

Predicting tomato crop yield from weather data using statistical learning techniques

by

Margaret de Villiers

*Thesis presented in partial fulfilment of the requirements
for the degree of Master of Commerce in Mathematical
Statistics in the Faculty of Economic and Management
Sciences at Stellenbosch University*



Department of Statistics and Actuarial Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Prof. Daniel W. Uys

March 2017

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2017

Copyright © 2017 Stellenbosch University
All rights reserved.

Abstract

Predicting crop harvest quantities accurately is important in managing a farming enterprise effectively, facilitating decisions regarding crop management, allocation of resources, anticipated delivery times and quantities to customers and produce pricing, to name but a few. The aim of this project is to develop statistical models for predicting harvest quantity of field-grown crops on a commercial tomato farm in South Africa using weather data.

Planting and harvest data for a seven-year period were provided by the tomato farm, while daily and 10-daily weather data for the same period were obtained from a nearby weather station and from satellite data. The data sets were cleaned, and the time series data were summarised in the form of a single summary statistic for each weather variable over the growing period of each crop. Median and total harvest density (t/ha) were each modelled using multiple linear regression, the lasso, regression trees, bagged regression trees, random forests and boosted regression trees.

All of the crop and weather variables turned out to be informative in predicting tomato yield, with the average of the daily average wind speed and the average of the daily maximum relative humidity readings over the crops' growing periods emerging as the most important predictors of median and total harvest density, respectively. Random forests modelled median harvest density the most accurately with an estimated mean absolute prediction error of 0.37 t/ha, while bagged regression trees modelled total harvest density the most accurately with a mean absolute prediction error of 12.67 t/ha. The model parameter estimators of all of the modelling techniques tended to have low variances, and the sizes of the prediction errors are most likely due to factors such as the absence of important predictors (soil fertility, irrigation regimes, etc.) from the models and the summary of the weather time series over the crops' growing periods into single values.

Opsomming

Die akkurate voorspelling van oes opbrengs is belangrik in die effektiewe bestuur van 'n boerdery. Beter besluite ten opsigte van gewasbestuur, die toewysing van hulpbronne, afleweringstye, hoeveelhede wat aan klante gelewer moet word en produkpryse kan geneem word indien opbrengs akkuraat voorspel word. Die doel van hierdie projek is om statistiese modelle te ontwikkel wat aangewend kan word om tamatie oes opbrengs te voorspel deur gebruik te maak van klimaatsveranderlikes. Oop land tamatie aanplantings van 'n kommersiële tamatie boerdery in Suid-Afrika is vir dié doel gebruik.

Aanplantings- en opbrengsdata vir 'n sewejaar tydperk is vanaf die tamatieplaas verkry, terwyl daaglikse en 10-daaglikse klimaatsmetings vir dieselfde tydperk by 'n nabygeleë weerstasie asook vanaf satellietdata ingesamel is. Die datastelle is skoongemaak, en opsommende maatstawwe is vir elke klimaatsveranderlike oor die groeitydperk van elke aanplanting bereken. Mediaan en totale opbrengs (t/ha) is afsonderlik gemodelleer met behulp van meervoudige lineêre regressie, die lasso, regressiebome, “bagged” regressiebome, ewekansige woude en “boosted” regressiebome.

Al die aanplantings- en klimaatsveranderlikes is betekenisvol in die voorspelling van opbrengs, met die gemiddelde van die daaglikse gemiddelde windspoed en die gemiddelde van die daaglikse maksimum relatiewe humiditeitslesings oor die aanplantings se groeitydperke as belangrikste voorspellers van onderskeidelik mediaan en totale oes opbrengs. Ewekansige woude het mediaan opbrengs die akkuraatste voorspel met 'n beraamde gemiddelde absolute voorspellingsfout van 0.37 t/ha, terwyl “bagged” regressiebome die totale opbrengs die akkuraatste voorspel het met 'n gemiddelde absolute voorspellingsfout van 12.67 t/ha. Die beraamers van modelparameters van al die modeleringstegnieke het klein variansies. Die groottes van die voorspellingsfoute is waarskynlik te wyte aan faktore soos die afwesigheid van belangrike voor-

spellers (soos bv. grondvrugbaarheid en besproeiingstegnieke), asook die opsommende maatstawwe wat vir die klimaatsveranderlikes oor die groeitydperk van elke aanplanting bereken is.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

- My supervisor, Prof. Danie Uys, whose unwavering guidance, support, patience and understanding throughout the project were invaluable for the completion of my Masters degree. You are a true friend, Danie!
- The South African Statistical Association (SASA) and the National Research Foundation (NRF) for jointly granting me a bursary for pursuing a Masters in Mathematical Statistics.
- The academic staff of the Department of Statistics and Actuarial Science at Stellenbosch University. It is a pleasure to be taught by lecturers that are passionate about their subjects and dedicated to their roles in shaping future statisticians. The friendliness and support of the academic and support staff and the comradeship of the students made for a pleasant working environment.
- George Schwalb of GRS Actuarial Consulting, Stellenbosch. George initially suggested the topic for this thesis and set up communication channels with staff on the tomato farm.
- The CEO and staff of the tomato farm that provided me with crop data for this project. They spent many hours of their busy schedules not only preparing and sending me the data, but also explaining their tomato cultivation practices to me. Their advice was indispensable during the data-cleaning process and for placing the results of the analyses within the context of the agricultural practices on the farm. I hope that the results of this study prove useful to them.

- The South African Weather Service (SAWS) furnished me with hourly weather data from some of their weather stations in Limpopo Province. Although this data set did not end up in the final analyses, it was very useful for comparative purposes during the data-cleaning process.
- The Agricultural Research Council (ARC) supplied me with daily and hourly weather data from weather stations very close to the localities of interest in Limpopo Province. The statistical models were trained on some of this data.
- Environmental Analysis and Remote Sensing Earth Environment Monitoring B.V. (EARS-E2M), a Dutch company, sent me relative evapotranspiration data, which were also used in the training of the statistical models.
- On a more personal level, I would like to thank my family, whose unfaltering love, help and support made the work load feel lighter and the seemingly impossible seem more achievable.
- Our neighbour's cat, Kitts. She sat with me through many long evenings and nights of work. Her companionship during the quieter hours was comforting. I am grateful to our neighbours for allowing Kitts to spend her time with us.

Contents

Declaration	i
Abstract	ii
Opsomming	iii
Acknowledgements	v
Contents	vii
List of Figures	x
List of Tables	xvi
Notation and abbreviations	xviii
1 Thesis overview	1
1.1 Problem statement and study objectives	1
1.2 Clarification of key concepts	2
1.3 Research design and methodology	3
1.4 Importance of the study	4
1.5 Chapter outline	4
2 Literature Review	6
2.1 Tomatoes as an agricultural crop	6
2.2 Predicting crop yield	10
2.3 Commonly recorded weather variables	13
2.4 Statistical modelling techniques	18
2.5 Aims of this study	65

<i>CONTENTS</i>	viii
3 Data description and exploratory analyses	67
3.1 The tomato crop data set	67
3.2 The weather and evapotranspiration data set	92
3.3 The crop and weather variables combined	119
4 The linear model and related methods	126
4.1 Introduction	126
4.2 The null model	127
4.3 Multiple linear regression	128
4.4 The lasso	144
4.5 Discussion	151
5 Regression trees and tree-based methods	155
5.1 Introduction	155
5.2 Regression trees	156
5.3 Bagged regression trees	166
5.4 Random forests	173
5.5 Discussion	179
6 An adaptive tree-based method: Boosting	183
6.1 Introduction	183
6.2 Methods	184
6.3 Results	184
6.4 Discussion	190
7 Discussion	196
Appendices	203
A R code for the modelling analyses	204
A.1 Preparing the data sets	204
A.2 The null model	207
A.3 Multiple linear regression	208
A.4 The lasso	215
A.5 Regression trees	218
A.6 Bagged regression trees	223
A.7 Random forests	226

<i>CONTENTS</i>	ix
A.8 Boosted regression trees	227
List of References	233

List of Figures

2.1	Partitioning of the input space by a decision tree	35
3.1	Harvest time series of three block crops in Field crop 3	69
3.2	Planting events and harvest time series of two block crops in Field crop 20	70
3.3	Planting events and harvest time series of two block crops in Field crop 32	71
3.4	Original and cleaned distributions of the separate harvest masses of the block crops	76
3.5	Original and cleaned distributions of the separate harvest densities of the block crops	76
3.6	Original and cleaned distributions of the median harvest masses of the block crops	76
3.7	Original and cleaned distributions of the median harvest densities of the block crops	77
3.8	Original and cleaned distributions of the total harvest masses of the block crops	77
3.9	Original and cleaned distributions of the total harvest densities of the block crops	77
3.10	Original and cleaned distributions of the areas of the block crops .	78
3.11	Original and cleaned distributions of the growing periods of the block crops	78
3.12	Original and cleaned distributions of the harvest periods of the block crops	78
3.13	Original and cleaned distributions of the crop durations of the block crops	79

LIST OF FIGURES

xi

3.14	Original and cleaned distributions of the number of harvest events of the block crops	79
3.15	Distributions of the harvest masses of the training block crops	80
3.16	Distributions of the harvest densities of the training block crops	81
3.17	Median harvest densities of the fields in the training data	82
3.18	Total harvest densities of the fields in the training data	82
3.19	Harvest mass versus area for the training block crops	83
3.20	Harvest density versus area for the training block crops	83
3.21	Harvest density versus planting density for the training block crops	84
3.22	Planting and harvest dates for the training block crops	84
3.23	Median harvest density for different planting times	85
3.24	Total harvest density for different planting times	85
3.25	Growing period for different planting times	86
3.26	Harvest period for different planting times	86
3.27	Crop duration for different planting times	87
3.28	Harvest density versus growing period	87
3.29	Harvest density versus harvest period	88
3.30	Harvest density versus total crop duration	88
3.31	Harvest density versus number of harvest events	89
3.32	Median versus total harvest density	90
3.33	Scatterplots of the crop variables in the training data set	91
3.34	Map of the tomato farm and ARC weather station	93
3.35	Original and cleaned time series of daily maximum and minimum temperature	95
3.36	Original and cleaned time series of daily maximum and minimum relative humidity	96
3.37	Original and cleaned time series of daily average wind speed	97
3.38	Original and cleaned time series of daily total rainfall	98
3.39	Original and cleaned time series of daily total reference evapotranspiration	99
3.40	Time series of dekad relative evapotranspiration	100
3.41	Daily maximum and minimum temperature and relative humidity throughout the year	101

3.42	Daily average wind speed, daily total rainfall, daily total reference evapotranspiration and dekad relative evapotranspiration throughout the year	102
3.43	Summary statistics of daily maximum temperature for the training block crops	103
3.44	Summary statistics of daily minimum temperature for the training block crops	104
3.45	Summary statistics of maximum relative humidity for the training block crops	105
3.46	Summary statistics of minimum relative humidity for the training block crops	106
3.47	Summary statistics of daily average wind speed for the training block crops	107
3.48	Summary statistics of daily total rainfall for the training block crops	108
3.49	Summary statistics of daily total reference evapotranspiration for the training block crops	109
3.50	Summary statistics of dekad relative evapotranspiration for the training block crops	110
3.51	Harvest density versus median daily maximum temperature for the training block crops	114
3.52	Harvest density versus average daily minimum temperature for the training block crops	114
3.53	Harvest density versus average daily maximum relative humidity for the training block crops	115
3.54	Harvest density versus average daily minimum relative humidity for the training block crops	115
3.55	Harvest density versus average daily average wind speed for the training block crops	116
3.56	Harvest density versus average daily total rainfall for the training block crops	116
3.57	Harvest density versus minimum daily total reference evapotranspiration for the training block crops	117
3.58	Harvest density versus minimum dekad relative evapotranspiration for the training block crops	117

<i>LIST OF FIGURES</i>	xiii
3.59 Scatterplots of the ARC and EARS-E2M weather time series summary statistics in the training data set	119
3.60 Scatterplots of the crop variables and the weather time series summary statistics in the training data set	120
4.1 Model predictions versus validation responses for the null models of median and total harvest density	127
4.2 Statistics for median harvest density models of different sizes chosen by best subset selection	130
4.3 Input terms included in the median harvest density models for different complexity measures	131
4.4 Statistics for total harvest density models of different sizes chosen by best subset selection	132
4.5 Input terms included in the total harvest density models for different complexity measures	133
4.6 Model residuals of the multiple linear regression model of median harvest density containing the eleven linear terms	134
4.7 Model residuals of the multiple linear regression model of total harvest density containing the eleven linear terms	136
4.8 Model predictions versus validation responses for the multiple linear regression models of median and total harvest density containing the linear terms	137
4.9 Model residuals of the multiple linear regression model of median harvest density containing the eleven linear and four quadratic terms	138
4.10 Model residuals of the multiple linear regression model of total harvest density containing the eleven linear and four quadratic terms	140
4.11 Model predictions versus validation responses for the multiple linear regression models of median and total harvest density containing the linear and quadratic terms	142
4.12 Model residuals of the multiple linear regression model of median harvest density containing the linear and quadratic terms from best subset selection	143
4.13 Model residuals of the multiple linear regression model of total harvest density containing the linear and quadratic terms from best subset selection	145

4.14	Model predictions versus validation responses for the multiple linear regression models of median and total harvest density containing the linear and quadratic terms selected by best subset selection	146
4.15	Cross-validation to determine the optimum value of the complexity parameter λ for the lasso models of median and total harvest density	147
4.16	Coefficient paths over the complexity parameter values for the lasso model of median harvest density	148
4.17	Coefficient paths over the complexity parameter values for the lasso model of total harvest density	148
4.18	Model predictions versus validation responses for the lasso models of median and total harvest density containing linear and quadratic terms	151
5.1	Cross-validation errors achieved on different complexity parameter settings for the regression trees	157
5.2	Regression tree of median harvest density	158
5.3	Regression tree of total harvest density	161
5.4	Cross-validation errors achieved on the nested sequence of regression trees for pruning	165
5.5	Pruned regression tree of median harvest density	166
5.6	Pruned regression trees of total harvest density	167
5.7	Model predictions versus validation responses for the full and pruned regression trees of median and total harvest density	168
5.8	Prediction error for different numbers of trees in the bagged regression tree model	169
5.9	Importance of the input variables in the bagged regression tree models of median and total harvest density	170
5.10	Partial dependence plots of the fitted response curve against each of the input variables in the bagged regression tree model of median harvest density	171
5.11	Partial dependence plots of the fitted response curve against each of the input variables in the bagged regression tree model of total harvest density	172
5.12	Model predictions versus validation responses for the bagged regression tree models of median and total harvest density	173

5.13	Prediction error for different numbers of trees in the random forest model	174
5.14	Importance of the input variables in the random forest models of median and total harvest density	175
5.15	Partial dependence plots of the fitted response curve against each of the input variables in the random forest model of median harvest density	176
5.16	Partial dependence plots of the fitted response curve against each of the input variables in the random forest model of total harvest density	177
5.17	Model predictions versus validation responses for the random forest models of median and total harvest density	178
5.18	Validation errors achieved with different input variable subset sizes for the random forest models	178
6.1	Cross-validation errors of the full boosted regression tree models containing different numbers of trees	185
6.2	Importance of the input variables in the full boosted regression tree model of median and total harvest density	186
6.3	Partial dependence plots of the fitted response values against each of the input variables in the full boosted regression tree model of median harvest density	187
6.4	Partial dependence plots of the fitted response values against each of the input variables in the full boosted regression tree model of total harvest density	188
6.5	Partial dependence plots of the fitted response curve against each of the input variables in the full boosted regression tree model of median harvest density	189
6.6	Partial dependence plots of the fitted response curve against each of the input variables in the full boosted regression tree model of total harvest density	190
6.7	Model predictions versus validation responses for the full boosted regression tree model of median and total harvest density	191
6.8	Cross-validation errors for the number of input variables to be dropped from the full boosted regression tree model	191

List of Tables

3.1	Variables in the tomato crop data set	73
3.2	Correlation coefficients for the crop variables in the training data set	90
3.3	Time series variables in the weather data set	94
3.4	Correlation coefficients for the daily maximum temperature summary statistics and harvest density in the training data set	111
3.5	Correlation coefficients for the daily minimum temperature summary statistics and harvest density in the training data set	111
3.6	Correlation coefficients for the daily maximum relative humidity summary statistics and harvest density in the training data set . .	111
3.7	Correlation coefficients for the daily minimum relative humidity summary statistics and harvest density in the training data set . .	112
3.8	Correlation coefficients for the daily average wind speed summary statistics and harvest density in the training data set	112
3.9	Correlation coefficients for the daily total rainfall summary statistics and harvest density in the training data set	112
3.10	Correlation coefficients for the daily total reference evapotranspiration summary statistics and harvest density in the training data set	113
3.11	Correlation coefficients for the dekad relative evapotranspiration summary statistics and harvest density in the training data set . .	113
3.12	Correlation coefficients for the summary statistics of the weather variables in the training data set	122
3.13	Correlation coefficients for the crop variables and summary statistics of the weather variables in the training data set	123
3.14	Input variables for modelling tomato harvest density	124
3.15	Variables modelled by the tomato harvest models	125

4.1	Coefficients for the multiple linear regression model of median harvest density containing only the linear terms	135
4.2	Coefficients for the multiple linear regression model of total harvest density containing only the linear terms	137
4.3	Coefficients for the multiple linear regression model of median harvest density containing the linear and quadratic terms	139
4.4	Coefficients for the multiple linear regression model of total harvest density containing the linear and quadratic terms	141
4.5	Coefficients for the multiple linear regression model of median harvest density containing the linear and quadratic terms selected by best subset selection	144
4.6	Coefficients for the multiple linear regression model of total harvest density containing the linear and quadratic terms selected by best subset selection	146
4.7	Coefficients for the lasso models of median and total harvest density containing the linear and quadratic terms	150
4.8	Validation errors of linear models of total and median harvest density	152
5.1	Cross-validation errors of regression trees with different cost parameters specified	157
5.2	Validation errors of linear and tree-based models of total and median harvest density	179
6.1	Validation errors of linear, tree-based models and boosting of total and median harvest density	192

