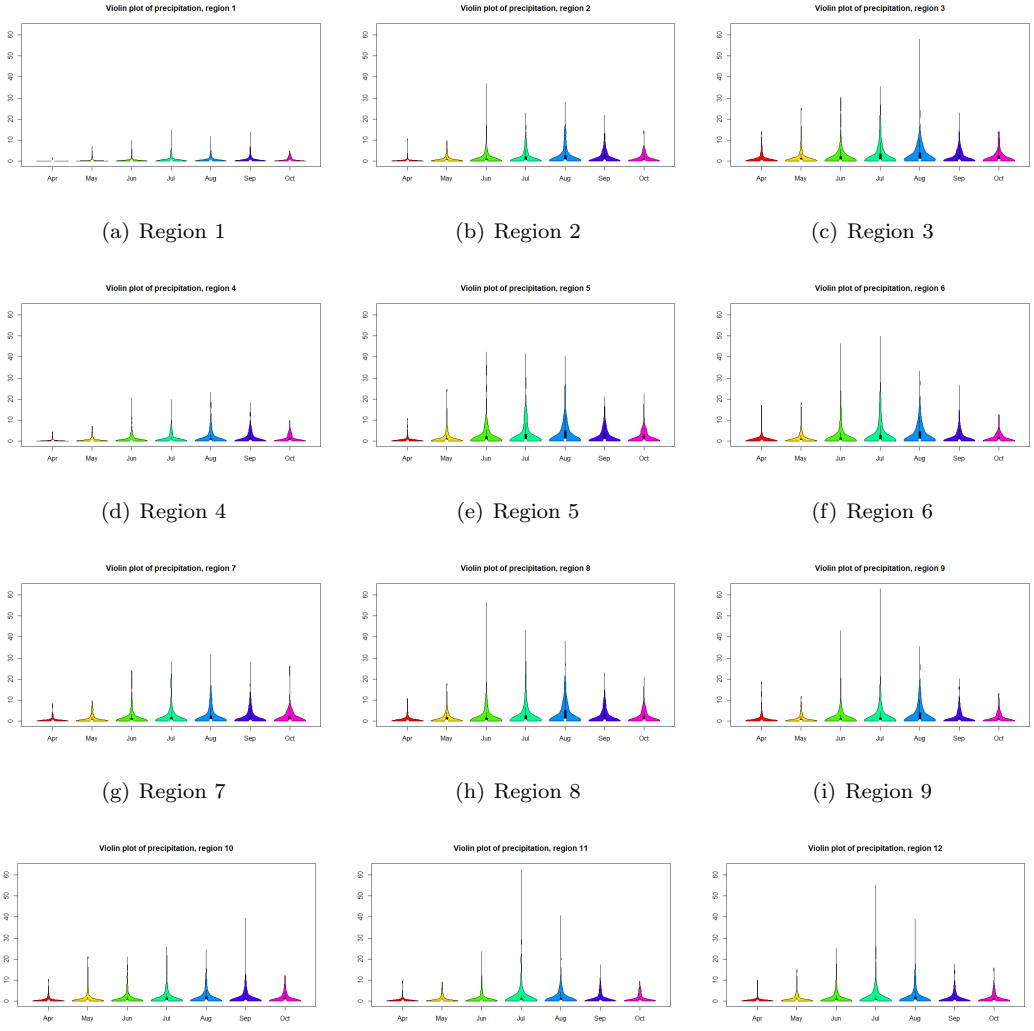


(f) September



(g) October

Figure A.1: Violin plots of the observed hourly precipitation for all twelve regions, split by the seven months.