Notation and abbreviations

Mathematical notation

∞	infinity
\rightarrow	approaches, e.g. $n \rightarrow \infty$
$:=$	is defined as
\approx	is approximately equal to
\implies	implies
\gg ; \ll	is much larger than; is much smaller than
\forall	for all, e.g. $x^2 \geq 0 \forall x \in \mathbb{R}$
\subseteq	is a subset of or is equal to
\cup	union
\in	is an element of
$\min\{f(x)\}$	the minimum value that $f(x)$ can reach, e.g. $x \in \mathbb{R} \implies \min(x^2 + 1) = 1$
$\operatorname{argmin}_x\{f(x)\}$	the argument x that minimises $f(x)$, e.g. $x \in \mathbb{R} \implies \operatorname{argmin}_x(x^2 + 1) = 0$, since $f(0) = 1$
$\max\{f(x)\}$	the maximum value that $f(x)$ can reach, e.g. $x \in \mathbb{R} \implies \max(1 - x^2) = 1$
$\operatorname{argmax}_x\{f(x)\}$	the argument x that maximises $f(x)$, e.g. $x \in \mathbb{R} \implies \operatorname{argmax}_x(1 - x^2) = 0$, since $f(0) = 1$
w	a scalar
$ w $	the absolute value of w

$\mathbf{0}_p$	a column vector consisting of p zeroes
$\mathbf{1}_p$	a column vector consisting of p ones
\mathbf{w}	a column vector containing one or more entries
\mathbf{w}_n	column vectors containing n entries
$(\mathbf{w}_n)^T$	the transpose of the column vector \mathbf{w}_n
$\ \mathbf{w}_n\ $	the norm of \mathbf{w}_n
$\mathbf{w}^T \mathbf{u}$	the inner product of \mathbf{w} with \mathbf{u}
\mathbf{W}	the matrix \mathbf{W} of unspecified dimensions
$\mathbf{W}_{n \times p}$	the $n \times p$ matrix \mathbf{W}
$(\mathbf{W}_{n \times p})^{-1}$	the inverse of the $n \times p$ matrix \mathbf{W}
$(\mathbf{W}_{n \times p})^T$	the transpose of the $n \times p$ matrix \mathbf{W}

Statistical symbols

B	number of bootstrap samples
$C_\alpha(T)$	the cost-complexity criterion for a nested subtree $T \subseteq T_0$ of the unpruned tree T_0 for a fixed value of the tuning parameter α
d	the size of the subset of input variables included in a model
$H_0; H_a$	the null and alternative hypotheses of a hypothesis test
$\mathcal{I}_l^2(T); \mathcal{I}_l^2$	the squared relative importance of the l^{th} input variable x_l in a single regression tree and in a linear combination of regression trees, respectively
\hat{i}_j^2	the squared improvement in the training error as a result of the split at the j^{th} internal node in a regression tree
J	number of mutually exclusive rectangular regions into which the input space is partitioned in a regression tree model

K	number of folds into which the training set is randomly partitioned in K -fold cross-validation
M	the number of terms in a multiple linear regression model or the number of iterations in a boosted regression tree model
n	the number of observations in the training data set
n_1, n_2, \dots, n_J	the number of training observations in each of the J regions of a decision tree model
p	the number of input variables
q	the number $q \leq p$ of the p input variables to be excluded from a model
R^2	coefficient of determination
R_1, R_2, \dots, R_J	the J regions of the input space determined by a decision tree model corresponding to the J leaves of the decision tree
$R_{1m}, R_{2m}, \dots, R_{Jm}$	the J regions of the input space determined by the decision tree fit during the m^{th} iteration of a boosted regression tree model
$R; R_1(j, s); R_2(j, s)$	region R of the input space, along with the two regions $R_1(j, s)$ and $R_2(j, s)$ resulting from the split of R into two subregions using the j^{th} input variable X_j and split point $X_j = s$
r or $r(x, y)$	sample correlation coefficient (between the observed values of variables X and Y)
$\mathbf{r} = (r_1, \dots, r_n)^T$	the model residuals of the n training observations
$\mathbf{r}_m = (r_{1m}, \dots, r_{nm})^T$	the model residuals of the n training observations after $m - 1$ iterations in a boosted regression tree model
T	a decision tree
T_0, T_1, T_2, \dots	a nested sequence of decision trees with T_0 representing the full (unpruned) tree and T_1, T_2, \dots representing progressively shallower trees in which the terminal node resulting in the smallest decrease in the $\text{RSS}_{\text{train}}$ has successively been pruned from the previous tree in the sequence

$ T $	the size of the tree T , measured in terms of the number of terminal nodes in T
$T(\mathbf{x}_i \Theta_m)$	the prediction for \mathbf{x}_i from the tree fit during the m^{th} iteration of a boosted regression tree model with parameters Θ_m
\mathbf{X} or $\mathbf{X}_{n \times (p+1)}$	the $n \times (p+1)$ matrix with $\mathbf{1}_n$ as the first column and the i^{th} row containing 1 as the first entry, followed by the observed values of the p input variables for the i^{th} training observation \mathbf{x}_i
\mathbf{X} or \mathbf{X}_p or \mathbf{X}_{p+1}	the p -dimensional (p -D) vector of the input random variables $(X_1, X_2, \dots, X_p)^T$ or the $(p+1)$ -D vector $(1, X_1, X_2, \dots, X_p)^T$, which includes an intercept
$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$	the n observations of the input vector in the training data set
x_1, x_2, \dots, x_p	the p input variables in the training data set
Y	the response random variable
y	an observed value of the response
$\mathbf{y} = (y_1, \dots, y_n)^T$	the n responses of the training observations
\bar{y}	the average of the n observed response values in the training data set
\hat{y}	a predicted value for the response
$\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_n)^T$	the predicted responses for the n training observations
$\hat{y}_{R_1}, \hat{y}_{R_2}, \dots, \hat{y}_{R_J}$	the mean responses of the training observations in each of the J regions of a decision tree model
\mathbf{Z} or $\mathbf{Z}_{n \times (p+1)}$	the training data set i.e. $\mathbf{Z}_{n \times (p+1)} = [\mathbf{z}_1, \dots, \mathbf{z}_n]^T = [(\mathbf{x}_1^T, y_1)^T, \dots, (\mathbf{x}_n^T, y_n)^T]^T$
$\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n$	the n observations of the p input random vectors and of the response in the training data set, where $\mathbf{z}_i = (\mathbf{x}_i^T, y_i)^T$
\mathbf{Z}^* or $\mathbf{Z}_{n \times (p+1)}^*$	a bootstrap sample of the training data set i.e. $\mathbf{Z}_{n \times (p+1)}^* = [\mathbf{z}_1^*, \dots, \mathbf{z}_n^*]^T$, where $\mathbf{z}_1^*, \dots, \mathbf{z}_n^*$ are n observations randomly selected with replacement from the training data set

$\mathbf{Z}^{*1}, \dots, \mathbf{Z}^{*B}$	the B bootstrap samples of the training data set
α	the tuning parameter in pruned decision trees
$\boldsymbol{\beta}$ or $\boldsymbol{\beta}_p$ or $\boldsymbol{\beta}_{p+1}$	the vector of true coefficients $(\beta_1, \dots, \beta_p)^T$ or $(\beta_0, \beta_1, \dots, \beta_p)^T$
$\beta_0, \beta_1, \dots, \beta_p$	the true coefficients (based on the population)
$\widehat{\boldsymbol{\beta}}$ or $\widehat{\boldsymbol{\beta}}_p$ or $\widehat{\boldsymbol{\beta}}_{p+1}$	the vector of estimated coefficients
$\widehat{\beta}_0, \widehat{\beta}_1, \dots, \widehat{\beta}_p$	the estimated coefficients
$\widehat{\boldsymbol{\beta}}^{\text{lasso}}$ or $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{lasso}}$	the coefficients estimated by the lasso algorithm, given a specific tuning parameter λ
$\widehat{\boldsymbol{\beta}}^{\text{LS}}$ or $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{LS}}$	the coefficients estimated by the multiple linear regression algorithm using the least-squares approach
$\ \boldsymbol{\beta}_p\ _1$	the ℓ_1 norm of the coefficient vector $\boldsymbol{\beta}_p$ is given by $\ \boldsymbol{\beta}_p\ _1 = \sum_{j=1}^p \beta_j $
$\ \boldsymbol{\beta}_p\ _2$	the ℓ_2 norm of the coefficient vector $\boldsymbol{\beta}_p$ is given by $\ \boldsymbol{\beta}_p\ _2 = \sqrt{\sum_{j=1}^p \beta_j^2}$
γ_{jm}	the constant that, in the m^{th} iteration of a boosted regression tree model, maximises the fit of the training observations in the j^{th} region of the input space to the model
ε	the error term in a statistical model
η	the bag fraction in a boosted regression tree model
Θ_m	the parameters (split variables, cutpoints and predictions) for the tree model $T(\cdot \Theta_m)$ fit during the m^{th} iteration of a boosted regression tree model
λ	tuning parameter in the lasso and in ridge regression
μ_X	the expected value of the random variable X
ν	the shrinkage or learning parameter in a boosted regression tree model
ρ or ρ_{XY}	the correlation between the random variables X and Y
σ_X^2	the variance of the random variable X

$Z \sim N(\mu_Z, \sigma_Z^2)$	Z is normally distributed with a mean of μ_Z and a variance of σ_Z^2
Bias $(\hat{\theta})$	the bias of $\hat{\theta}$ as an estimator of the parameter θ
$\text{Cov}_{XY}(X, Y)$	the covariance between the random variables X and Y
$E_X(X)$	the expected value of the random variable X
F_n	the empirical cumulative distribution function of the random vector \mathbf{X}_p based on n independent and identically distributed observations
$F_{\mathbf{X}}$	the cumulative distribution function of \mathbf{X}_p
$f_{\mathbf{X}}$	the density function of \mathbf{X}_p
f	the function giving the true relationship between \mathbf{x}_p and Y
\hat{f}	the function giving the estimated relationship between \mathbf{x}_p and Y
$\hat{f}_{\text{bag}}^{*b}(\mathbf{x}_p); \hat{f}_{\text{bag}}(\mathbf{x}_p)$	the prediction from the b^{th} bootstrap tree and from the whole bagged regression tree model, respectively, of the response for the observation \mathbf{x}_p
$\hat{f}_{\text{rf}}^{*b}(\mathbf{x}_p); \hat{f}_{\text{rf}}(\mathbf{x}_p)$	the prediction from the b^{th} bootstrap tree and from the whole random forest model, respectively, of the response for the observation \mathbf{x}_p
$f_{m-1}(\mathbf{x}_i); f_m(\mathbf{x}_i)$	the prediction for \mathbf{x}_i from the boosted regression tree model after $m - 1$ and m iterations, respectively
$I(A)$	the indicator function, which returns 1 if A is true, and 0 otherwise
$h(\mathbf{x}_p)$	a transformation of the predictor variables in the training data set
$L\{Y, \hat{f}(\mathbf{x}_p)\}$	a loss function
ℓ_1	the ℓ_1 norm of the coefficient vector $\boldsymbol{\beta}_p$ is given by $\ \boldsymbol{\beta}_p\ _1 = \sum_{j=1}^p \beta_j $
ℓ_2	the ℓ_2 norm of the coefficient vector $\boldsymbol{\beta}_p$ is given by $\ \boldsymbol{\beta}_p\ _2 = \sqrt{\sum_{j=1}^p \beta_j^2}$

$\text{Var}_X(X)$ the variance of the random variable X

Statistical abbreviations

adj. R^2	adjusted R^2
BIC	Bayesian information criterion
BRT	boosted regression trees
BS	bootstrap
BSS	best subset selection
CART	classification and regression trees
CoD	curse of dimensionality
CV	cross-validation
D	dimensional
DCM(s)	dynamic crop model(s)
df	degrees of freedom
EPE	expected prediction error
FDA	functional data analysis
GAM(s)	generalised additive model(s)
i.d.	identically distributed
i.i.d.	independent and identically distributed
lin.	linear
LLN	law of large numbers
MAE	mean absolute error
MAE_{test}	MAE of the test data set
MLR	multiple linear regression
MSE	mean squared error
$\text{MSE}_{\text{train}}$	MSE of the training data set
$\text{MSE}_{\text{validate}}$	MSE of the validation data set
MSE_{test}	MSE of the test data set
OOB	out-of-bag
QQ plot	quantile-quantile plot

quad.	quadratic
RF(s)	random forest(s)
RMSE	root mean squared error
RSE	residual standard error
RSS	residual sum of squares
RSS ₀ ; RSS ₁	the RSS of the training data set resulting from a multiple linear regression model containing $p - q$ (RSS ₀) or all p of the input variables (RSS ₁)
RSS _{R_m} ; RSS _{R_m(j,s)}	the RSS of the m^{th} region in the input space (based on a split of the j^{th} input variable at cutpoint $x_j = s$) in a regression tree model
RSS _{train}	the RSS of the training data set
SE	standard error
SLT	Statistical Learning Theory
TSS	total sum of squares

Chemical symbols

C	carbon
CO ₂	carbon dioxide

Other abbreviations

ARC	South African Agricultural Research Council
EARS-E2M	Environmental Analysis and Remote Sensing Earth Environment Monitoring B.V.
ET	evapotranspiration
ET ₀	reference evapotranspiration
ET _{pan}	pan evapotranspiration
GPS	global positioning system
RE	relative evapotranspiration
RH	relative humidity
SAWS	South African Weather Service

Units of measurement

°C	degree(s) Celsius
cm	centimetre(s)
d	day(s)
g	gram(s)
h	hour(s)
ha	hectare(s)
km	kilometre(s)
m	metre(s)
mm	millimetre(s)
s	second(s)
t	metric tonne(s)

Editorial abbreviations

edn.	edition
e.g.	for example
<i>et al.</i>	and others
i.e.	that is
no.	number
pers. comm.	personal communication
pp.	pages
vol.	volume

Chapter 1

Thesis overview

This chapter contains an outline of the thesis. Brief descriptions of the goals of the study and the methods applied to address these goals are given. This chapter also provides concise explanations of terminology used throughout the document. The objectives of this study are discussed within the broader context of the utility of statistical modelling in agriculture. Finally, the chapter closes with a description of the layout of the rest of the document.

1.1 Problem statement and study objectives

Growing crops in an open-field setting presents diverse problems to a farmer. Crop plants exposed to the open air are more vulnerable to pest infestations, and diseases from nearby plants can decimate crops. However, one of the chief sources of uncertainty in the success of a crop comes from the weather: Wet conditions can flood crops leading to root rot, dry conditions cause desiccation, and high winds damage the plants, to name but a few. Although a farmer cannot control the weather, precautionary measures can be taken to protect crops from impending changes in the weather, and remedial steps can minimise the damage caused by unforeseen weather conditions.

However, minimising the effects of adverse weather on harvest quantities requires knowledge of the effects of weather conditions on crop success. In addition to knowing how to maximise yield in the face of inclement weather, being able to predict crop yields based on weather records from previous weeks and the weather conditions expected over the coming weeks can help in the synchronisation of produce harvest and distribution.

This study investigates the effects of growing conditions on the densities (t/ha) of tomatoes harvested from crops containing closely related tomato cultivars on a commercial tomato farm in South Africa. Statistical models are developed for understanding the effects of weather conditions on harvest densities, and for predicting harvest densities from prevailing weather conditions during the development of the crop plants.

1.2 Clarification of key concepts

The following terms are used throughout the thesis, and are therefore defined here for the reader's convenience:

Field crops and block crops The tomato crops investigated in this project were grown in fields that are subdivided into blocks to facilitate crop management on the farm. The plants grown in a particular field in a particular year constitute a *field crop*, with most field crops containing several *block crops*. Since crops are chiefly managed at the block level, the block crops were chosen as the objects in the modelling analyses.

Growth period, harvest period and crop duration A block crop's *growth period* refers to the length of time between the date that the tomato seedlings were planted out into the field and the first harvest date of the crop. The *harvest period* of a block crop is the time between its first and last harvest dates, and consists of numerous harvest events. *Crop duration* is simply the sum of a block crop's growth and harvest periods.

Yield versus harvest *Yield* is a measure of the quantity of fruit produced by a crop. *Harvest*, on the other hand, is the quantity of fruit picked and delivered to the warehouse. Although the analyses in this document model harvest quantities, a crop's harvest can be viewed as an approximation of its yield (provided that the farmer is relatively consistent in his decisions of when to start and stop harvesting a crop based on yield density (tonnes/hectare), and that most of the produce tends to be harvested).

Harvest quantity, harvest mass and harvest density This study models *harvest quantities* of tomato crops in the form of *harvest densities* to

make the relative successes of different sized block crops more comparable. Two measures of harvest density are modelled: total and median. *Total harvest density* is the cumulative harvest from a block crop over its harvest period i.e. its *total harvest mass*, after correcting for its area. Similarly, *median harvest density* is the median quantity of tomatoes harvested during the crop's separate harvest events i.e. its *median harvest mass*, after correcting for area. Harvest mass is measured in tonnes, block crop areas in hectares, and harvest densities are therefore expressed in tonnes/hectare (or t/ha).

Transpiration, reference and relative evapotranspiration and dekad

Evapotranspiration (ET) measures the evaporation of moisture from the soil, as well as transpiration from the crop plants. *Transpiration* is the loss of water from a vegetative surface to the atmosphere, predominantly during gaseous exchange. Total daily *reference ET* (ET_0) measures the ET from a reference crop, e.g. grass, in terms of water depth (in mm) per day. *Relative ET* (RE), on the other hand, gives the reduction in ET as a result of reduced water availability.