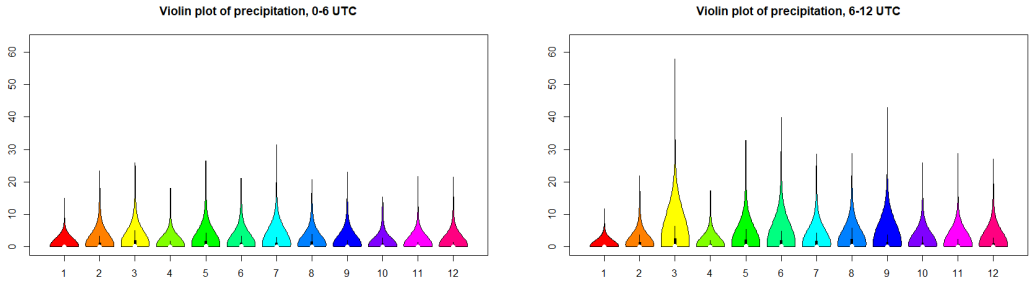
(a) Region 1 (b) Region 2 (c) Region 3

(d) Region 4 (e) Region 5 (f) Region 6

(g) Region 7 (h) Region 8 (i) Region 9

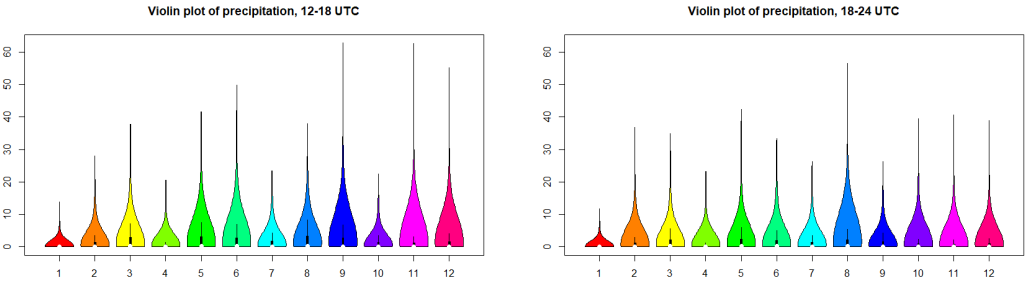
(j) Region 10 (k) Region 11 (l) Region 12

Figure A.2: Violin plots of the observed hourly precipitation for the seven months, split by all twelve regions.



(a) 0-6 UTC

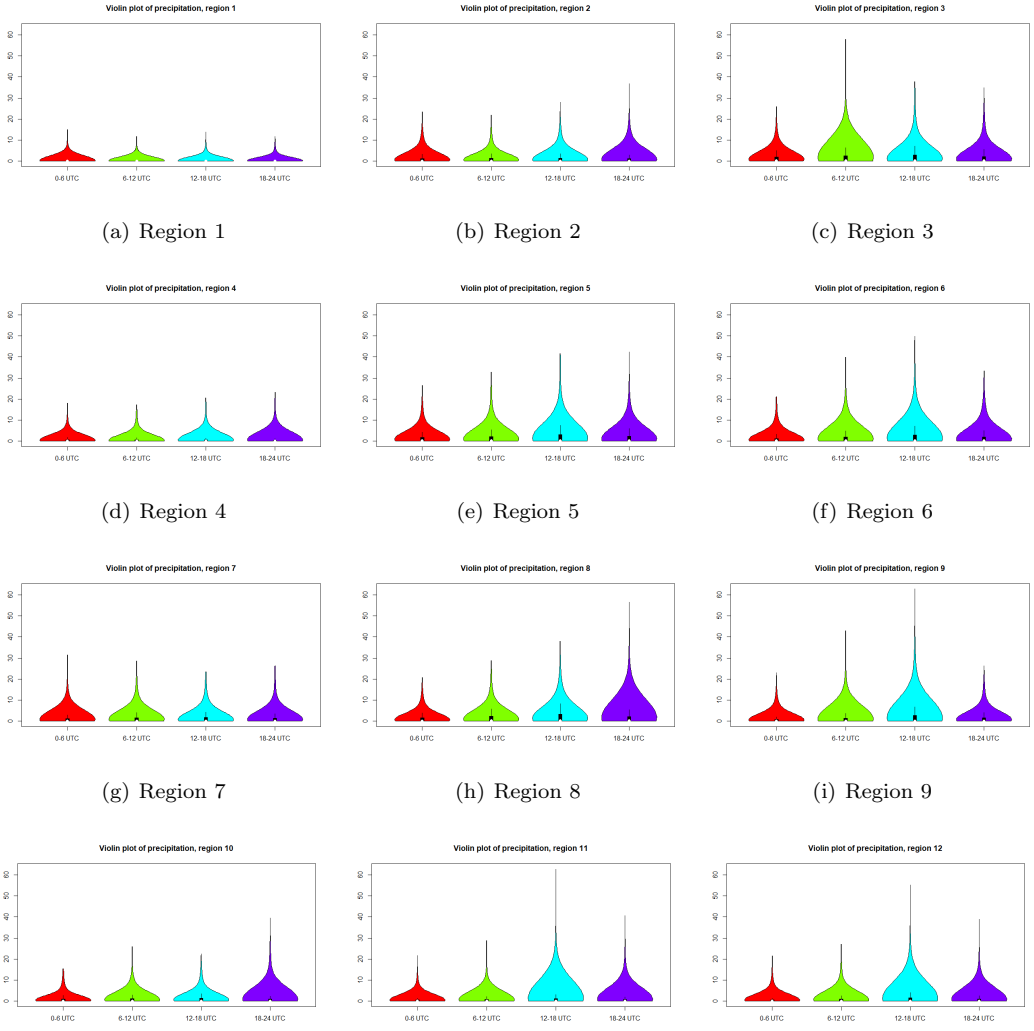
(b) 6-12 UTC



(c) 12-18 UTC

(d) 18-24 UTC

Figure A.3: Violin plots of the observed hourly precipitation for all twelve regions, split by the four time windows.