The ET_0 variable used in this study consists of daily readings taken by a ground-based weather station, whereas the RE variable comprises estimates made every ten days (i.e. *dekad* readings) from satellite data.

1.3 Research design and methodology

The harvest quantities total and median harvest density were each modelled using planting, harvest and weather variables. For each weather variable the median, average or minimum of the values recorded over the growing periods of the block crops was calculated, depending on which summary statistic was most strongly correlated with the two response variables. The cleaned crop and weather data set was partitioned into a training, validation and test set. Preliminary analyses were conducted on the training set to investigate the relationship between each predictor and the response variables, and a subset of the original variables was selected as input variables for the statistical modelling analyses.

The harvest densities were analysed using several different statistical learning theory algorithms in an effort to understand the effects of weather conditions on crop success and to predict harvest densities accurately. The algorithms applied were multiple linear regression, the lasso, regression trees, bagged regression trees, random forests and boosted regression trees. The accuracy of each model in predicting unseen cases was assessed using the validation set.

The results of the different modelling techniques were compared, and an independent estimate of the accuracy of the best model for each response was obtained using the test set. The most important influences on harvest density are discussed, and the models providing the highest prediction accuracies are highlighted.

1.4 Importance of the study

Since the models in this study were fit to data from a particular tomato farm, these models are of primary interest to the the tomato farm under investigation. However, the cultivars included in this study constitute tomato cultivars commonly grown for commercial purposes. Moreover, the weather variables included in the analyses (temperature, rainfall, relative humidity, wind speed and evapotranspiration) are amongst the most commonly recorded weather variables for agricultural and other purposes. Consequently, the insights regarding the effects of various growing conditions on tomato yield gleaned from these analyses are widely applicable. Furthermore, the results of the statistical analyses provide indications of the applicability of these algorithms to modelling harvest densities in tomatoes, as well as in other agricultural crops. This study therefore contributes towards the field of modelling harvest densities for agricultural applications.

1.5 Chapter outline

The following chapter (Chapter 2) contains a review of the cultivation of tomato crops. The discussion then shifts towards methods that have been used for predicting the yield of tomato and other crops, and describes some of the weather variables that have been used for this purpose. Some theoretical

background on the statistical modelling approaches applied in this project is then given, before closing with the objectives of the study.

Chapter 3 contains a preliminary investigation of the data. The chapter starts with a description of the planting and harvest data obtained from the tomato farm, and the division of the crop data into the training, validation and test sets. The next section describes the weather data acquired from a nearby weather station and estimated from satellite data, and the integration of the weather data into the training, validation and test sets. The harvest and weather variables are then analysed together in order to choose the most informative variables for the modelling analyses.

Chapters 4 to 6 present the results obtained using the statistical modelling algorithms. In Chapter 4, multiple linear regression and the lasso are applied, the latter involving the regularisation of the multiple linear regression coefficients. The analyses then shift towards nonlinear statistical modelling techniques in Chapter 5, which contains methods based on decision trees. These methods include regression trees, bagged regression trees and random forests. Chapter 6 concludes the statistical modelling chapters with the more complex, black-box approach—boosted regression trees—which tends to produce accurate (although less interpretable) statistical models.

Chapter 7 contains a discussion of the findings of the statistical analyses. The most accurate models for predicting harvest density are identified and the most important predictors are discussed. The limitations of the study are outlined, and suggestions for alternative approaches and techniques for future work are also provided.

Chapter 2

Literature Review

This study involves modelling tomato crop harvest density using weather and other predictors that reflect the growing conditions of the crops. This chapter provides the theoretical background for the subsequent analyses, containing an overview of tomatoes as a commercial crop, a review of methods applied so far for modelling crop yield and a discussion of the statistical modelling techniques performed in this project. The chapter ends with the aims of the study.

2.1 Tomatoes as an agricultural crop

Except where otherwise stated, the reference for this section is Heuvelink (2005*b*).

The commercial tomato cultivars of today all find their origins in the species *Solanum lycopersicum* L. (The International Plant Names Index), a native of Central and South America. Tomatoes were domesticated and used as a food crop in Mexico, before being introduced to Spain and its colonies by the conquistadors during the 16th century. From Spain, tomatoes spread to Italy and gradually to other major European countries. Tomatoes were initially considered poisonous due to their bright red colour, and were therefore grown mainly as an ornamental plant. However, by the mid-16th century tomatoes were being cultivated as a food crop in southern Europe. Europeans introduced the tomato to many parts of Asia in the 17th century and to North America and Japan in the 18th century. By the end of the 18th century tomatoes were also a widespread food crop in north-western Europe. Today tomatoes are

consumed worldwide, their popularity at least in part due to their versatility (they can be used in sandwiches, salads, pizza, ketchup, sauces, drinks, etc.). Tomatoes are amongst the world's top 15 food crops, with well over a 100 million metric tonnes produced annually (Food and Agricultural Organization of the United Nations statistical database). China is currently the world's chief producer and consumer of tomatoes.

2.1.1 Tomatoes for fresh consumption

Tomato cultivars that produce fruit destined for the fresh market are perennial indeterminate vines that can grow for up to three years under suitable conditions, but are usually grown as an annual in temperate regions. Seedlings are cared for in greenhouses before being transplanted to their final destination in the open field or in protective structures (greenhouses, nethouses, plastic tunnels, etc.). The plants are provided with adequate moisture and nutrients regularly. Depending on the cultivar, the first inflorescence appears 21 days after transplanting under optimal conditions. Indeterminate varieties can reach a length of 180 cm or more, and therefore require support in the form of poles linked with tightly-stretched twine. Plants are typically tied for the first time when they are 30 cm to 40 cm tall, and every 25 cm to 30 cm thereafter. Some of the side branches are pruned when 7 cm to 10 cm long in order to increase fruit size and shorten the growing period, and the fruit is kept clear of the soil by tying. Towards the end of the growing period tomato plants are decapitated (the growing tip is removed) in order to increase the proportion of resources that the plant allocates to the fruit. Fruit is borne in clusters along the stems. The fruit develop over a protracted period of time, resulting in a harvest period of several weeks or a few months. The tomatoes are harvested manually in order to minimise damage to the fruit. The level of ripeness at which the tomatoes are harvested depends on, amongst others, the desired quality of the fruit and the distance of the market from the production area. Growing cultivars for fresh consumption is therefore labour-intensive and costly. That being said, good-quality fruit can be sold at higher prices in the market.

The tomato plant evolved in the tropical highlands of Central and South America, and therefore does best in warm climates. Tomatoes need 90 to 120 days of frost-free weather and an average temperature above 16 °C, and suffer damage to their vegetative and reproductive parts (including the fruit) if

exposed to temperatures from 0 °C to 12 °C for prolonged periods. *Chilling injury* causes growth to slow and the foliage to discolour or develop lesions. Chilling of the roots results in the plant wilting. Optimum daytime temperatures are 25 °C to 30 °C, while optimum night-time temperatures are 16 °C to 20 °C. Waterlogging and other causes of anaerobic conditions around the roots induce the downward curvature of the leaves.

2.1.2 Stages in the development of tomato plants

After the tomato seed begins to grow, the taproot emerges from the seed. Adventitious roots subsequently emerge along the decumbent stem, augmenting the existing root system. The roots of a tomato plant can extend to a depth of up to 2 m, although the most active part of the root system is close to the soil surface. Factors affecting root development include atmospheric humidity as well as soil temperature, moisture and nutrient content.

The young stem grows from an apical bud at the tip of the main stem. Seven to eleven alternate, compound leaves form before the initial apical bud is converted into a terminal inflorescence. The lateral bud in the apex of the most recently formed leaf then becomes the new apex from which stem elongation takes place and new leaves and flower parts are initiated. Despite the fact that stem elongation proceeds from a lateral bud, growth occurs along the initial axis since the newly-formed terminal inflorescence is displaced into a lateral position. This succession of lateral growths is repeated for the rest of the plant's life, with the current apical bud becoming a terminal inflorescence and a lateral bud assuming the role of the apical bud after the formation of every three leaves. The elongation rate of the resulting sympodial stem is influenced by, amongst others, daytime temperature and light intensity. Temperatures a little over 28 °C are optimal and low daily irradiances result in longer but thinner (and therefore weaker) stems. Temperature (24 h mean temperature), and to a lesser extent light intensity, are also critical in determining the rate of leaf appearance, which in turn limits the rate of flower initiation (since one inflorescence is formed for every three leaves initiated). Flower initiation is also influenced by environmental factors such as light intensity, ambient temperature, carbon dioxide concentrations, nutrient concentrations in the soil and soil moisture. Leaf size is highly variable, with an increased *photoassimilate* (photosynthate or photosynthesised sugars) supply generally leading to larger

and heavier leaves. However, temperature and light intensity also play crucial roles in leaf size.

After the initiation of an inflorescence, which is in the form of a monochasial cyme terminated by a king flower, its flower buds must develop into flowers, which in turn must be pollinated. Fertilisation and fruit set must also occur, each involving several steps, before the inflorescence can bear fruit. The number of flowers in an inflorescence is determined by the cultivar, as well as by temperature, light intensity, water availability and plant density. Flower development speeds up with an increase in daytime temperatures, although higher daytime temperatures can also increase the rate of flower abortion in the presence of low photosynthate levels. *Pollination* is accomplished when pollen is successfully transferred to the stigmatic surface of the flower, whereas *fertilisation* involves the growth of the pollen tube down from the stigmatic surface into the ovule where male and female gametes unite. Pollination in modern tomato cultivars occurs chiefly in the form of self-pollination. Insect pollinators are therefore not essential for fruit development; some air movement is sufficient for the pollen of a flower to be transferred to its stigmatic surface. Relative humidity levels between 50 % to 90 % are optimal for pollination—too high and the pollen tends to remain in the anthers, and too low and the pollen does not stick to the stigmatic surface. A temperature range of 17 °C to 24 °C is optimal for pollination. In comparison to pollination, fertilisation is largely independent of environmental factors. *Fruit set*, which Heuvelink (2005*b*) uses to refer to the proportion of flowers that develop into fruit of an acceptable (marketable) size, is primarily determined by the quantity of photoassimilate available, but is adversely affected by high temperatures (over 40 °C). The period from the opening of the flower (*anthesis*) to the harvesting of the fruit typically takes about 1.5 months to 2.5 months, and is primarily dependent on temperature. Tomatoes are typically picked before they are physiologically ripe to prolong their shelf-life. Fresh fruit mass ranges from about 15 g for cherry tomatoes (a tomato type that is small and sweet) to 500 g for beefsteak tomatoes (a tomato type larger than the traditional round tomato).

2.1.3 Factors affecting tomato yield

Tomato crop yield is determined by *total/cumulative biomass production*, as well as how the biomass is partitioned amongst the organs of the plant (*biomass*

partitioning). Total biomass production is chiefly determined by the amount of photosynthesis that the crop plants have been able to carry out during the growing period i.e. by the quantity of photoassimilate available (*source strength*), which is influenced by factors such as the health of the plants, the length of the growing season, temperature, humidity, light interception and cultural practices such as row spacing and pruning. However, a high overall biomass production does not guarantee a good yield, since it is only the biomass allocated to the fruit that determines the profitability of the crop. A poor yield can be the result of a low fruit set, which can be caused by poor pollen production or the failure of pollination, pollen germination, pollen-tube growth, ovule production or fertilisation to occur. Low yield can also be caused by a poor *fruit sink strength* i.e. the fruit is underdeveloped due to its poor ability to import photosynthate from the rest of the plant. Fruit sink strength is determined by the number of seeds that the fruit contains relative to the other fruit on the plant. Environmental factors that affect fruit sink strength include temperature and light intensity.

Fruit size is strongly affected by photoassimilate availability and temperature during the development of the fruit. Since ripening is accelerated by higher temperatures, high temperatures tend to result in smaller fruit since the fruit has less time in which to grow. Tomatoes tend to remain small if the plants are carrying a heavy fruit load, and the size of the remaining fruit can be increased by fruit pruning. Fruit size can also be improved by pruning vegetative parts of the plant, especially older, shaded stems and leaves, since this increases the proportion of resources sent to the fruit. Subjecting plants to water and salinity stress are commonly used tactics for stimulating fruit formation. However, too much stress early in a plant's life will hinder its ability to continue with both vegetative (growth of roots and stems and leaves) and generative (growth of flowers and fruit) growth, thereby reducing the future yield of the plant.

2.2 Predicting crop yield

Harvest prediction is an important tool for managing a commercial farming enterprise effectively, affecting crop management, allocation of farming resources, customer interaction, and produce pricing, amongst others. Insight into fac-

tors affecting yield enables the optimisation of crop management practices, and the mitigation or counteraction of detrimental environmental influences. Generating accurate harvest predictions for the crops currently under cultivation facilitates the allocation of a farm's resources, enabling more precise planning of labour, warehouse space and transportation of the produce, to name but a few. Moreover, accurate harvest predictions enable customers (wholesalers and retailers) to be given more advanced notice of anticipated delivery times and quantities. The effects of produce shortage or surplus in the foreseeable future can be reduced by adjusting the price of the produce in advance.

Empirical statistical models and process-based models (dynamic crop models or DCMs) constitute the two main approaches for modelling crop yield in a field.

2.2.1 Empirical statistical modelling techniques for predicting crop yield

Empirical statistical models use observations of the relationship between variables describing the growing conditions of the crop of interest (weather variables, soil characteristics, etc.) and the resulting yield to model crop yield. The model coefficients are fit to historical crop data using statistical techniques such as multiple linear regression (e.g. Stanger and Lauer, 2008; Kotsiri *et al.*, 2014) and neural networks (e.g. Irmak *et al.*, 2006; Ehret *et al.*, 2011). Simplified weather measurements such as average temperature and precipitation over fixed periods of time (e.g. weeks, ten days, months) or over the different crop stages (e.g. seedling establishment, vegetative growth, flower formation, fruit formation, fruit ripening) are usually used.

Statistical models have the distinct advantage over DCMs in that they are relatively easy to fit, and can therefore be retrained regularly to incorporate the latest yield data. They, however, also have disadvantages. Since weather variables tend to be highly correlated, statistical models tend to suffer from multicollinearity. Multicollinearity is, however, more of a hindrance to inference than to prediction. Since crop yield is a complex process that is affected by a multitude of different factors, statistical models also tend to suffer from dimensionality problems. Consequently, variable selection is an important step in modelling crop yield. A further disadvantage is that, since the model co-

efficients are trained exclusively on historical data records, these statistical models are sensitive to noise in the crop data. Data cleaning is therefore very important for enhancing model accuracy.

2.2.2 Dynamic crop prediction models

DCMs use mathematical equations and model constants specific to a particular crop plant or cultivar to emulate physiological, physical and chemical processes. They use this information to simulate the growth, development and yield of the crop plants, usually in daily time increments, from weather time series (hourly, daily or weekly), soil and crop management variables, and sometimes even genetic information. Their complex model structure means that different stages in crop development can be modelled differently, which is advantageous since different developmental stages are sensitive to different conditions. Examples of DCMs developed for predicting yield include WOFOST (Van Keulen and Wolf, 1986; Van Diepen *et al.*, 1989), APSIM (Probert *et al.*, 1995; McCown *et al.*, 1995), CropSyst (Stöckle *et al.*, 2003), DSSAT (Jones *et al.*, 2003) and AquaCrop (Steduto *et al.*, 2009; Raes *et al.*, 2009). CROPGRO-Tomato (Scholberg *et al.*, 1997), TOMPOUSSE (Gary *et al.*, 1997) and TOMSIM (Heuvelink, 2005a) are examples of DCMs developed specifically for tomatoes.

The model structure of DCMs is based on what is known about the physiology, reproduction, agronomy, soil science and agrometeorology of the crop plant of interest. This makes DCMs applicable to all crops of the same cultivar and grown under the conditions accommodated by the model (as opposed to empirical statistical models, which can only be applied to the crop from which the training data was obtained). Typical inputs of DCMs include weather time series (most commonly solar radiation, temperature and precipitation), soil properties (e.g. pH, organic carbon) and crop management (planting date, planting density and irrigation and fertiliser regimes). Such detailed data are difficult to obtain outside of research settings. Furthermore, their wider use in agricultural circles has been seriously curtailed by their need for cultivar-specific information, which are inadequately represented in existing crop models.

Although custom-made DCMs tend to provide relatively accurate predictive models, they require many model constants for physiological processes in tomato plants specific to the cultivar for fitting the model, many of which

may not yet have been published. DCMs therefore require a lot of initial work to train—at least some of which has to be repeated as agricultural practices change (different cultivars grown, a shift from field-grown crops to crops grown in protective structures, conversion from a sprinkler system to drip irrigation, a change in the fertiliser applied, etc.). Consequently, DCMs were not investigated in this current study; attention was focused on the development of empirical statistical models, which have less of a biological and more of a statistical focus.

2.3 Commonly recorded weather variables

The information for this section was obtained from Ahrens (2008), Tyson and Preston-Whyte (2000) and Taiz and Zeiger (2002). Allen *et al.* (1998) was used in writing the section on evapotranspiration.

2.3.1 Air temperature

Temperature measures the average speed at which the atoms and molecules in a substance are travelling—the higher the average speed, the higher the temperature. When a substance is heated, its component molecules speed up and move further apart, and the substance expands and becomes less dense. Cooling the substance has the reverse effect. Consequently, temperature does not only determine the average kinetic energy of the molecules, but also the density of the substance. Temperature is most commonly measured on the *Celsius scale* for meteorological purposes.

Air temperature is a very important environmental factor influencing the development of plants. Plants deplete their carbon (C) reserves during *cellular respiration*,¹ and replenish C stores from atmospheric carbon dioxide (CO₂) through their *stomata*² during *photosynthesis*.³ As ambient temperature in-

¹*Cellular respiration* is the process of metabolising carbohydrates, which are C-rich molecules, in order to use the energy contained in their chemical bonds for the chemical reactions that are necessary for the ongoing survival of the plant. Plants and other living organisms therefore use carbohydrates for, amongst other things, the long-term storage of energy, which is necessary for all cell functioning.

²*Stomata* (singular: “stoma”) are tiny apertures in the leaf surface that allow the plant to exchange gases (CO₂, oxygen and water vapour) with the atmosphere.

³*Photosynthesis* is a metabolic process during which the energy from solar radiation is used to produce carbohydrates from the C atoms in atmospheric CO₂. The energy captured

creases, stomata close to minimise water loss. This cuts off the supply of atmospheric CO₂, resulting in the levels of C-based molecules such as carbohydrates to plummet. High temperatures over prolonged periods therefore tend to result in retarded plant growth and less sweet fruit and vegetables.

2.3.2 Wind speed

Wind refers to the horizontal movement of air from an area of high atmospheric pressure to one with a lower atmospheric pressure, which is usually caused by temperature differences. *Wind speed*—the rate at which the wind is moving relative to a stationary observer—is measured in distance per unit of time, e.g. metres per second (m/s). The wind is an important dispersal mechanism for heat energy, water vapour and pollen across the Earth's surface, but can cause extensive damage if too strong.

2.3.3 Relative humidity

The Earth's atmosphere is unique amongst the known planets in that it contains water vapour and experiences temperature ranges that allow water to exist in its solid, liquid and gaseous forms. Water evaporates from water bodies and from the Earth's surface to form water vapour, and the atmosphere contains about 0.35% of the Earth's water at any given time. *Humidity* gives the water vapour content of the air, which can be expressed in many different ways.

The most commonly reported measure of humidity is relative humidity. *Relative humidity* (RH) indicates how far away from saturation the air is at its current temperature and pressure, reported as a percentage. Air is *saturated* (has 100% RH) when it cannot hold any more water vapour—in other words, when the rate of evaporation from a ready source of water is equal to the rate of condensation back into the water source. If a wet surface is exposed to saturated air, then it will remain moist, whereas if it is exposed to unsaturated air, then the moisture will evaporate. Thus, as air approaches saturation, the rate of condensation increases, whereas evaporation predominates in less saturated air. Hence, RH does not give the absolute water content of the air

in the chemical bonds of carbohydrates can later be broken down during cellular respiration when and where the energy is needed in the plant.

(which is not so important), but rather how much water the air can still take up, and therefore how much evaporation is expected to take place.

RH and temperature are inextricably linked. In warmer air, the water molecules have a higher average kinetic energy. Consequently, the water vapour molecules are more likely to remain in gaseous form, reducing the rate of condensation and therefore decreasing the RH. The opposite occurs in colder air. RH can also be changed by adjusting the water content of the air. However, the atmospheric water content of more inland regions does not change much during the course of a day, and it is predominantly the temperature that determines RH: At night, the air cools down and the RH therefore increases, peaking around dawn. RH then decreases throughout the day, reaching a minimum in the afternoon when temperatures are at their hottest.

2.3.4 Rain

Rainfall is measured in length per unit time, e.g. millimetres per day (mm/d), and gives the depth of rain that would accumulate if it had fallen on an impermeable surface over that length of time. *Rain intensity* describes (in words rather than in numbers) the amount of rain received in a unit of time.

2.3.5 Evapotranspiration

Evapotranspiration (ET) refers to the loss of water from the crop to the atmosphere. This happens in two ways:

- evaporation of water from the soil surface, and
- transpiration from the aerial parts of the plant.

Evaporation refers to the transformation of water from its liquid to gaseous form and its removal from the evaporating surface. *Transpiration* refers to the evaporation of water from a plant through its stomata into the atmosphere. Water molecules escape through these tiny openings into the drier air that comes into contact with the leaf surface. This, in turn, causes the underlying leaf tissue to dry out, which causes water to flow from the stem to replace the water lost, which is in turn replaced by water absorbed through the roots. Hence the evaporation of water vapour from the leaves causes a potential difference that drives the movement of water from the soil into the roots, up

the stem and out via the leaves. In this way, water can be raised well over 100 m above the ground (in the case of trees), driven by the large potential difference between the leaves and roots and supported by the cohesiveness of the hydrogen bonds that exist amongst the water molecules and the adhesiveness between water and the surface of the very thin water-conducting tissue in the plant. Provided that there continues to be enough moisture in the soil to replace the water lost to the atmosphere, this column of water continues to be dragged unbroken up through the plant and out via its leaves. However, if the moisture in the soil dries up, then this water column breaks and the plant wilts due to lack of water.

Water plays several crucial roles within a plant. Nutrients become transportable by being dissolved in water. Within plant cells, water serves as the medium that enables cell components to be brought into contact with other cell components and dissolved nutrients, water serves as the medium in which chemical reactions take place, sometimes itself being a reagent in the chemical reactions, and water provides structural support to the cell (through turgor pressure) and therefore to the plant as a whole.

Although water is vital to a plant's survival, transpiration (which results in water loss) is equally important, since it is the main means by which water is transported throughout the plant. The water taken up by the roots contains dissolved nutrients from the soil, which are dispersed throughout the plant via this water column. Plants also use water loss as a cooling mechanism. Since water vapour possesses more energy than does liquid water, evaporation requires a net input of energy for the water molecules to break free of the hydrogen bonds that hold water molecules together in their liquid form. Consequently, evaporation (and therefore also transpiration) has a cooling effect, since some of the energy required to evaporate the water comes from the surface from which it is evaporating. High humidity levels and water shortage both slow down water loss (high humidity reduces the potential difference between the inside of the plant and the atmosphere, thereby reducing evaporation, and water shortage leads to stomatal closure, which also reduces water loss), thereby reducing the plant's ability to dissipate heat. This can cause plant tissue to rise in temperature by up to 4°C or 5°C in the midday sun. Overheating in turn impairs photosynthesis as well as cellular functioning and maintenance in general. Transpiration is therefore essential for a plant's

survival.

The extent to which plants are able to continue functioning during water shortages depends on the water-use efficiency of the plant. Most plants possess adaptive mechanisms that enable them to acclimatise to water shortages. For example, a long-term onset of water stress leads to the inhibition of leaf expansion in a plant. This reduces the plant's surface area, which in turn reduces water loss. If water shortages only develop later in a plant's development, then the plant can abscise leaves in order to reduce its surface area. Although these strategies increase the chances of the plant surviving, they are not conducive to high crop yields, since less foliage results in lower rates of photosynthesis and therefore lower productivity.

ET is affected by weather conditions such as temperature, humidity and wind speed. Higher temperatures increase the speed of the water vapour molecules, thereby making it much more likely that they will transform into their gaseous form. Hence, high temperatures increase the rate of evaporation (and therefore also transpiration), and on a hot, sunny day, a leaf can exchange up to 100% of its water per hour. However, evaporation occurs more slowly in high humidity due to high condensation rates, and if both the temperature and RH are high (which occurs in hot regions close to large bodies of water), a plant can more easily overheat. A breeze can offset the effects of extreme temperatures on plants. Air movement removes the humid layer that forms around the leaves during the day when they are transpiring, enabling more evaporation to take place, thereby facilitating cooling. However, a persistent wind can also result in crop plants wilting due to excessive water loss if there is insufficient moisture to replace that lost through transpiration.

Environmental and biological factors also affect ET. Soil characteristics determine the capacity of the soil to hold on to and replenish moisture. Access to water sources such as a shallow water table, regular irrigation and/or frequent rain replenish moisture levels in the soil, fuelling ET. However, in the absence of a ready supply of water, the soil can quickly dry out, resulting in the crop plants wilting and ET slowing down or even stopping. Crop characteristics influencing ET include the crop type, developmental stage, foliage density and type of foliage (e.g. succulent, sclerophyllous (woody) or herbaceous).

ET levels therefore reflect the capacity of the atmosphere to absorb water from the crop's habitat, thereby indirectly reflecting the ability of the crop

plants to photosynthesise and maintain optimum tissue temperatures. ET therefore reflects numerous aspects of a crop's well-being, and is an important variable for predicting crop yield. ET is reported in several different ways. *Reference ET* (ET_0) refers to the ET 2 m above an extensive reference crop over a prescribed period. A popular reference crop is an actively growing lawn of grass of a uniform height of 0.12 m and an *albedo*⁴ of 0.23 that completely shades the underlying soil surface and is given sufficient water (sufficient to replace the water lost by ET) about once a week. Since the crop is fixed and grown under standard, controlled conditions, ET_0 conveys the evaporative demand of the atmosphere during that period. ET_0 is measured in length per unit time, e.g. mm/d. ET_0 can be estimated as *pan ET* (ET_{pan}), which is the height of water evaporated from a pan of a specific diameter, depth and water level in the given period. ET_{pan} is then converted to ET_0 by

$$ET_0 = K_{\text{pan}} \cdot ET_{\text{pan}},$$

where K_{pan} is the pan coefficient determined by the pan colour, size, position and ground cover (see Allen *et al.*, 1998). In contrast, *relative evapotranspiration* (RE) gives the relative reduction in ET, reported as a percentage. The shortfall (i.e. $RE < 100\%$) is due to factors such as insufficient soil moisture to replenish the water lost via ET.

2.4 Statistical modelling techniques

Unless otherwise stated, the material in this section was obtained from Hastie *et al.* (2009) and James *et al.* (2013).

2.4.1 Regression

Regression is used to model the relationship between a quantitative *outcome variable* or *response* Y and the quantitative and/or qualitative *input variables* or *predictors* $\mathbf{x}_p = (x_1, x_2, \dots, x_p)$ in the training data set in the presence of additional, unrecorded influences on the response. This is a realistic assumption, since it is seldom possible to analyse all of the manifold factors

⁴*Albedo* refers to the fraction of incident radiation that is reflected off the surface of interest i.e. the reflectivity of the surface. Freshly fallen snow has an albedo of about 0.95, thick clouds 0.6 to 0.9, green vegetation 0.20 to 0.25, and a dry, ploughed field 0.05 to 0.20.

affecting a variable. The nondeterministic relationship between the collected input variables and the response is usually expressed using the *additive error model*

$$Y = f(\mathbf{x}_p) + \varepsilon \quad (2.4.1)$$

in which f is a function of the input variables and $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ represents random noise caused by measurement error and unobserved influences on the response. It is typically assumed that ε is independent of \mathbf{x}_p . Under these assumptions,

$$f(\mathbf{x}_p) = E_{Y|\mathbf{X}}[Y | \mathbf{X}_p = \mathbf{x}_p]. \quad (2.4.2)$$

Hence, the function being modelled is the conditional mean of the response (conditional on the input variable values), and Y only depends on \mathbf{x}_p through f . Furthermore, the additive error model assumes that all of the influences on Y not included in $f(\mathbf{x}_p)$ can be contained within ε . The error terms ε are modelled to be independent and identically distributed, which implies that the variance of Y , σ_ε^2 , is independent of the value of \mathbf{x}_p .

The total variance in Y is therefore partitioned between two terms in (2.4.1). The true relationship between the input variables and the response is denoted by $f(\mathbf{x}_p)$, which contains the variance in the response explained by changes in the input variable values. This is the relationship of interest, and is estimated by modelling the *signal* in the data. The *random error* term ε is independent of \mathbf{x}_p and represents the remaining variance in Y i.e. that part that cannot be explained by $f(\mathbf{x}_p)$. This additional variance is of less interest, since it is caused by unobserved predictors and *measurement error* during data collection, and constitutes the *noise* in the data. It has the effect of randomly perturbing the observations relative to the positions based on their measured input variable values. The relative influence of the signal and the random noise on the recorded measurements in a data set is known as the *signal-to-noise ratio*.

The true relationship f is usually unknown, and has to be estimated or *learnt* from a data set of n random observations of the p input variables and the response drawn from the underlying population—the *training data set*—in such a way that \hat{f} mostly reflects the general patterns present in the greater population—the signal—and mostly ignores the patterns specific to the training data set—the noise. If the signal is stronger than the noise in the training data set, then this can, in part, be accomplished by regulating the *bias-variance*

trade-off of models. A rigid model will tend to have high *bias*, since it will ignore much of the information in the training data set, resulting in the model predictions being far away from the true response values. On the other hand, a model that incorporates too much of the information unique to the training sample into its structure will have high *variance*. Those relationships shared by the general population will be less clear due to the model coefficients being influenced to a large extent by the patterns specific to the training data set, and the model is *overfit*. Such a model will incorporate relationships not present in the larger population into its structure, leading to lower predictive capabilities. Optimising model flexibility is therefore a crucial part of the modelling process.

The statistical techniques described in the rest of this chapter differ in that they each have a different function space from which they choose \hat{f} to approximate f in (2.4.1) on the previous page. Different methods will work best, depending on the nature of the underlying relationship, and there is no best model for all regression problems. A diversity of techniques should therefore be applied.

2.4.2 Multiple linear regression

2.4.2.1 The multiple linear regression model

One of the oldest and most commonly encountered supervised statistical modelling techniques in a regression setting is *multiple linear regression* (MLR), which involves modelling the response Y using a function *linear* in the *model coefficients* or *parameters*. This means that the MLR model assumes that a one-unit increase in the value of the input variable x_j will result in a constant change in Y , irrespective of the value of x_j .

Given the collected input variables, the most straightforward form of the

MLR model, i.e.

$$\begin{aligned}
 Y &= f(\mathbf{x}_p) + \varepsilon = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \\
 &= \beta_0 + \sum_{j=1}^p \beta_j x_j + \varepsilon \\
 &= \begin{bmatrix} \beta_0 & \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} + \varepsilon \\
 &= \boldsymbol{\beta}^T \mathbf{x} + \varepsilon,
 \end{aligned} \tag{2.4.3}$$

where $\beta_0, \beta_1, \beta_2, \dots, \beta_p$ are $p+1$ constants, is linear in both the coefficients and the input variables. The model (2.4.3) represents the best linear approximation (smallest MSE) of the true relationship between the response and the input variables in the original input space (the input space containing no transformed predictors), which is usually unknown. The corresponding fitted model is

$$\begin{aligned}
 \hat{Y} &= \hat{f}(\mathbf{x}_p) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p \\
 &= \hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j x_j \\
 &= \hat{\boldsymbol{\beta}}^T \mathbf{x},
 \end{aligned}$$

which is an approximation of (2.4.3) arrived at by applying an optimisation criterion to the training observations (see (2.4.6) on page 24). If all of the input variables are continuous variables, this fitted model forms a p -D hyperplane in the $(p+1)$ -D *input-output space* with its height above the p -D *input space* at any \mathbf{x} given by $\hat{\boldsymbol{\beta}}^T \mathbf{x}$ (see Section 2.4.12 on page 62 for a description of the input-output space). The *intercept* $\hat{\beta}_0$ specifies the height of the response surface at $\mathbf{x}_p = \mathbf{0}_p$. The *slope coefficients* $\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p$ each specifies the expected change in the response resulting from a one-unit increase in the corresponding input variable when the other input variable values are held constant.

The MLR model in (2.4.3) constitutes a relatively rigid model, especially if p/n is low, since it only accommodates *additive* relationships amongst the p input variables. In other words, it assumes independence of each input variable's influence on the response by assuming that a unit increase in x_j results in the

response increasing by β_j , irrespective of the values of the other input variables. This model can, however, be made more flexible by incorporating *transformations* of the input variables, e.g. *quadratic terms* $x_j^2, j \in \{1, 2, \dots, p\}$, and *interaction terms* $x_i x_j, i \neq j$. A more general form of the MLR model can therefore be expressed as

$$\begin{aligned} Y &= \beta_0 + \beta_1 h_1(\mathbf{x}_p) + \beta_2 h_2(\mathbf{x}_p) + \dots + \beta_M h_M(\mathbf{x}_p) + \varepsilon \\ &= \beta_0 + \sum_{m=1}^M \beta_m h_m(\mathbf{x}_p) + \varepsilon, \end{aligned} \quad (2.4.4)$$

where $h(\mathbf{x}_p)$ is a prespecified *transformation* of the input variables $\mathbf{x}_p = (x_1, \dots, x_p)$. The resulting model is still linear in the model (or regression) coefficients, implying that a change in any of the quantities $h_1(\mathbf{x}_p), h_2(\mathbf{x}_p), \dots, h_M(\mathbf{x}_p)$ results in a fixed change in the response, specified by the slope coefficients $\beta_1, \beta_2, \dots, \beta_M$. It is, however, *not* linear in the input variables x_1, x_2, \dots, x_p . The quantities $h_1(\mathbf{x}_p), \dots, h_M(\mathbf{x}_p)$ are modelled to have additive relationships amongst one another. If $M = p$ and $h_m(\mathbf{x}_p) = x_m$ for all $m \in \{1, \dots, p\}$, then (2.4.4) is equivalent to the special form of the MLR model in (2.4.3). However, other transformations of the input variables such as $h_m(\mathbf{x}_p) = x_j^2, m \in \{1, \dots, M\}$ and $j \in \{1, \dots, p\}$, or $h_m(\mathbf{x}_p) = x_i x_j, i \neq j$, can also be added, resulting in a more flexible curved response surface (rather than a hyperplane) of $\leq p$ dimensions in the $(p + 1)$ -D input-output space.

Although the rest of the discussion in this section will centre around the MLR model in (2.4.3), all of these concepts are equally applicable to the more general MLR model in (2.4.4).

2.4.2.2 Fitting a multiple linear regression model

Fitting the MLR model in (2.4.3) on the preceding page involves estimation of the $p + 1$ model coefficients $\beta_0, \beta_1, \beta_2, \dots, \beta_p$. Consequently, modelling the response by MLR reduces the problem of finding a completely unknown function f in (2.4.1) on page 19 to the estimation of a fixed number of model coefficients in (2.4.3), since the structure of f is already fixed. This reduction in the dimensionality of the model's function space is the rationale behind parametric modelling.