(a) Region 1 (b) Region 2 (c) Region 3

(d) Region 4 (e) Region 5 (f) Region 6

(g) Region 7 (h) Region 8 (i) Region 9

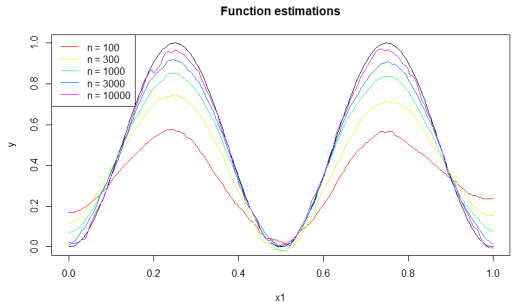
(j) Region 10 (k) Region 11 (l) Region 12

Figure A.4: Violin plots of the observed hourly precipitation for the four time windows, split by all twelve regions.

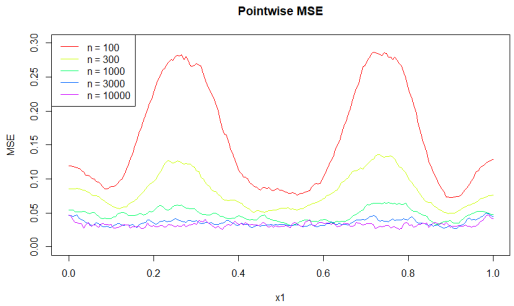
Appendix B

Additional simulation results

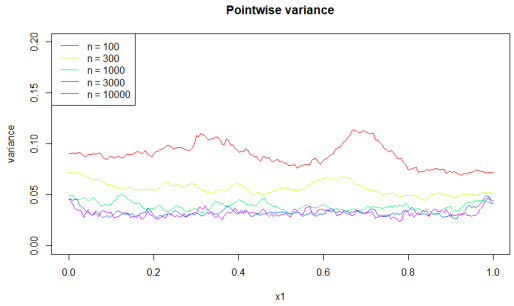
In this appendix, some additional illustrations are given, which were not relevant enough to be included in the chapters, but might be interesting to have a look at. All plots and table below have been referenced in at least one of the chapters.