Let $\mathbf{X}_{n \times (p+1)} = [(1, \mathbf{x}_1^T)^T, \dots, (1, \mathbf{x}_n^T)^T]^T$ be the matrix with n rows, with the i^{th} row containing the observed values of the p input variables for the

i^{th} training observation \mathbf{x}_i , with a 1 in the first position. The MLR model is usually fit by the *least-squares approach*, which finds the coefficient vector $\widehat{\boldsymbol{\beta}}_{p+1} = (\widehat{\beta}_0, \widehat{\beta}_1, \dots, \widehat{\beta}_p)^T$ that minimises the vertical distance between the resulting predicted values

$$\begin{aligned}\widehat{\mathbf{y}}_n &= \mathbf{X}_{n \times (p+1)} \widehat{\boldsymbol{\beta}}_{p+1} = \mathbf{X} \widehat{\boldsymbol{\beta}} \\ &= (\widehat{y}_1, \widehat{y}_2, \dots, \widehat{y}_n)^T\end{aligned}$$

and the true values $\mathbf{y}_n = (y_1, y_2, \dots, y_n)^T$ of the training observations. The *model residuals* are then

$$\begin{aligned}\mathbf{r}_n &= \mathbf{y}_n - \widehat{\mathbf{y}}_n \\ &= (r_1, r_2, \dots, r_n)^T.\end{aligned}$$

The vector $\widehat{\mathbf{y}}_n = \mathbf{X} \widehat{\boldsymbol{\beta}}$ consists of a linear combination of the columns of \mathbf{X} , and therefore lies in the *column space* of \mathbf{X} . In contrast, \mathbf{y}_n usually lies outside of the column space of \mathbf{X} , and therefore cannot be reached by a linear combination of the columns of \mathbf{X} . It can, however, be approximated. Since the shortest distance between a point and a hyperplane is the line that passes through the point perpendicular to the hyperplane, the shortest \mathbf{r}_n will always be perpendicular to $\widehat{\mathbf{y}}_n$ with the repercussion that

$$\begin{aligned}\|\mathbf{y}_n\|^2 &= \|\widehat{\mathbf{y}}_n\|^2 + \|\mathbf{r}_n\|^2 && \text{(by Pythagoras' Theorem)} \\ \implies \sum_{i=1}^n y_i^2 &= \sum_{i=1}^n \widehat{y}_i^2 + \sum_{i=1}^n r_i^2 \\ \implies \sum_{i=1}^n r_i^2 &= \sum_{i=1}^n y_i^2 - \sum_{i=1}^n \widehat{y}_i^2.\end{aligned}$$

The method of least-squares therefore models the training responses \mathbf{y}_n using the linear combination of the columns of \mathbf{X} i.e. $\widehat{\mathbf{y}}_n = \mathbf{X} \widehat{\boldsymbol{\beta}}$, closest to \mathbf{y}_n by finding the coefficient vector $\widehat{\boldsymbol{\beta}}_{p+1}$ that minimises the sum of squared errors. This results in the linear model being selected that maximises the squared correlation between the model predictions and the training responses.

The *residual sum of squares* (RSS) is the sum of the squared model residuals

of the training observations

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^n [y_i - \hat{y}_i]^2 = \sum_{i=1}^n [y_i - \hat{f}(\mathbf{x}_i)]^2 = \sum_{i=1}^n r_i^2 \\ &= \sum_{i=1}^n \left[y_i - \hat{\boldsymbol{\beta}}^T \mathbf{x}_i \right]^2 = \sum_{i=1}^n \left[y_i - \hat{\beta}_0 - \sum_{j=1}^p \hat{\beta}_j x_{ij} \right]^2, \end{aligned} \quad (2.4.5)$$

where y_i is the response of the i^{th} training observation, and \hat{y}_i is the value for the i^{th} response predicted by the model. Using squared-error loss as the loss function (see Section 2.4.9 on page 56), the MLR model is fit to the training data by minimising the RSS. Provided that $\mathbf{X}^T \mathbf{X}$ is invertible i.e. that \mathbf{X} is of full rank, the coefficient vector for the MLR model $\hat{\boldsymbol{\beta}}_{p+1}^{\text{LS}}$ is found by

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{p+1}^{\text{LS}} &= \underset{\boldsymbol{\beta}_{p+1}}{\text{argmin}} \{ \text{RSS} \} \\ &= \underset{\boldsymbol{\beta}_{p+1}}{\text{argmin}} \left\{ \sum_{i=1}^n \left[y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right]^2 \right\} \\ &= \{ [\mathbf{X}_{n \times (p+1)}]^T \mathbf{X}_{n \times (p+1)} \}^{-1} [\mathbf{X}_{n \times (p+1)}]^T \mathbf{y}_n, \\ &= \{ \mathbf{X}^T \mathbf{X} \}^{-1} \mathbf{X}^T \mathbf{y}_n, \end{aligned} \quad (2.4.6)$$

which provides unique, unbiased estimators of the unknown coefficients in (2.4.3) on page 21. The prediction for an observation $\mathbf{x}_{p+1} = (1, \mathbf{x}_p^T)^T$ is then

$$\hat{y} = (\hat{\boldsymbol{\beta}}_{p+1}^{\text{LS}})^T \mathbf{x}_{p+1}$$

and

$$\hat{\mathbf{y}}_n = \mathbf{X} \hat{\boldsymbol{\beta}} = \mathbf{X} \{ \mathbf{X}^T \mathbf{X} \}^{-1} \mathbf{X}^T \mathbf{y}_n$$

is the *orthogonal projection* of \mathbf{y}_n onto the column space of \mathbf{X} . Since $\mathbf{X} \{ \mathbf{X}^T \mathbf{X} \}^{-1} \mathbf{X}^T$ projects \mathbf{y}_n into the column space of \mathbf{X} , converting \mathbf{y}_n to $\hat{\mathbf{y}}_n$, it is known as the *hat matrix* or *projection matrix*.

Let $\mathbf{x}_0 = \mathbf{1}_n$ and $\mathbf{x}_1, \dots, \mathbf{x}_p$ be the p input vectors in the training data set, each of length n , such that $\mathbf{X}_{n \times (p+1)} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_p]$. If $\mathbf{x}_1, \dots, \mathbf{x}_p$ are *orthogonal* to one another i.e. if $\mathbf{x}_i^T \mathbf{x}_j = 0 \forall i \neq j$, then

$$\hat{\beta}_j = \frac{\mathbf{x}_j^T \mathbf{y}}{\mathbf{x}_j^T \mathbf{x}_j}, \quad j \in \{1, \dots, p\}$$

i.e. each coefficient is only dependent on its own input variable (not on any of the other input variables). However, the input vectors of observational data

are seldom orthogonal. In general, the coefficient $\widehat{\beta}_j$ reflects the influence of the input vector \mathbf{x}_j on \mathbf{y} after the effects of $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{j-1}, \mathbf{x}_{j+1}, \dots, \mathbf{x}_p$ have been taken into account. Hence, if some of the columns $\mathbf{x}_1, \dots, \mathbf{x}_p$ of \mathbf{X} are highly correlated, then the variance of the corresponding model coefficient estimators may be high relative to their sizes, resulting in highly unstable coefficients and therefore a less accurate model.

2.4.2.3 Assessing the fit of a multiple linear regression model to the data

The fit of the resulting model to the training data can be ascertained using the RSE and R^2 statistics. One of the assumptions of the MLR model is that the error term ε in (2.4.3) on page 21 has a $N(0, \sigma_\varepsilon^2)$ distribution. This implies *homoscedasticity* i.e. that the error terms of all of the observations of the response share the same variance. As is the case with the model coefficients $\beta_0, \beta_1, \beta_2, \dots, \beta_p$, σ_ε^2 is unknown, but can be estimated from the training observations using the *residual standard error* (RSE)

$$\widehat{\sigma}_\varepsilon = \sqrt{\widehat{\sigma}_\varepsilon^2} = \text{RSE} = \sqrt{\frac{\text{RSS}}{n - p - 1}}. \quad (2.4.7)$$

The RSS in (2.4.5) on the previous page (the numerator in (2.4.7)) measures the cumulative deviation of the training responses from the model predictions, and $n - p - 1$ (the denominator in (2.4.7)) adjusts for the degrees of freedom of the fitted model.

A potential disadvantage of the RSE for assessing the fit of the model to the data is that it measures the spread of observed responses around the MLR model in the units of the response variable. An alternative measure of the fit is the *coefficient of determination* R^2 :

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}, \quad (2.4.8)$$

$$\text{where } \text{TSS} = \sum_{i=1}^n [y_i - \bar{y}]^2 \quad (2.4.9)$$

$$\text{and } \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i.$$

The term RSS/TSS normalises the sample variance of the training observations around the model predictions by the total variance in the response, thereby

providing the proportion of the total variance in the response *not explained* by the model. R^2 therefore measures the proportion of the variance in the response *explained* by the model, providing a measure of the fit of the model to the data that lies between 0 and 1.

In addition to examining model statistics such as the RSE and R^2 , the fit of the model can be assessed graphically by plotting the model residuals against the fitted values. The linearity assumption and the assumption that $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ means that the residuals of a MLR model should have a constant distribution across the input space. However, if an inappropriate model is applied to the data, then in addition to random noise, ε also accommodates deviations between the fitted model and true relationship. Any divergence from Gaussian noise in the model residuals therefore suggests that one or more of these assumptions may be invalid, bringing into question the interpretations derived from the MLR model. Residual plots also help in the identification of *outliers*, which may represent erroneous observations.

2.4.2.4 Importance of input variables in predicting the response under linearity

To determine the usefulness of an input variable x_j in predicting the response Y , the *null hypothesis* $H_0 : \beta_j = 0$ can be tested against the *alternative hypothesis* $H_a : \beta_j \neq 0$ using a *t-test*. This is equivalent to testing whether there is enough evidence to indicate that a relationship exists between x_j and Y , since setting $\beta_j = 0$ in (2.4.3) on page 21 would result in x_j being excluded from the MLR model. The corresponding test statistic

$$t = \frac{\widehat{\beta}_j - 0}{\text{SE}(\widehat{\beta}_j)}, \quad (2.4.10)$$

where $\text{SE}(\widehat{\beta}_j)$ is the standard error of $\widehat{\beta}_j$, measures whether $|\widehat{\beta}_j|$ is far enough away from zero to make it highly unlikely to observe a value as large as $|\widehat{\beta}_j|$ when $\beta_j = 0$. The test statistic has a t_{n-p-1} distribution under the null hypothesis and under the assumption that $\varepsilon \sim N(0, \sigma_\varepsilon^2)$.

2.4.2.5 Usefulness of the multiple linear regression model

The informativeness of several input variables can be assessed simultaneously by determining the impact of excluding the input variables from the model.

Hence, determining the usefulness of the input variables $x_{p-q+1}, x_{p-q+2}, \dots, x_p$, where $q \leq p$, is tested using the null hypothesis

$$H_0 : \beta_{p-q+1} = \beta_{p-q+2} = \dots = \beta_p = 0$$

versus the *alternative hypothesis*

H_a : at least one of the coefficients $\beta_{p-q+1}, \beta_{p-q+2}, \dots, \beta_p$ is nonzero.

This can be assessed by the *F-test*. The resulting *F-statistic* is

$$F = \frac{(\text{RSS}_0 - \text{RSS}_1)/q}{\text{RSS}_1/(n - p - 1)} = \frac{(\text{RSS}_0 - \text{RSS})/q}{(\text{RSE})^2} \quad (2.4.11)$$

where RSS_1 is the training RSS of the model containing all p input variables, RSS_0 is the training RSS of the nested, *reduced model* i.e. the model only containing $p - q$ of the input variables, q is the number of input variable coefficients constrained to be zero, and n and p are the number of training observations and the number of input variables, respectively. The *F*-statistic therefore measures the difference between the RSS of the full and nested models normalised by the difference in the model sizes (numerator in (2.4.11)), which is in turn normalised by $\hat{\sigma}_\varepsilon^2$ given in (2.4.7) on page 25 (denominator in (2.4.11)). If the normalised difference of the training RSS values between the full and reduced models (numerator in (2.4.11)) is much higher than $\hat{\sigma}_\varepsilon^2$ (denominator in (2.4.11)), then F will be much larger than 1, indicating that the q excluded input variables do indeed contribute valuable information towards modelling the response. The *F*-statistic has a $F_{q, n-p-1}$ distribution under the null hypothesis and under the assumption that $\varepsilon \sim N(0, \sigma_\varepsilon^2)$.

It follows that the contribution of a single input variable $x_j, j \in \{1, 2, \dots, p\}$, to the model can be estimated by setting $q = 1$ in (2.4.11) and calculating

$$F = \frac{\text{RSS}_0 - \text{RSS}_1}{\text{RSS}_1/(n - p - 1)}, \quad (2.4.12)$$

where RSS_0 is the training RSS of the model in which $\hat{\beta}_j = 0$. Since the full and reduced models only differ by one input variable, the *F*-statistic in (2.4.12) has an $F_{1, n-p-1}$ distribution. It can therefore also be tested using the *t*-test given in (2.4.10) on the preceding page, which has a t_{n-p-1} distribution under the null hypothesis.

Similarly, in order to test the usefulness of the input variables collectively for modelling the response, $H_0 : \beta_1 = \beta_2 = \dots = \beta_p = 0$ can be tested using

$$F = \frac{(\text{RSS}_0 - \text{RSS}_1)/p}{\text{RSS}_1/(n - p - 1)} = \frac{(\text{TSS} - \text{RSS})/p}{(\text{RSE})^2}, \quad (2.4.13)$$

where RSE is given in (2.4.7) on page 25 and TSS in (2.4.9) on page 25.

2.4.2.6 Selecting a model size

The F -test discussed in the previous section provides a means of determining whether the input variables are useful for modelling the response. However, the F -test can only be used to ascertain whether at least one of the input variables included in the model is informative—it does not indicate how many or which are the most useful input variables. Although a significant F -statistic suggests that the model is useful for modelling the response, including too many unnecessary terms in a model can lead to problems such as loss of interpretability and loss of generality to future observations due to dimensionality problems and multicollinearity (see Section 2.4.12 on page 62). Hence, including too few input variables i.e. excluding informative input variables, will result in a less powerful model; including too many input variables i.e. also including uninformative input variables, will also lead to an inferior model. A model that includes all of the useful input variables while excluding the uninformative inputs tends to predict the response of general observations from the greater population most accurately i.e. tends to have a lower *generalisation error* (see Section 2.4.11 on page 59).

Numerous techniques are available for performing *variable selection*, many of which select an optimum model based on estimates of the generalisation error of the candidate models. Statistics that estimate the generalisation error by adjusting the training error RSS in (2.4.5) on page 24 for the number of parameters include, amongst others, the adjusted R^2 , Mallows' C_p statistic and the Bayesian information criterion (BIC).

The coefficient of determination R^2 in (2.4.8) on page 25 measures the variance of the response explained by the model. However, R^2 is not very useful for variable selection: Even if an input variable is entirely unrelated to the response, it is likely to be weakly correlated with the response by chance. The repercussion of this is that R^2 always increases with the addition of more model terms, irrespective of their usefulness. Similarly, the RSS tends to decrease

with the addition of input variables. Hence, statistics that measure the training error such as R^2 and RSS are poor estimators of the generalisation error, and are therefore unsuitable for variable selection. However, these statistics can be adjusted to make them less biased towards the most complex model.

Let $d \leq p$ denote the number of input variables included in a fitted MLR model. Adjusted R^2 (adj. R^2) is then given by

$$\text{adj. } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}.$$

The quantity that changes with model size in adjusted R^2 is $\text{RSS}/(n-d-1)$. If a useful input variable is added, the RSS is likely to decrease substantially relative to the small decrease in $n-d-1$ due to adding an extra input variable, resulting in a *decrease* in $\text{RSS}/(n-d-1)$ and therefore an *increase* in adjusted R^2 . In contrast, if an uninformative input variable is added, then the RSS will only decrease a little, and the adjusted R^2 of the larger model (the model containing more terms) is likely to be smaller than that of the smaller model. Consequently, based on the adjusted R^2 statistic, the optimum model size is the one that *maximises* adjusted R^2 .

Alternative statistics include the C_p statistic and BIC. The C_p statistic adds an extra $2d\hat{\sigma}_\varepsilon^2$ to the RSS

$$C_p = \frac{1}{n} [\text{RSS} + 2d\hat{\sigma}_\varepsilon^2],$$

a penalty that increases as d grows, thereby compensating for the concomitant decrease in the RSS with the addition of extra variables. The quantity $\hat{\sigma}_\varepsilon^2$ can be estimated using RSE in (2.4.7) on page 25. Adding a useful variable is likely to cause the RSS to decrease more than $2d\hat{\sigma}_\varepsilon^2$ increases, resulting in a smaller C_p . The optimum model size based on the C_p statistic is therefore the size that *minimises* C_p . BIC behaves in a similar manner. For MLR,

$$\text{BIC} = \frac{1}{n} [\text{RSS} + \ln(n)d\hat{\sigma}_\varepsilon^2],$$

and therefore penalises the RSS by adding an additional $\ln(n)d\hat{\sigma}_\varepsilon^2$ to the RSS. Since $\ln(n) > 2 \forall n > 7$, BIC tends to penalise models containing many input variables more heavily than does C_p . Consequently, optimising model size using BIC generally results in smaller models. Unlike adjusted R^2 and the C_p statistic, BIC is a consistent estimator in that it selects the correct model

with increasing probability as $n \rightarrow \infty$ i.e. as n approaches the population size (Efron and Tibshirani, 1993).