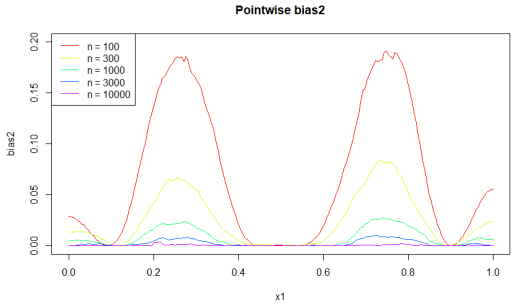
(a) Estimations



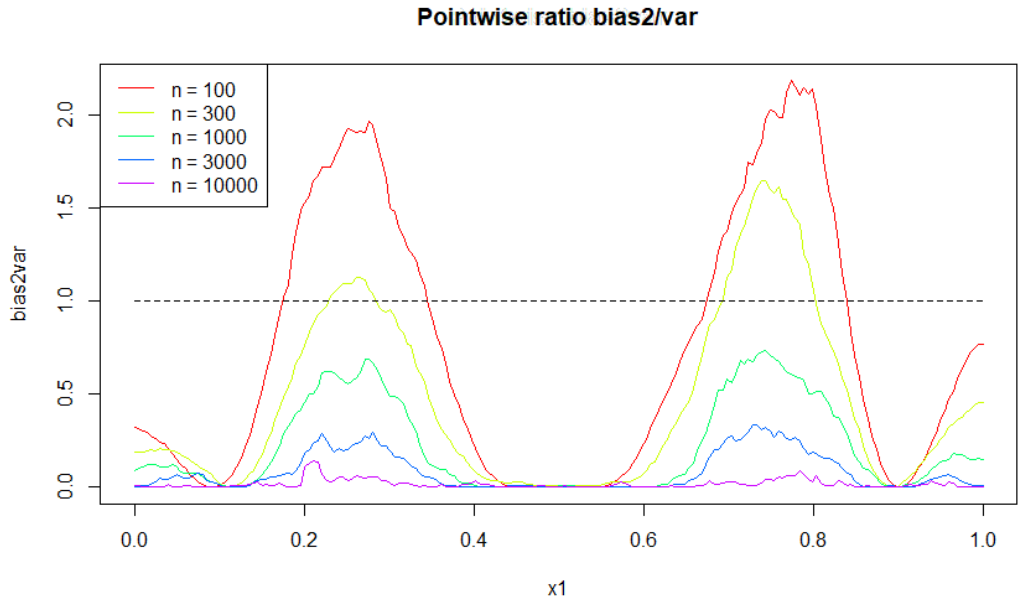
(b) Pointwise MSEs



(c) Pointwise variances

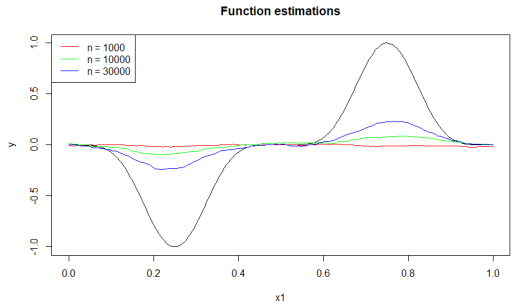


(d) Pointwise squared biases

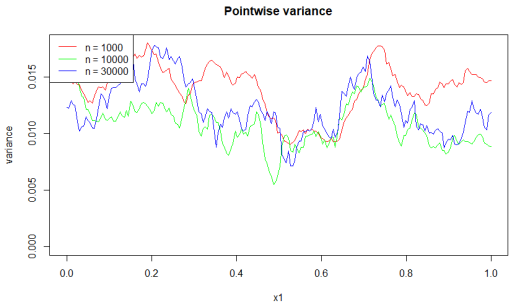


(e) Ratio between squared biases and variances

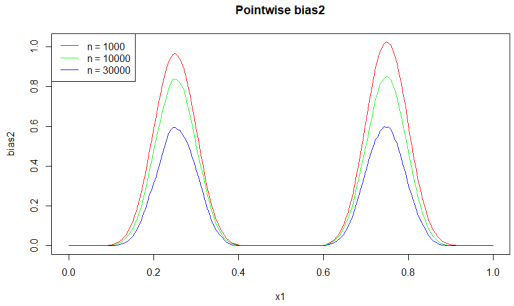
Figure B.1: Estimations and errors when using k-nearest forest neighbors with $d = 2$, $f = f_2$ and $n = 100, 300, 1000, 3000, 10000$, the parameters s and K are chosen in an optimal way. The bias domination has been greatly reduced.



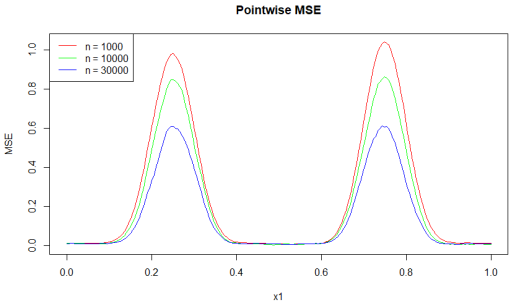
(a) Estimations



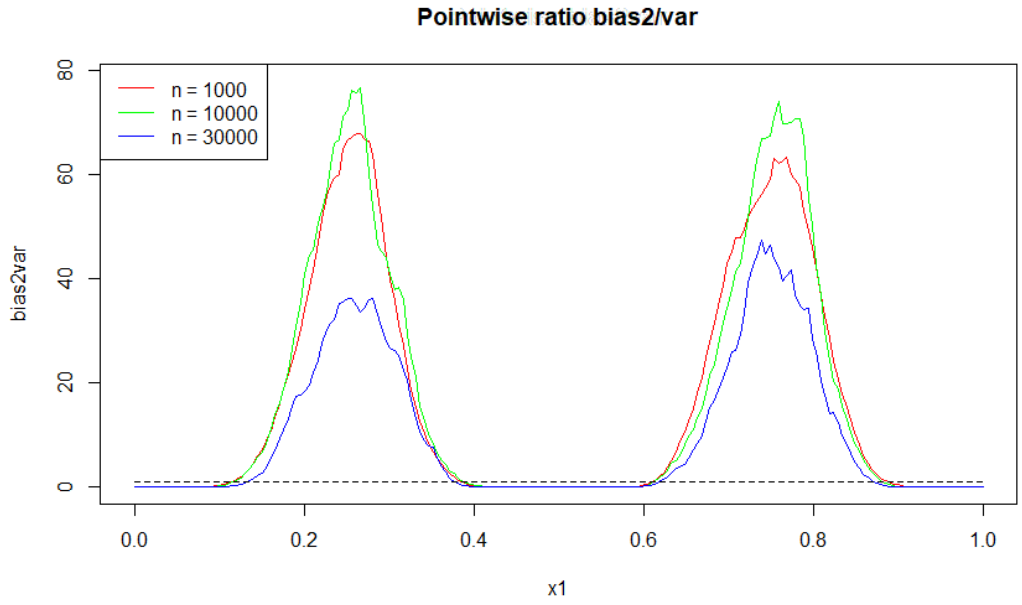
(b) Pointwise variances



(c) Pointwise squared biases

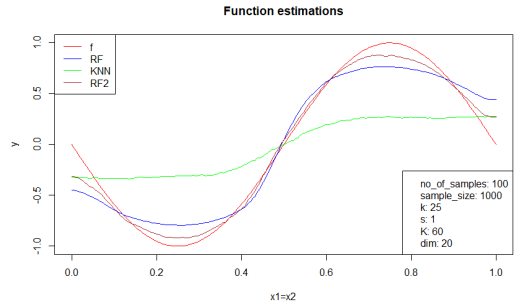


(d) Pointwise MSEs

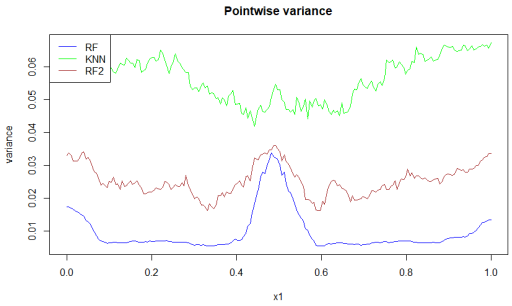


(e) Ratio between squared biases and variances

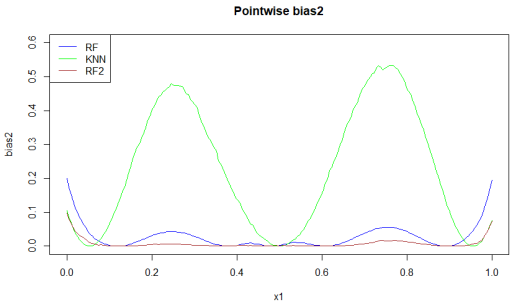
Figure B.2: Estimations and errors when using k-nearest forest neighbors with $d = 5$, $f = f_2$ and $n = 1000, 10000, 30000$, the parameters s and K are chosen in an optimal way. The bias domination has been greatly reduced, but the estimations are still poor.



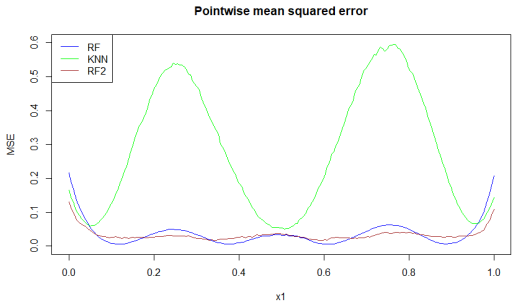
(a) Estimations



(b) Pointwise variances



(c) Pointwise squared biases



(d) Pointwise MSEs

Figure B.3: Estimations and errors when using k-nearest neighbors, random forests and k-nearest forest neighbors with $d = 2$, $f = f_1$ and $n = 1000$. The parameters k, s and K have been chosen optimally.