The adjusted R^2 , Mallows' C_p statistic and BIC therefore all penalise the training error RSS in order to compensate for the bias resulting from overfitting to the training data. A disadvantage of these statistics is that they all require a knowledge of the number of model parameters. While this is straightforward in the case of MLR—the number of model parameters is equivalent to the number of model coefficients plus 1 (for the intercept)—this is far more challenging in the case of most other modelling techniques such as the lasso and tree-based techniques. In addition, the model needs to be accurate in order to obtain an accurate value for $\hat{\sigma}_\varepsilon^2$. Moreover, the estimator of the noise variance $\hat{\sigma}_\varepsilon^2$ becomes increasingly inaccurate for large p . There are, however, alternative strategies for determining the optimum model size, one of which is cross-validation (see Section 2.4.10 on page 57).

Variable selection is discussed further in Section 2.4.3 on the next page and in Section 2.4.12 on page 62.

2.4.2.7 Quantitative versus qualitative input variables

Quantitative or *numerical variables* are variables that are measured on the real number line. Quantitative variables can therefore assume a large or infinite number of different values that can be ordered with respect to one another. Moreover, the difference between two values can be measured exactly, with smaller distances implying greater similarity between two observations. Examples include temperature readings and the population size of different countries. *Ordered variables*, which include arbitrarily delineated categories such as “high”, “medium” and “low” and “hot” and “cold”, are similar to quantitative variables in that their *levels* can be ordered. However, they differ in that there is no obvious distance measure for their levels. *Qualitative* or *categorical variables*, which can only take on (a small number of) fixed values, fall at the other end of the scale. There is no obvious way to order their levels nor to measure distances between their levels. Examples include the languages spoken in a particular geographical region and the colour of houses in a neighbourhood.

Quantitative input variables have a number of advantages over qualitative input variables. Due to the fact that the relationships amongst the possible values are known, relatively little information is needed in order to incorporate

a quantitative input variable into a statistical model. Knowledge about the relationships amongst the levels also makes interpolation and extrapolation to unobserved values possible. In contrast, the relatively unstructured nature of categorical input variables usually means that more information (more observations) is needed in order to incorporate them into a statistical model with the same level of accuracy, and extrapolation and interpolation to unobserved levels are impossible. In MLR models in particular, which assume linearity between the input variables and the response, only a single coefficient needs to be estimated for each quantitative input variable. This slope coefficient provides an estimate of the change in the response as a result of a one-unit increase in the input variable when all of the other input variables are kept constant. On the other hand, ordered and categorical input variables are incorporated into the model structure as separate *dummy variables*, one for each level of the categorical variable but one. Categorical input variables therefore tend to inflate the dimensionality of the input space, which can lead to problems (see Section 2.4.12 on page 62). It is therefore advisable to keep the number of categorical input variables to a minimum in an analysis.

2.4.3 The lasso

The lasso, introduced by Tibshirani (1996), is a modelling technique that shrinks the MLR coefficients described in Section 2.4.2.2 on page 22 towards zero in an attempt to reduce the variance of the coefficient estimators, thereby yielding a model with greater predictive power. The well-known Gauss-Markov Theorem asserts that the least-squares estimator in (2.4.6) on page 24 of the coefficients of the linear model in (2.4.3) on page 21 has the lowest variance of all *unbiased* estimators of β_{p+1} . However, if the scope of solutions is broadened to include some biased estimators, then an alternative estimator for β_{p+1} may be found by trading a small increase in bias for a large decrease in variance. This is the rationale behind shrinkage techniques such as the lasso, which shrink the least-squares estimators towards zero to reduce their variance. Since no model is entirely accurate—all of the factors affecting the response are seldom known, and the functions fit by statistical modelling techniques are at best gross simplifications of the true relationship between the response and the input variables—a small increase in bias is not necessarily bad. In fact, the main aim when fitting a statistical model to a data set is to select the model

that optimises the balance between the bias and variance of \hat{f} to minimise the generalisation error (see Section 2.4.11 on page 59).

The lasso coefficients $\hat{\beta}_{p+1}^{\text{lasso}}$ are obtained by minimising the ℓ_1 *penalisation criterion*

$$\hat{\beta}_{p+1}^{\text{lasso}} = \underset{\beta_{p+1}}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^n \left[y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right]^2 + \lambda \sum_{j=1}^p |\beta_j| \right\} \quad (2.4.14)$$

$$\begin{aligned} &= \underset{\beta_{p+1}}{\operatorname{argmin}} \left\{ \frac{1}{2} \text{RSS}_{\text{train}} + \lambda \sum_{j=1}^p |\beta_j| \right\} \\ &= \underset{\beta_{p+1}}{\operatorname{argmin}} \{ \text{RSS}_{\text{train}} \} \quad \text{subject to} \quad \sum_{j=1}^p |\beta_j| \leq t, \end{aligned} \quad (2.4.15)$$

where $t, \lambda \geq 0$. The lasso criterion in (2.4.14) consists of two terms—the training error $\text{RSS}_{\text{train}}$ (see (2.4.5) on page 24) and the ℓ_1 *penalty term*—with their relative influences on the solution $\hat{\beta}_{p+1}^{\text{lasso}}$ controlled by the *tuning parameter* λ . $\text{RSS}_{\text{train}}$ measures the fit of the model to the training data and is minimised by $\hat{\beta}_{p+1}^{\text{LS}}$. In contrast, the penalty term measures the ℓ_1 norm of the coefficient vector, $\|\beta_p\|_1 = \sum_{j=1}^p |\beta_j|$, and is minimised by $\hat{\beta}_p = \mathbf{0}_p$. Consequently, the solution will lie somewhere between these two extreme values. The restriction on the sizes of the input variable coefficients is more explicit in (2.4.15). Since the intercept $\hat{\beta}_0$ does not reflect the relative usefulness of any of the input variables, it is not shrunk and is therefore not included in the penalty term. $\hat{\beta}_{p+1}^{\text{lasso}}$ is sensitive to the scale of the input variables, and the input variables must be standardised to have a mean of zero and a variance of one prior to the modelling analysis. There is no closed form solution to $\hat{\beta}_{p+1}^{\text{lasso}}$ (as opposed to least-squares, which has the solution given in (2.4.6) on page 24).

The *tuning parameter* λ regulates the relative effects of the $\text{RSS}_{\text{train}}$ and the penalty term on the solution $\hat{\beta}_{p+1}^{\text{lasso}}$. As $\lambda \rightarrow 0$ (as $t \rightarrow \infty$), the penalty term will have progressively less influence, resulting in successive linear models that fit the training data more closely. At the extreme, $\lambda = 0$ will yield $\hat{\beta}_{p+1}^{\text{lasso}} = \hat{\beta}_{p+1}^{\text{LS}}$. On the other hand, as $\lambda \rightarrow \infty$ (as $t \rightarrow 0$), the penalty term will increasingly dominate, leading to linear models with more coefficients close to or equal to zero. At the extreme, $t = 0$ will yield $\hat{\beta}_{p+1}^{\text{lasso}} = (\hat{\beta}_0, \mathbf{0}_p^T)^T$ i.e. the *null model*. If λ is sufficiently large, the algorithm will shrink some of the coefficients to zero, and the lasso model is therefore not linear in the input variables (although it is linear in the input variables with nonzero coefficients).

The penalty term therefore results in $\|\widehat{\boldsymbol{\beta}}_{p+1}^{\text{lasso}}\| \leq \|\widehat{\boldsymbol{\beta}}_{p+1}^{\text{LS}}\|$, with equality only if $\lambda = 0$. Thus the lasso is a penalised version of MLR, with the coefficient estimators being biased towards zero (unless the restrictions are relaxed enough to include $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{LS}}$ within the set of permissible solutions). In the p -D input space spanned by the p input variables, the ℓ_1 penalty term results in the allowable values of $\boldsymbol{\beta}$ together forming a hypercube (because the variables have been standardised) centred at $\mathbf{0}_p$, with all of its corners positioned on the coefficients' axes. The solution to $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{lasso}}$ is the $\boldsymbol{\beta}$ vector in this region of permissible coefficient values that is positioned closest to the $\boldsymbol{\beta}$ that minimises the $\text{RSS}_{\text{train}}$ i.e. positioned closest to $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{LS}}$, which usually lies outside of the hypercube. Since it is highly likely that one of the hypercube's edges will be the part of the permissible region positioned closest to $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{LS}}$, some of the coefficients in $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{lasso}}$ will be exactly equal to zero, effectively excluding their corresponding input variables from the resulting lasso model. Hence, in addition to coefficient shrinkage, the lasso also performs variable selection. This has the advantage of making the model more interpretable.

Since the minimisation criterion in (2.4.15) on the previous page penalises the absolute values of the coefficients, the ℓ_1 penalty of the lasso shrinks all of the $\widehat{\boldsymbol{\beta}}_{p+1}^{\text{LS}}$ coefficients to similar extents, irrespective of their relative sizes (as opposed to the ℓ_2 penalty $\sum_{j=1}^p \beta_j^2$ in ridge regression, which penalises larger coefficients more severely than it does smaller coefficients). Consequently, the MLR coefficients of the standardised input variables that are close to zero are likely to be set equal to zero by the lasso algorithm, while those standardised MLR coefficients further away from zero tend to require larger values of λ before they are also set equal to zero, all else being equal. However, the shrinkage of the coefficients of useful input variables is countered by the $\text{RSS}_{\text{train}}$, which ensures that, amongst the candidate coefficient vectors available under the restrictions imposed by the penalty term, the coefficient vector that maximises the fit of the model to the training data is chosen. Consequently, the $\text{RSS}_{\text{train}}$ and penalty terms together result in uninformative input variables being penalised more heavily than informative input variables.

The lasso therefore selects shrunken coefficients that result in a model that fits closely to the training data, depending on the value of λ . Since the tuning parameter regulates the fit of the model to the training data, its value cannot be estimated directly from the training data set, and must be optimised using

a method such as cross-validation (see Section 2.4.10 on page 57).

The lasso can be fit using the `glmnet` package (Friedman *et al.*, 2010) in R.

The techniques discussed so far—MLR and the lasso—make highly restrictive and often unrealistic assumptions regarding the relationship between the input variables and the response. They tend to yield stable models (low variance) that are easy to interpret, and can outperform more sophisticated models in low signal-to-noise settings or when data are limited. However, their rigid structures usually do not enable them to model the true relationship accurately (high bias), resulting in poor predictions. The remaining modelling techniques discussed in this chapter allow nonlinear relationships between the input variables and the response in their structures, often resulting in models that have greater flexibility to capture more complex relationships.

2.4.4 Regression trees

In addition to Hastie *et al.* (2009) and James *et al.* (2013), Therneau and Atkinson (2015) was also consulted in writing this section.

2.4.4.1 Introduction

Regression trees are one of the nonlinear techniques available for supervised learning. A popular method for constructing decision trees is by CART (classification and regression trees; Breiman *et al.*, 1984), which implements *recursive binary splitting* to partition the input space. The algorithm divides the input space up into J mutually exclusive regions R_1, R_2, \dots, R_J based on the relationships between the input variables and response, and assigns the same prediction to all unseen cases that fall into a particular region. The resulting splitting rules are easily visualised by means of a decision tree (Figure 2.1 on the next page). If the splits only involve the variables themselves i.e. x_1, x_2, \dots, x_p (as opposed to transformations of the variables, e.g. x_1^2 , $\ln(x_2)$ and $\sqrt{x_3}$, or combinations of the variables, e.g. x_1x_2), then the resulting regions will be rectangular. The variables towards the top of the tree are the most important input variables i.e. the input variables most informative for predicting the responses in the training set.

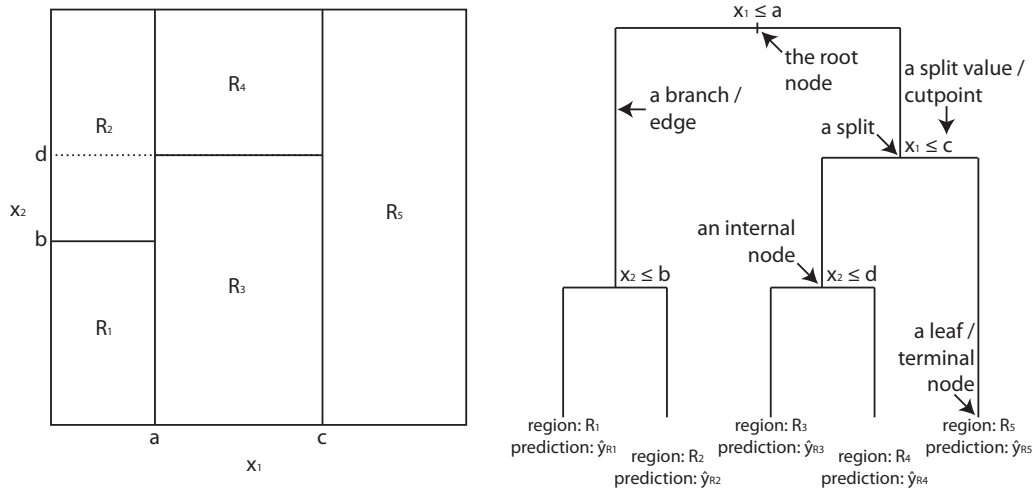


Figure 2.1: A partition by recursive binary splitting of the 2-D input space into five regions (left), and the same partitioning represented in the form of a regression tree (right), with some terminology indicated.

Associated with each leaf is the prediction for that region. Using squared-error loss (see Section 2.4.9 on page 56) for measuring prediction error

$$\hat{y}_{R_j} = \underset{c}{\operatorname{argmin}} \left\{ \sum_{i: \mathbf{x}_i \in R_j} [y_i - c]^2 \right\} = \frac{1}{n_j} \sum_{i: \mathbf{x}_i \in R_j} y_i, \quad (2.4.16)$$

where n_j is the number of training observations in the j^{th} region and y_i is the i^{th} response value in the training data set. In other words, the prediction for each terminal node of the regression tree that maximises the fit of the model to the training data under squared-error loss is the mean of the training responses in that region. Thus each region has a single prediction—the conditional mean of the training responses—that is assigned to all cases that fall into that region, resulting in the *piecewise-constant model*

$$f(\mathbf{x}_p) = \sum_{j=1}^J c_j \cdot I(\mathbf{x}_p \in R_j), \quad (2.4.17)$$

where $c_j = \hat{y}_{R_j}$. The response of an unseen case is predicted by moving down the tree, at each split following the branch corresponding to the input variable values of the unseen case. Once a particular terminal node is reached, the prediction is the mean response of the training cases in that region.

2.4.4.2 Partitioning the input space

Especially in the case of many variables, it is impractical to consider every possible split of every variable. A more efficient strategy is to construct the tree by segmenting the feature space into successively smaller, non-overlapping regions. This results in the stepwise, top-down, greedy approach known as *recursive binary splitting*.

The algorithm starts with the whole input space, successively dividing the input space into more and more regions. Using squared-error loss for assessing the fit of the model, the algorithm aims to find the split that results in the largest reduction in the training error at each step:

$$\text{RSS}_{\text{train}} = \sum_{j=1}^J \sum_{i:\mathbf{x}_i \in R_j} [y_i - \hat{y}_{R_j}]^2, \quad (2.4.18)$$

where J denotes the number of regions. This involves finding the input variable x_j and *split point* or *cutpoint* s for each split that results in the smallest RSS values within the two daughter nodes $\text{RSS}_{R_1(j,s)}$ and $\text{RSS}_{R_2(j,s)}$. The *split criterion* therefore involves choosing the input variable and split point that maximises

$$\text{RSS}_R - [\text{RSS}_{R_1(j,s)} + \text{RSS}_{R_2(j,s)}] \quad (2.4.19)$$

at each split, where

$$\begin{aligned} R &= R_1(j, s) \cup R_2(j, s) & \text{RSS}_R &= \sum_{i:\mathbf{x}_i \in R} [y_i - \hat{y}_i]^2 \\ R_1(j, s) &= \{\mathbf{x}_p \in R \mid x_j \leq s\} & \text{RSS}_{R_1(j,s)} &= \sum_{i:\mathbf{x}_i \in R_1(j,s)} [y_i - \hat{y}_{R_1(j,s)}]^2 \\ R_2(j, s) &= \{\mathbf{x}_p \in R \mid x_j > s\} & \text{RSS}_{R_2(j,s)} &= \sum_{i:\mathbf{x}_i \in R_2(j,s)} [y_i - \hat{y}_{R_2(j,s)}]^2. \end{aligned}$$

Finding the split that results in the largest reduction in the $\text{RSS}_{\text{train}}$ is tantamount to segmenting the input space in such a way as to make the training responses in each region as similar as possible to one another, thereby minimising the variance in each region. The process continues until some *stopping criterion* is reached, e.g. stopping when there are fewer than a prespecified number of training observations in each region.

Consequently, the regression tree algorithm is a *greedy* approach, because each split is chosen based on what would be the best split based on the current

tree, rather than choosing a split that might result in a better tree further on in the process.

2.4.4.3 Pruning regression trees

Regression trees constitute a nonlinear technique, and are therefore more flexible than linear techniques such as MLR. Regression trees can be made even more flexible by adding more splits to the tree. However, an increase in flexibility results in an increase in variance due to overfitting to the training data. This is because, the more regions that the input space is partitioned into, the fewer the training observations in each region. Since the prediction of any unseen case is the conditional mean of the training responses, the resulting predictions will be more heavily influenced by the noise in the training data set, making them less accurate. On the other hand, reducing the flexibility by decreasing the number of splits leads to an increase in bias due to there being fewer regions for predicting observation responses. Regulating the tree size therefore regulates the bias-variance trade-off, and choosing the optimal tree size forms an important part of the tree construction process. Cross-validation (CV; see Section 2.4.10 on page 57) can be used to estimate the optimal tree size.