Covariate	LT1	LT2	LT3	LT4	LT5	LT6	LT7
Boyden_0m.hrlymax	0.04275	0.04104	0.04116	0.04091	0.04197	0.03717	0.05119
Boyden_0m.hrlymin	0.03401	0.03811	0.03407	0.03571	0.0416	0.0408	0.03364
Bradbury_0m.hrlymax	0.00112	0.00061	0.00088	0.00129	0.00067	0.00083	0.00053
Bradbury_0m.hrlymin	0.00087	0.00085	0.00081	0.00075	0.00091	0.00102	0.00095
DPT_500mb.hrlymax	0.00223	0.00142	0.0012	0.00162	0.00158	0.00178	0.00185
DPT_500mb.hrlymin	7e-04	0.00045	6e-04	0.00046	0.00061	0.00079	0.00071
DPT_600mb.hrlymax	0.00185	0.00231	0.00322	0.0118	0.01107	0.00966	0.01373
DPT_600mb.hrlymin	0.0011	0.00095	0.00185	0.00113	0.00105	0.00372	0.00186
DPT_700mb.hrlymax	0.00235	0.00298	0.00702	0.01431	0.01088	0.01026	0.00756
DPT_700mb.hrlymin	0.00316	0.00404	0.0327	0.00437	0.00521	0.00393	0.00271
DPT_850mb.hrlymax	0.00094	0.00079	0.00088	0.00103	0.00142	0.00175	0.00168
DPT_850mb.hrlymin	0.00189	0.00169	0.0016	0.0013	0.00172	0.00155	0.00206
Fateev_0m.hrlymax	0.02042	0.01895	0.01646	0.02644	0.01899	0.01479	0.02016
Fateev_0m.hrlymin	0.00266	0.00202	0.00375	0.00476	0.00445	0.00771	0.00419
Helicity_0m.hrlymax	0.02812	0.04133	0.03523	0.03467	0.03523	0.03521	0.04729
Helicity_0m.hrlymin	0.04367	0.03972	0.0344	0.03784	0.03703	0.0381	0.04591
Jefferson_0m.hrlymax	0.00199	0.00421	0.00648	0.00647	0.00285	0.00779	0.00585
Jefferson_0m.hrlymin	0.00097	0.00103	0.00148	0.00125	0.00139	0.00238	0.00142
K_index_0m.hrlymax	0.00712	0.00967	0.00989	0.01455	0.01489	0.01012	0.01079
K_index_0m.hrlymin	0.00249	0.00229	0.00182	0.00257	0.00301	0.00251	0.00355
LCL_0m.hrlymax	0.04013	0.03922	0.02377	0.01181	0.0123	0.01344	0.01098
LCL_0m.hrlymin	0.04014	0.04179	0.03676	0.03103	0.03916	0.04021	0.02692
LFC_0m.hrlymax	0.02349	0.01784	0.02462	0.02571	0.03437	0.02225	0.04302
LFC_0m.hrlymin	0.04346	0.01985	0.03413	0.03342	0.02793	0.03639	0.03073
LidStrength_0m.hrlymax	0.00122	0.00157	0.00181	0.00202	0.00204	0.00127	0.00121
LidStrength_0m.hrlymin	0.00026	0.00029	0.00023	0.00027	0.00029	0.00031	0.00021
Lifted_0m.hrlymax	0.04304	0.03676	0.03819	0.03752	0.03991	0.04233	0.04271
Lifted_0m.hrlymin	0.04375	0.04106	0.04077	0.03635	0.01651	0.01187	0.02737
PrecipitableWater_0m.hrlymax	0.00251	0.00184	0.00219	0.00338	0.00336	0.00327	0.00274
PrecipitableWater_0m.hrlymin	0.00101	0.00087	0.00126	0.00105	0.00135	0.00115	0.00111
Rackliff_0m.hrlymax	0.0371	0.03714	0.03322	0.03359	0.03696	0.04407	0.03942
Rackliff_0m.hrlymin	0.04145	0.03778	0.03822	0.03966	0.04123	0.0407	0.04176
Rain_0m.hrlymax	0.03608	0.03511	0.03518	0.02724	0.02579	0.02366	0.02479
Rain_0m.hrlymin	0.00044	0.00037	0.00277	0.00176	0.00137	0.00126	0.00139
Richardson_1m.hrlymax	0.00091	0.00114	0.00117	0.00115	0.00119	0.00121	0.00121
Richardson_1m.hrlymin	0	0	0	0	1e-05	1e-05	0
Richardson_2m.hrlymax	0.00094	0.00172	0.00133	0.00162	0.00135	0.0014	0.00108
Richardson_2m.hrlymin	5e-05	6e-05	0.00014	0.00011	7e-05	0.00015	9e-05
SWEAT_0m.hrlymax	0.03788	0.04049	0.03765	0.03288	0.03543	0.04126	0.04395
SWEAT_0m.hrlymin	0.00076	0.00073	0.00056	7e-04	0.00093	0.00077	0.00126
SWEAT_sin_0m.hrlymax	0.03841	0.03754	0.03341	0.03262	0.03121	0.03095	0.01998
SWEAT_sin_0m.hrlymin	0.0066	0.00883	0.00466	0.0045	0.0067	0.00283	0.00461
Shear_0m.hrlymax	0.00056	0.00063	0.00075	0.00066	0.00056	0.00074	0.00077
Shear_0m.hrlymin	0.00074	0.00065	0.00114	0.00149	0.00109	0.00079	0.00087

Table B.1: Covariate importances, after building a random forest with $s = 1$ (part 1).