One way to decrease the variance of a regression tree is to terminate the splitting process early. This can be done by continuing to partition the input space until the improvement in the $\text{RSS}_{\text{train}}$ falls below a prespecified threshold value. However, the (negative) gradient of the $\text{RSS}_{\text{train}}$ does not increase monotonically as the number of nodes in the tree increases. This strategy therefore has the disadvantage that, if earlier splits cause a relatively small reduction in the $\text{RSS}_{\text{train}}$, then some good splits (splits that cause a substantial reduction in the $\text{RSS}_{\text{train}}$) that would have occurred later on might be missed.

A better strategy to reduce variance is by pruning a deep regression tree T_0 (a tree containing many splits) to obtain a *subtree* T , where $T \subseteq T_0$ can be any tree obtained by sequentially pruning terminal branches from T_0 . Ideally, the best subtree would be chosen by producing all possible pruned trees from T_0 and choosing the one that achieves the smallest error rate on unseen cases. However, for deep regression trees this is impractical, and it is necessary to choose a suitable subtree in another way.

In *cost-complexity pruning* or *weakest link pruning* a sequence of nested

subtrees is produced, and the subtree that results in the best balance (measured by CV) between fit to the training data and reduced flexibility is chosen. The sequence of nested subtrees is constructed by starting with the deepest possible tree T_0 and sequentially pruning away the weakest link, which is the node that yields the smallest decrease in the $\text{RSS}_{\text{train}}$ value. This results in a sequence of successively shallower, nested trees, with the last tree in the sequence only containing one split (called a *stump*). The algorithm chooses the best subtree $T \subseteq T_0$ from this nested sequence by minimising the *cost-complexity criterion*:

$$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i:\mathbf{x}_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|, \quad (2.4.20)$$

where $|T|$ denotes the number of terminal nodes in T , and α is a small, non-negative tuning parameter. Minimising only the first term (the *bias term*) in (2.4.20), which corresponds to the case where $\alpha = 0$, would maximise the fit of the tree to the training data, resulting in T_0 being chosen as the best subtree. However, the second term (the *complexity/variance term*) in (2.4.20) penalises the deeper subtrees in the sequence more severely, and increasing α would therefore lead to progressively shallower subtrees being chosen. Equation (2.4.20) therefore minimises the $\text{RSS}_{\text{train}}$ over all regions while penalising flexibility (measured as the number of terminal nodes). The severity of the penalisation is controlled by α , with larger values of α penalising complexity to a greater extent, and $|T|$ therefore decreases monotonically as α increases. The optimum value of α can be determined by CV, and, once known, the best subtree from the nested sequence of trees can be chosen.

2.4.4.4 Algorithm for building a regression tree

A regression tree can therefore be constructed as follows:

1. Build a deep tree T_0 on the training data using recursive binary splitting, stopping when there are fewer than a prespecified number of training observations in each region.
2. Prune T_0 by removing the terminal node at each step that is responsible for the smallest decrease of the $\text{RSS}_{\text{train}}$. In this way a nested sequence

of best subtrees is obtained, each subtree corresponding to a different value of α .

3. Perform K -fold CV to determine the optimum value of α .
4. Choose the tree in step 2 that corresponds to the α selected in step 3.

2.4.4.5 Building a regression tree in R

The `rpart` package (Therneau *et al.*, 2014) provides functions for building classification and regression trees in R. For regression problems, the `anova` method is implemented, which uses the splitting criterion given in (2.4.19) on page 36 to decide on the best input variable and split point for each node.

Some of the parameters that can be specified in a call to the `rpart` function include:

minsplit The minimum number of training observations in a node for the algorithm to attempt to split the node (default: `minsplit = 20`).

minbucket The minimum number of observations allowed in a terminal node (default: `minbucket = round(minsplit/3)`).

xval The number of CV folds (default: `xval = 10`).

maxdepth The maximum allowable depth of any node in the tree (default: `maxdepth = 30`).

cp The minimum by which a proposed split must improve the overall fit of the tree (measured in terms of R^2) to justify the split being added to the model structure (default: `cp = 0.01`).

The complexity parameter `cp` controls the size of the final tree. If the best split of a node does not increase the model's overall R^2 by an amount greater than or equal to the value specified for `cp`, then that split is not made and that branch of the tree is not split any further. Although the default setting of `cp = 0.01` works well in general, lower values have been found to work better for larger data sets (Therneau and Atkinson, 2015).

The `rpart` function uses cost-complexity pruning to prune the full tree, dividing the training set up into the number of folds specified by `xval` for CV. The mean of the prediction errors obtained in CV can be plotted as a function

of the complexity tuning parameters tested using the `plotcp` function. The number of terminal nodes in the pruned tree corresponding to each complexity parameter can be indicated on the top axis of the plot. This plot provides a graphical indication of the optimum complexity parameter for pruning the full regression tree model, which is achieved with the `prune` function.

2.4.4.6 Conclusion

Regression trees have a number of advantages: they are simple, highly interpretable (shallow trees are easy to understand, even for non-experts), naturally perform variable selection, can easily handle both qualitative and quantitative input variables, and have low bias if grown deep enough. However, a disadvantage of modelling the response with a regression tree is that the resulting response surface is piecewise constant rather than smooth. This will lead to prediction estimators with high variance if the tree is deep—the most appropriate prediction for an observation located towards the edge of a region in a partition of the input space is dubious—and high bias if the tree is shallow—all of the observations in a region receive the same prediction. Furthermore, the hierarchical nature of the tree structure means that inaccurate splits made in the upper nodes of a branch affect the rest of the nodes in that branch. Moreover, the regression tree algorithm only splits on one input variable at a time in a region, and can therefore only model boundaries perpendicular to the input axes in the input space (as opposed to, e.g. MLR, which can model linear combinations of input variables). Regression trees therefore tend to be unstable, especially if the signal is blurred by phenomena such as multicollinearity, and can also have high bias if they contain only a few splits, resulting in relatively poor prediction accuracy.

To mitigate these drawbacks, a number of *ensemble methods* have been proposed that involve combining the results from a large number of regression trees to produce a single aggregate prediction for each input variable value. These methods include bagging, random forests and boosting. Although these more modern techniques typically lead to a loss in interpretability, they tend to increase the prediction accuracy substantially, making them competitive with other nonlinear supervised techniques. These three ensemble methods are discussed in the next three sections.

2.4.5 Bagging

Bagged regression tree models are composed of decision trees that are constructed from data sets resulting from the random sampling of observations from the training set. Bagging makes use of the nonparametric bootstrap approach, which it uses to generate prediction estimators with relatively little bias and variance, thereby producing more stable models. The bootstrap technique is described next.

Material regarding the bootstrap approach was obtained from Efron and Tibshirani (1993).

2.4.5.1 The bootstrap method

To determine the *parameter* of interest, e.g. the expected value, of a random vector \mathbf{X}_p in a population, the variable should ideally be measured for the whole population i.e. a *census* should be taken. However, this is often infeasible since populations are usually both spatially and temporally large or infinite in size. *Statistical inference* often involves estimating a parameter from the distribution of \mathbf{X}_p using an *independent and identically distributed* (i.i.d.) *sample* (also known as a *random sample*), which consists of n observations each sampled with equal probability from the distribution of \mathbf{X}_p . A *point estimate* for the parameter based on a suitable *statistic*, e.g. the sample average or the sample median if the parameter of interest is the expected value, can be calculated from a random sample. To get an idea of the accuracy of the point estimate, the same statistic could be calculated from numerous independent samples from the population, resulting in an estimate of the *sampling distribution* of n observations for the statistic. The spread of the estimates for the parameter from the independent samples can be used to estimate the standard error of the statistic (and confidence intervals) as well as other measures of accuracy such as bias.

However, data are often limited, and only one sample from the population is usually available. Extensive research has been undertaken for the statistics of parameters commonly of interest, amongst others the sample mean, median, variance, maximum, minimum and correlation coefficient, for many univariate and some multivariate distributions. Consequently, if the underlying distribution $F_{\mathbf{X}}$ that generated the observations is known, then for these

parameters a point estimate can be calculated from a sample of n observations using the appropriate statistic, and the standard error can be calculated from the same sample based on the known distribution of the statistic. However, $F_{\mathbf{X}}$ is usually unknown, and a distribution must be assumed that may or may not be appropriate for the data. For a few statistics such as the mean, median, minimum and maximum, asymptotic distributions are known for distributions that satisfy certain regularity conditions, e.g. finite variance, and the accuracy of estimates can therefore still be approximated in the absence of complete knowledge of $F_{\mathbf{X}}$. However, assumptions made by asymptotic methods are often inappropriate in the case of small samples. A further restriction on using known distributions (asymptotic or otherwise) to quantify uncertainty is that the distributions can be complicated to calculate, as is the case for the median. Hence, the distributions of most statistics (except for the mean for large samples) cannot be determined analytically without making assumptions regarding the population distribution.

The *bootstrap* (BS), introduced by Efron (1979), is a computer-based method that provides a means of estimating the sampling distribution of a statistic from a random sample of n observations without the need to know either the distribution of \mathbf{X}_p or the (asymptotic) distribution of the statistic. Let $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$ be an i.i.d. sample of size n obtained from the distribution of $\mathbf{X}_p, F_{\mathbf{X}}$. The *empirical distribution* of $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n, F_n$, is a discrete step function whose probability mass function assigns a probability of $1/n$ to each of $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$, and a probability of 0 to all other observations of \mathbf{X}_p . The *Law of Large Numbers* (LLN) implies *pointwise convergence* in that, given an i.i.d. sample $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$ from a population with finite variance, F_n converges to $F_{\mathbf{X}}$ as n approaches the size of the population i.e. $F_n \rightarrow F_{\mathbf{X}}$ as $n \rightarrow \infty$, for each \mathbf{x}_p in the support of $F_{\mathbf{X}}$ (Ross, 2013). F_n is therefore a *consistent estimator* for $F_{\mathbf{X}}$. Furthermore, the *Glivenko-Cantelli Theorem* guarantees *uniform convergence* i.e. that the rate of convergence $F_n \rightarrow F_{\mathbf{X}}$ as $n \rightarrow \infty$ is identical for all areas of the support of $F_{\mathbf{X}}$ represented by the same number of sample observations (Ross, 2013).

The BS approach therefore involves using the distribution of \mathbf{X}_p in the original sample, F_n , to estimate the distribution of \mathbf{X}_p in the population, $F_{\mathbf{X}}$. This approximation tends to be more accurate for larger n . A point estimate is calculated from the original sample of size n , and the uncertainty of the

point estimate is quantified using the BS. In the case of a sample containing n independent observations of \mathbf{X}_p , this is achieved in the *nonparametric bootstrap* by randomly sampling n observations from the original sample B times to form B independent *bootstrap samples*, each of size n . Randomly sampling n observations from a data set of size n involves selecting each observation with a probability of $1/n$. This will result in some observations being absent and others being present more than once in a particular BS sample, and sampling observations with equal probability is therefore called *sampling with replacement*. The statistic used for the point estimator is calculated for each of the BS samples, resulting in B *bootstrap replicates* that together form a *bootstrap distribution* for the statistic. Since the original sample is used to approximate the population, and the BS samples are sampled from the original sample in a way analogous to taking numerous i.i.d. samples from the population, the bootstrap distribution emulates the sampling distribution of the statistic in the original population. Hence, a standard error for the point estimate can be obtained by finding the standard deviation of the BS distribution, which is an approximation of the standard error of the sampling distribution. Similarly, confidence intervals and other accuracy measures can also be found.

If all possible n^n unique BS permutations are taken from the original sample of size n , then the BS analysis results in the *ideal bootstrap estimate* of the standard error of the estimator for the parameter, which is the expectation of the standard error of the statistic over F_n . However, n is usually quite large, making n^n prohibitively large in terms of computer time—more so for complicated statistics. Consequently, the standard error for the vast majority of statistics is usually estimated from $B \ll n^n$ BS samples drawn randomly i.e. with replacement, from the n^n possible BS permutations. This results in a BS estimate of the standard error that is an estimate of the corresponding ideal BS estimate. Consequently, the BS estimate of the standard error of a statistic for a parameter is an approximation of the ideal BS estimate, which in turn is an approximation of an estimate that could be obtained from a sampling distribution of the statistic. A point estimate together with a standard error constitute an estimate of the population parameter of interest.

The BS therefore provides a way of assessing the accuracy of (mathematically complicated) statistical estimators of population parameters from a single sample without the need for (intricate) theoretical calculations. That being

said, the BS is not an equally effective method for all parameters. For example, the BS approach estimates the accuracy of smooth statistics such as the mean and variance more effectively than it does statistics that assume discrete values such as the median. Furthermore, the BS is more effective for statistics for estimating parameters towards the centre of the population distribution such as the mean and median than for parameters deep in the tails of the distribution such as the minimum and maximum and other extreme quantiles.

Consequently, although the BS has limitations, it is an effective technique for statistical inference using a limited number of observations in the absence of theoretical knowledge about the distributions of statistics.

2.4.5.2 Bagged regression trees

Bagging or *bootstrap aggregation*, due to Breiman (1996), is an ensemble technique for constructing composite models consisting of numerous individual models of the same type using the bootstrap. Although the BS approach is usually used to quantify the uncertainty of a point estimator from a sample, as described in the previous section, the BS is used for another purpose in bagging: To obtain more stable predictions of the response. More specifically, bagging uses the BS technique to generate a multitude of (highly variable) predictions. The algorithm then exploits the reduction in variance that results from taking the average of identically distributed quantities to calculate more stable aggregate predictions, thereby increasing prediction accuracy. Bagging can be used to reduce variance of models that are nonlinear and/or adaptive functions of the training data. However, bagging is most useful in cases where the individual models have low bias but high variance, as is the case for deep decision trees.

Let X_1, X_2, \dots, X_B denote B identically distributed (i.d.) random variables with expected value μ_X , variance σ_X^2 and pairwise positive correlations ρ_X . The expected value of the average is

$$\mathbb{E}_X \left[\frac{1}{B} \sum_{b=1}^B X_b \right] = \frac{1}{B} \sum_{b=1}^B \mathbb{E}_X(X_b) = \frac{1}{B} \sum_{b=1}^B \mu_X = \frac{B\mu_X}{B} = \mu_X, \quad (2.4.21)$$

and the variance of the average is

$$\begin{aligned}
\text{Var}_X \left[\frac{1}{B} \sum_{b=1}^B X_b \right] &= \sum_{b=1}^B \left(\frac{1}{B} \right)^2 \text{Var}_X(X_b) + 2 \sum_{1 \leq i < j \leq B} \left(\frac{1}{B} \right)^2 \text{Cov}_X(X_i, X_j) \\
&= \frac{1}{B^2} \sum_{b=1}^B \sigma_X^2 + \frac{1}{B^2} \sum_{i \neq j} \rho_X \sigma_X^2 \\
&= \frac{1}{B^2} B \sigma_X^2 + \frac{1}{B^2} (B^2 - B) \rho_X \sigma_X^2 \\
&= \frac{1}{B} \sigma_X^2 + \left(1 - \frac{1}{B} \right) \rho_X \sigma_X^2 \\
&= \rho_X \sigma_X^2 + \frac{1}{B} (\sigma_X^2 - \rho_X \sigma_X^2) \\
&= \rho_X \sigma_X^2 + \frac{1 - \rho_X}{B} \sigma_X^2.
\end{aligned}$$

If B is large, then the second term is negligibly small, and

$$\text{Var}_X \left[\frac{1}{B} \sum_{b=1}^B X_b \right] \approx \rho_X \sigma_X^2 \text{ for } B \text{ large.} \quad (2.4.22)$$

Hence, taking the average of i.d. quantities leaves the expected value μ_X unchanged, but reduces the variance σ_X^2 by a factor $\rho_X \in [0, 1]$. It follows that, the closer to zero the pairwise correlations ρ_X amongst the quantities are, the lower will be the variance of the average compared to that of the individual quantities.

The above results provide the justification for the bagging technique, which proceeds as follows: Let $\mathbf{Z}_{n \times (p+1)} = [\mathbf{z}_1, \dots, \mathbf{z}_n]^T = [(\mathbf{x}_1^T, y_1)^T, \dots, (\mathbf{x}_n^T, y_n)^T]^T$ denote the training data set containing n observations of the input variables and response variable. To generate numerous predictions for each part of the input space from a single training data set \mathbf{Z} , the bagging algorithm generates B independent BS samples $\mathbf{Z}^{*1}, \dots, \mathbf{Z}^{*B}$ of \mathbf{Z} , where $\mathbf{Z}_{n \times (p+1)}^* = [(\mathbf{x}_1^{*T}, y_1^*)^T, \dots, (\mathbf{x}_n^{*T}, y_n^*)^T]^T$ is a *resampled* version of $\mathbf{Z}_{n \times (p+1)}$ consisting of n training observations selected randomly with replacement. $\mathbf{Z}^{*1}, \dots, \mathbf{Z}^{*B}$ are *independent* with respect to the empirical distribution function F_n , but *not independent* with respect to the distribution function $F_{\mathbf{Z}}$ that generated the training observations.

In the context of regression trees, a deep, unpruned regression tree is then fit to each BS sample. The B bootstrap trees are likely to involve different partitions of the input space. The differences amongst the BS samples as

well as the high flexibility of the models fit to them will result in prediction estimators from the individual BS trees having low bias but high variance. From (2.4.21) on page 44, the expected value of the average of i.d. quantities is the same as the expected value of each of the quantities, and the bias of the bagged model will therefore be the same as that of the individual BS trees. However, from (2.4.22) on the preceding page the variance of the average of i.d. quantities tends to be lower than that of the individual quantities. Hence, taking the average of the predictions generated by the individual BS trees for a given input vector is likely to cancel out a large proportion of the random error present in the original training data set, thereby yielding aggregate prediction estimators with far lower variances, while leaving the bias unchanged.