Covariate	LT1	LT2	LT3	LT4	LT5	LT6	LT7
Showalter_0m.hrlymax	6e-04	0.001	0.00066	0.00217	0.00153	0.00195	0.00322
Showalter_0m.hrlymin	0.00074	0.00135	0.00099	0.00088	0.00147	0.00098	0.00081
StormTravel_0m.hrlymax	6e-04	0.00071	0.00088	0.00096	0.0011	0.00056	0.00101
StormTravel_0m.hrlymin	0.00077	0.00067	0.00075	0.0011	0.00116	0.00079	0.00101
Surface_CAPE_0m.hrlymax	0.00151	0.00211	0.003	0.00252	0.00244	0.00415	0.00276
Surface_CAPE_0m.hrlymin	1e-05	2e-05	1e-05	0	0	1e-05	0
Surface_CAPE_35m.hrlymax	0.00739	0.0085	0.00913	0.00918	0.00918	0.00887	0.008
Surface_CAPE_35m.hrlymin	0.00012	0.00018	0.00022	0.00021	0.00023	0.00022	0.00022
Surface_ConvInhib_0m.hrlymax	0.00116	0.00075	6e-04	0.00072	0.00087	0.00096	0.00071
Surface_ConvInhib_0m.hrlymin	0.03872	0.0393	0.0374	0.03707	0.03661	0.04714	0.04229
TQ_0m.hrlymax	0.01545	0.02011	0.01828	0.02569	0.01362	0.01528	0.00881
TQ_0m.hrlymin	0.00407	0.00346	0.00446	0.00544	0.00349	0.00297	0.0055
TotalTotals_0m.hrlymax	0.01111	0.01946	0.01002	0.01804	0.00666	0.00482	0.00668
TotalTotals_0m.hrlymin	0.00152	0.00164	0.00222	0.00322	0.00145	0.00137	0.00145
TotalTotals_1e_0m.hrlymax	0.04144	0.03699	0.03417	0.03141	0.03364	0.04131	0.04676
TotalTotals_1e_0m.hrlymin	0.04144	0.04015	0.03365	0.03086	0.02783	0.04428	0.03268
TotalTotals_2e_0m.hrlymax	0.00866	0.01388	0.01098	0.02174	0.01052	0.00618	0.00755
TotalTotals_2e_0m.hrlymin	0.00275	0.00223	0.00209	0.01446	0.00475	0.00268	0.00379
UWND_700mb.hrlymax	0.04002	0.03599	0.03635	0.03129	0.03755	0.03849	0.04465
UWND_700mb.hrlymin	0.04069	0.03857	0.03743	0.0378	0.03419	0.04285	0.03798
VWND_700mb.hrlymax	0.02592	0.03296	0.03921	0.03505	0.02752	0.0062	0.00492
VWND_700mb.hrlymin	0.03579	0.04196	0.04017	0.03396	0.03533	0.03846	0.04324
WDIR_500mb.hrlymax	0.00219	0.00265	0.00195	0.00262	0.00236	0.00194	0.00106
WDIR_500mb.hrlymin	0.00241	0.00362	0.0079	0.01467	0.02239	0.01017	0.01392
WDIR_850mb.hrlymax	0.00586	0.00484	0.00865	0.00403	0.01319	0.00466	0.0188
WDIR_850mb.hrlymin	0.00056	0.00095	5e-04	0.00099	0.00116	0.00051	0.00053
WSPD_500mb.hrlymax	0.00062	0.00067	0.00068	0.00067	0.00073	0.00075	0.00074
WSPD_500mb.hrlymin	0.00134	0.00103	0.00142	0.00097	0.00095	0.00095	0.00091
WSPD_850mb.hrlymax	0.00178	0.00181	0.00115	0.00098	0.0011	0.00146	0.00254
WSPD_850mb.hrlymin	0.00077	0.00084	0.00056	0.00072	0.00088	9e-04	0.00128
modJefferson_0m.hrlymax	0.01168	0.01428	0.01485	0.01706	0.01831	0.01851	0.01555
modJefferson_0m.hrlymin	0.00371	0.00402	0.00366	0.00238	0.00391	0.0037	0.00608
theataW_500mb.hrlymax	0.00134	0.00057	0.00072	0.00079	0.00101	0.00115	0.00098
theataW_500mb.hrlymin	0.00052	0.00044	0.00052	0.00069	0.00092	0.00077	0.00083
theataW_850mb.hrlymax	0.00056	0.00048	0.00071	0.00068	0.00098	0.00103	0.00097
theataW_850mb.hrlymin	0.00072	0.00083	0.00066	0.00098	0.00058	0.00098	0.00059
theataW_925mb.hrlymax	0.00081	0.00064	0.00074	0.00079	0.00095	0.00106	0.00135
theataW_925mb.hrlymin	0.00068	0.00049	6e-04	0.00062	8e-04	0.00062	0.00068
theataWs_500mb.hrlymax	0.00079	0.00053	0.00064	0.00066	0.00096	0.00074	0.00066
theataWs_500mb.hrlymin	6e-04	0.00041	0.00074	0.00067	0.00104	0.00089	0.00075
region_x	0.00078	0.00059	0.00071	0.00047	0.00089	0.00038	0.00131
region_y	0.00014	0.00016	0.00015	0.00014	0.00022	0.00025	0.00018
date	0.00016	0.00018	0.00015	0.00067	0.03763	0.0436	0.00013
time1	9e-05	9e-05	8e-05	0.00012	0.00017	0.00015	0.00018
time2	0.00011	0.00016	0.00014	1e-04	1e-04	0.00015	0.00015

Table B.2: Covariate importances, after building a random forest with $s = 1$ (part 2).

Appendix C

R code documentation

`add_covariates`

Description

Adds covariates about region, date and time to the existing covariates.

Input

`data`: list containing all covariates, the response variable and information about region, date and time

Output

List containing original and newly added covariates and response variable.

Dependencies

`R` built-in: `ceiling`, `list`, `min`, `strftime`
`rlist` package: `list.append`

`calculate`

Description

Calculates accuracy of predictions.

Input

`predictions`: matrix of which each row is a sample with predictions for all test points
`f`: regression function

Output

List containing the pointwise mean, pointwise bias, pointwise squared bias, pointwise MSE (all vectors) and MISE (a single number).

Dependencies

`R` built-in: `colMeans`, `integrate`, `length`, `list`, `round`

create_estimate

Description

Creates simulated data and applies the regression methods on these data.

Input

`no_of_samples`: number of samples
`sample_size`: sample size
`test`: vector of test points
`f`: regression function
`k`: k (only applicable if `alg` is 'knn' or 'all')
`s`: s (only applicable if `alg` is not 'knn')
`K`: K (only applicable if `alg` is 'knfn')
`d`: number of covariates
`error`: error function (noise term)
`alg`: one of 'knn', 'rf', 'knfn', 'all'

Output

Matrix of size `no_of_samples` $\times$ |`test`|, giving a prediction for each sample and each test point. If `alg` is 'all', then a list of three matrices is returned.

Dependencies

Own code: `estimate`
`R` built-in: `length`, `rnorm`, `runif`

create_test_set

Description

Creates test set, given size and number of dimensions.

Input

`test_size`: number of test points
`dim`: number of dimensions

Output

Matrix of size `test_size` $\times$ `dim`, where each of the `test_size` contains a `dim`-dimensional test point

Dependencies

`R` built-in: `matrix`, `seq`

cross_validation

Description

Applies 3-fold cross-validation on precipitation data.

Input

`lead`: lead time number
`D`: number of most predictive covariates to consider
`ks`: vector of k-values to use (only applicable if `method` is 'knn' or 'all')
`ss`: vector of s-values to use (only applicable if `method` is 'knn' or 'nwp')
`Ks`: vector of K-values to use (only applicable if `method` is 'knfn')
`method`: one of 'nwp', 'knn', 'rf', 'knfn', 'all'

Output

The minimal RMSE among all parameter values in the input, together with the parameter value for which the minimal RMSE was achieved.

Dependencies

Own code: `add_covariates`, `estimate`, `top_D`
`R` built-in: `match`, `mean`, `min`, `length`, `list`, `readRDS`

estimate

Description

Applies the regression methods on input data.

Input

`X`: matrix of covariates
`Y`: vector of response values
`no_of_samples`: number of samples
`test`: vector of test points
`k`: k (only applicable if `method` is 'knn' or 'all')
`s`: s (only applicable if `method` is not 'knn')
`K`: K (only applicable if `method` is 'knfn')
`method`: one of 'knn', 'rf', 'knfn', 'all'

Output

Matrix of size `no_of_samples` $\times$ |`test`|, giving a prediction for each sample and each test point. If `method` is 'all', then a list of three matrices is returned.

Dependencies

Own code: `K_largest`
`R` built-in: `length`, `list`, `matrix`, `sum`
`KernelKnn` package: `KernelKnn`
`grf` package: `get_sample_weights`, `predict`, `regression_forest`

K_largest

Description

Computes $w_i^{(K)}(\mathbf{x}^*)$ based on $w_i(\mathbf{x}^*)$ and K.

Input

`weights`: vector of weights
`K`: K

Output

Vector of length |`weights`|, where the all weights in `weights` except the K largest have been set to zero, and the non-zero weights rescaled to sum one.

Dependencies

`R` built-in: `sort`, `sum`, `vector`

top_D

Description

Gets top-D of most predictive covariates.

Input

`X`: matrix of covariates
`Y`: vector of response variables
`D`: number of most predictive covariates to consider

Output

A new matrix of covariates containing only the `D` most predictive ones.

Dependencies

`R` built-in: `matrix`, `order`
`grf` package: `regression_forest`, `variable_importance`

Index