To obtain a prediction for a test case, the prediction from each of the B trees is averaged to obtain the aggregate prediction:

$$\widehat{f}_{\text{bag}}(\mathbf{x}_p) = \frac{1}{B} \sum_{b=1}^B \widehat{f}_{\text{bag}}^{*b}(\mathbf{x}_p), \quad (2.4.23)$$

where $\widehat{f}_{\text{bag}}^{*b}(\mathbf{x}_p)$ is the prediction from the b^{th} regression tree for \mathbf{x}_p . The larger B is, the closer $\widehat{f}_{\text{bag}}(\mathbf{x}_p)$ is likely to be to the true bagging estimate, which is the ideal BS prediction at \mathbf{x}_p , and any large number for B should therefore yield good results. The BS replicates $\widehat{f}_{\text{bag}}^{*1}(\mathbf{x}_p), \dots, \widehat{f}_{\text{bag}}^{*B}(\mathbf{x}_p)$ constitute valid predictions for the response at \mathbf{x}_p , since they are based on the BS samples $\mathbf{Z}^{*1}, \dots, \mathbf{Z}^{*B}$ that were generated from the empirical distribution function F_n of the observations in the training data set, which is a consistent estimator of $F_{\mathbf{Z}}$ (see the previous section). The bagging algorithm therefore emulates the process of taking B i.i.d. samples from $F_{\mathbf{Z}}$ and averaging the predictions resulting from fitting a regression tree to each sample.

Since the height of the fitted response surface \widehat{f}_{bag} is the average of the predictions from many BS trees at each \mathbf{x}_p value (see (2.4.23)), and since each BS tree involves a (slightly) different partition of the input space, the response surface \widehat{f}_{bag} changes more gradually over the input space than does the piecewise-constant fitted response surface of a single regression tree (see (2.4.16) on page 35). A bagged model therefore has a greater potential to approximate a complex process accurately than does a regression tree model.

An added advantage of using the BS technique in bagging is that the test error does not need to be estimated using CV—it is very easy to estimate

using *out-of-bag* (OOB) *error estimation*. It can be shown that generating B BS samples of n observations will result in each observation being present in, on average, two-thirds of the BS samples. Consequently, about one-third of the training observations will be absent from each BS sample, and these are referred to as the *OOB observations* for that sample. Since these OOB observations were not used to construct the regression tree for that sample, these observations can be used as test cases for that regression tree. The test error of the i^{th} observation $\mathbf{z}_i = (\mathbf{x}_i^T, y_i)^T$ is therefore estimated using the $\approx \frac{1}{3}B$ regression trees for which \mathbf{z}_i was OOB. This is a valid way of estimating the test error, since only test cases are used to calculate it.

In summary, bagging achieves a substantial reduction in variance in the case of unstable methods such as regression trees by generating aggregate predictions that are the averages of those from the individual (high variance, low bias) BS trees. The random forest technique, which is discussed in Section 2.4.6, follows a similar strategy, but attempts to bring about an even greater degree of variance reduction by making the trees less similar to one another (reducing ρ_X).

2.4.5.3 Building bagged regression tree models in R

Bagged regression tree models can be fit in R using the package `randomForest` (Liaw and Wiener, 2002). For regression problems, parameters that can be adjusted in a call to the `randomForest` function include:

ntree The number of BS samples and therefore the number of BS regression trees making up the model (default: `ntree = 500`).

mtry The number of the p input variables to be randomly sampled as candidates for each split in each of the BS trees. The default setting for a quantitative response of `mtry = floor(p/3)` is more appropriate for random forests (discussed in the next section), whereas `mtry = p` results in a bagged regression tree model.

nodesize The minimum number of training observations permitted in the terminal nodes of the individual BS trees (default: `nodesize = 5` for regression trees).

maxnodes The maximum number of leaves allowed in each of the BS trees (default: `maxnodes = NULL`, which results in the trees being grown to their maximum depths, subject to constraints such as that specified by `nodesize`).

importance Indicates whether the function should also calculate the relative importance of the p input variables (default: `importance = FALSE`).

2.4.6 Random forests

2.4.6.1 Method description

Although the bagging technique usually achieves a substantial reduction in variance by averaging predictions emanating from a multitude of decision trees based on different BS samples of the training data set, the resulting predictions can still be quite similar to one another. For example, if the training data set contains an input variable that is much more highly correlated with the response than are the other input variables, then most, if not all, of the BS trees will split on this input variable first. This will result in BS trees that are still quite similar to one another, despite the fact that the BS trees are based on different BS samples of the original training data set. Since taking the average of highly correlated quantities does not achieve as great a reduction in variance as does averaging less correlated quantities (see (2.4.22) on page 45), bagging will accomplish a mediocre reduction of variance in this setting. The random forest (RF) method was proposed by Breiman (2001) to bring about an even greater reduction in variance, and therefore prediction error, by making the B BS trees and the resulting predictions even less similar to, and therefore less correlated with, one another than is the case in bagging.

As with bagging, the RF algorithm proceeds by taking B BS samples $\mathbf{Z}^{*1}, \dots, \mathbf{Z}^{*B}$ of the original training data set $\mathbf{Z}_{n \times (p+1)}$, constructing an unpruned regression tree from each BS sample and predicting the response of test cases by taking the average of the prediction from each of the B trees:

$$\hat{f}_{\text{rf}}(\mathbf{x}_p) = \frac{1}{B} \sum_{b=1}^B \hat{f}_{\text{rf}}^{*b}(\mathbf{x}_p)$$

(compare with (2.4.23) on page 46). The difference between the two techniques lies in the number of input variables considered for each split in the

BS regression trees. Whereas in bagging any of the p variables can be used in a split at any level, in RFs a random subset of size $m < p$ of these variables are the only permissible candidates for any particular split. This ensures that the most informative input variable is not always split on, thereby making the BS trees less similar to one another. A new set of random input variables is chosen for each split of each tree. Small values of m can lead to a substantial reduction in the variance of the final prediction estimator, especially if some of the variables are highly correlated with one another, while $m = p$ results in bagging. However, too small a value of m might be too restrictive, increasing the variance $\sigma_{Z^*}^2$ in (2.4.22) on page 45, thereby leading to a worse generalisation error (see Section 2.4.11 on page 59) than would be the case for a larger value of m . The optimum number of variables to be chosen at each level is that value of m that strikes the best balance between the sizes of $\sigma_{Z^*}^2$ and ρ_{Z^*} , and should be determined by a data-driven method such as CV.

As is the case in bagging, the fitted response surface of a RF model involves numerous, small changes across the input space, resulting in a relatively unbiased, flexible fitted model that has the potential to approximate complex surfaces accurately.

2.4.6.2 Building random forest models in R

A RF model can also be constructed using the `randomForest` package in R (see Section 2.4.5.3 on page 47). The settings for a RF model should be the same as those for bagging, except that the `mtry` option should be set to a number less than p . The `mtry` parameter specifies the number of input variables that should be randomly selected as split candidates for each node. Hence, specifying a number less than p results in less similar BS trees, which should help to reduce the variance of the model prediction estimators. The default setting for regression is `mtry = floor(p/3)`.

2.4.7 Boosted regression trees

Apart from Hastie *et al.* (2009) and James *et al.* (2013), information was also sourced from Elith *et al.* (2008) while writing this section.

2.4.7.1 Method description

A boosted regression tree (BRT) model is another ensemble method containing numerous individual regression trees whose predictions are averaged to generate aggregate predictions for observations. However, unlike in bagging and RFs, in which the individual trees are built independently of one another, the BRT algorithm proceeds in a sequential fashion, constructing successive trees based on the previous trees in the model. The BRT algorithm fits each successive tree predominantly to those observations that do not fit well into the current model in an attempt to improve the predictive power of the model. This could easily lead to overfitting if left unchecked, and the algorithm involves a number of complexity parameters to counter this.

In a regression setting with squared-error loss, the algorithm proceeds as follows: Initially, the mean of the training responses $\bar{y} = \sum_{i=1}^n y_i$ i.e. the null model, is fit to the data. The first and subsequent iterations start by calculating the model residuals for the current iteration $m \in \{1, \dots, M\}$

$$r_{im} = y_i - f_{m-1}(\mathbf{x}_i), \quad i \in \{1, \dots, n\},$$

where y_i is the i^{th} training response, $f_{m-1}(\mathbf{x}_i)$ is the prediction for \mathbf{x}_i for the growing BRT model after the previous $m - 1$ iterations, n is the number of training observations and M is the total number of iterations. The algorithm then fits the tree of the pre-specified maximum depth that minimises the residuals (see the following paragraph) of the current model. This tree is then added to the existing model terms, resulting in the model

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \sum_{j=1}^J [\gamma_{jm} \cdot I(\mathbf{x} \in R_{jm})], \quad (2.4.24)$$

(compare with (2.4.17) on page 35) where

$$\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \left\{ \sum_{\mathbf{x}_i \in R_{jm}} L[y_i, f_{m-1}(\mathbf{x}_i) + \gamma] \right\}$$

(compare with (2.4.16) on page 35). Thus, the algorithm fits successive trees to the *residuals* of the current model, enabling it to concentrate at each step (or iteration) on those observations whose responses are the most inaccurately predicted by the growing model. Successive trees (or model terms) therefore

tend to be quite different from one another, since they are each fit to different data. As is the case with bagging and RFs, the final BRT model is a linear combination of the trees making up the model, and predictions of observations are calculated by summing the predictions from the individual trees. BRT is therefore a *forward-stagewise additive procedure*—“forward” because it adds a single term to the model during each iteration, “stagewise” because it adds new trees without changing the trees already in the model, and “additive” because the model terms are summed to generate the final predictions.

The algorithm fits the model that minimises the loss function by adding the tree at each step that results in the maximum reduction in the lack of the model’s fit as measured by the loss function. Let $T_m(\mathbf{x}_i) := T(\mathbf{x}_i \mid \Theta_m)$ represent the prediction for \mathbf{x}_i from the tree fit during the m^{th} iteration with input variables, cutpoints and predictions denoted by Θ_m , where

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \left\{ \sum_{i=1}^n L[y_i, f_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i \mid \Theta_m)] \right\}.$$

Then

$$f_m(\mathbf{x}) = \sum_{l=1}^m T(\mathbf{x} \mid \Theta_l).$$

For squared-error loss,

$$\begin{aligned} L\{y_i, f_m(\mathbf{x}_i)\} &= \{y_i - f_m(\mathbf{x}_i)\}^2 \\ &= \{y_i - [f_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i \mid \Theta_m)]\}^2 \\ &= \{[y_i - f_{m-1}(\mathbf{x}_i)] - T(\mathbf{x}_i \mid \Theta_m)\}^2 \\ &= \{r_{im} - T(\mathbf{x}_i \mid \Theta_m)\}^2. \end{aligned}$$

Hence, for BRT, minimising the loss function is equivalent to maximising the fit of the current iteration’s tree $T(\cdot \mid \Theta_m)$ to the current model residuals $\mathbf{r}_m = (r_{1m}, \dots, r_{nm})^T$. In other words, the tree that best fits the model residuals is the tree that is added to the growing model at each iteration.

Since BRT is an *adaptive* method, adjusting the model based on the model residuals, it is prone to overfitting. The ability of the algorithm to build a good model can be enhanced in several ways. As in bagging and RFs, fitting each tree to a random subset of the training observations tends to improve performance by reducing overfitting (and computational time). Moreover, the test error can be estimated using the held-out observations, as is the case in

bagging and RFs. The proportion of training observations to be randomly sampled without replacement for each iteration is specified by the *bag fraction* $\eta \in (0, 1]$. The *tree complexity parameter* specifies the level of interaction allowed in the model. A tree complexity setting of 1 results in a linear combination of stumps, which does not allow for any interactions amongst the input variables, whereas larger values allow deeper levels of interactions in the trees, resulting in a more complex model. In addition to limiting the interaction level of the model, the tree complexity parameter also influences the rate at which the model can learn from the data: Deeper trees each contain more information about relationships, and fewer model terms are therefore required in order to learn patterns from the data (provided, that is, that the trees are not deep enough to incur high variance; see two paragraphs ahead). The optimum tree depth for an analysis will depend on both the true level of interaction (which is almost always unknown), and the number of independent observations in the data set (larger data sets contain more information about the process being modelled, and more complex models can therefore be fit to the data).

Another important parameter is the *shrinkage parameter* or *learning rate* $\nu \in (0, 1)$. Empirical evidence shows that taking smaller steps down the loss function's gradient tends to yield models with better predictive powers. *Slow learning* can be accomplished by dampening the effect of the model terms on the final predictions by multiplying each term by the pre-specified shrinkage parameter, which is much less than one:

$$\begin{aligned} f_m(\mathbf{x}) &= f_{m-1}(\mathbf{x}) + \nu \cdot \sum_{j=1}^J [\gamma_{jm} \cdot I(\mathbf{x} \in R_{jm})] \\ &= f_{m-1}(\mathbf{x}) + \nu \cdot T_m(\mathbf{x}) \end{aligned}$$

(compare with (2.4.24) on page 50). This biases the prediction from each model term in the linear combination towards zero, thereby requiring more model terms in order to reach a value that will result in convergence i.e. will minimise the error.

A fourth complexity parameter, the number of iterations (or trees) M , specifies the number of terms in the final model. This is an important setting to optimise, since a large number of iterations will eventually result in overfitting. The optimum number of trees depends on the shrinkage parameter and the tree complexity: Learning with stumps will reduce the rate of convergence

considerably, requiring numerous model terms to minimise the loss function. At the other extreme, very deep trees would also require numerous model terms in order to reduce the variance of the final model. Setting the tree complexity very high would therefore require a very slow learning rate, thereby increasing computational time considerably. For this reason, deep trees do not tend to be used in boosting. In general, a small shrinkage parameter (e.g. 0.1) and/or shallow trees (e.g. a tree complexity of 2) will result in far more model terms being added to the model before convergence occurs than will settings specifying a larger shrinkage parameter (e.g. 0.01) and deeper trees (e.g. a tree complexity of 5).

The bag fraction, tree complexity, shrinkage parameter and number of trees are all complexity parameters that affect the fit of the model to the training data, and should therefore be selected by a process such as CV. The bag fraction η is usually set to 0.5 or even less for large data sets. For a given tree complexity, an effective strategy is to specify a small shrinkage parameter ν , and then to determine the number of iterations M from an estimate of the generalisation error, e.g. the OOB error if the bag fraction η is less than 1.

BRT optimises predictive power differently to bagging and RFs. Since the bagging and RF algorithms construct each tree from an independent BS sample of the training data set, the BS samples have similar distributions to one another, and will therefore share similar biases and variances. To minimise bias, deep regression trees are fit to the individual BS replicates of the training data set, while the variance of the final model is reduced by averaging the predictions from the individual trees. BRT also performs model averaging in order to reduce the variance of the final model. Moreover, it fits relatively shallow trees to the data during each iteration. To counter the resulting high bias of the individual models, the BRT algorithm fits successive trees to the residuals of the current model, thereby enabling the model to adapt itself to the patterns in the data.

In summary, different ensemble methods follow different approaches in an attempt to produce a model with good predictive power. The method that does the best will depend on the particular problem at hand and the characteristics of the data that are available for modelling.

2.4.7.2 Constructing boosted regression tree models in R

The `dismo` package (Hijmans *et al.*, 2013) contains functions for constructing boosted regression tree models in R. The `gbm.step` function enables the tuning of the number of trees (or iterations) to include in the final model. It performs 10-fold CV (default setting), identifying the number of trees that results in the smallest average holdout prediction error amongst the folds. It then returns a model fit to the whole training data set containing the optimum number of trees identified.

Parameters of `gbm.step` include:

family The distribution of the error in the response. The default setting of `family = "bernoulli"` is appropriate for classification problems with a binary response, whereas `family = "gaussian"` is appropriate for minimising squared-error loss in regression problems.

n.trees The initial number of trees to be added to the model before the average holdout prediction error of the folds is calculated for the first time (default: `n.trees = 50`).

step.size Number of trees to be added to the model between successive calculations of the average holdout prediction error (default: `step.size = 50`).

max.trees The maximum number of trees to be added to the model (default: `max.trees = 10000`).

tree.complexity The maximum interaction level permitted amongst the input variables. The default setting `tree.complexity = 1` yields additive component models (additive amongst the input variables).

learning.rate The shrinkage of the prediction from each tree when calculating the aggregate prediction for an input (default: `learning.rate = 0.01`).

bag.fraction The fraction of observations randomly sampled, without replacement, from the training data set during each iteration to construct the next tree (default: `bag.fraction = 0.75`).

In addition to `gbm.step`, the `dismo` package also contains:

gbm.simplify Identifies uninformative input variables using K -fold CV and excludes them from the current model.

gbm.plot Draws partial dependence plots between the response and one or more of the input variables.

2.4.8 Importance of input variables in tree-based models

A disadvantage of model-averaging techniques such as bagging, RFs and boosting is that interpretability is lost, since the model no longer consists of a single tree from which the influence of the individual input variables on the response can be seen (as is the case in a regression tree model). However, there are alternative ways of investigating the roles of the different input variables in the model.

For a single tree T with J leaves (and therefore $J - 1$ internal nodes), the importance of x_l can be assessed by averaging the improvement in the $\text{RSS}_{\text{train}}$ in (2.4.18) on page 36 that results from splitting on that variable. The *squared relative importance* of the input variable x_l is

$$\mathcal{I}_l^2(T) = \sum_{j=1}^{J-1} \hat{i}_j^2 I[v(j) = l], \quad (2.4.25)$$

where $I[v(j) = l]$ is the indicator of whether x_l is the input variable split on at the j^{th} internal node and \hat{i}_j^2 is the squared improvement in $\text{RSS}_{\text{train}}$ as a result of the j^{th} split made during training. Thus, \mathcal{I}_l^2 gives the sum of the squared improvements in the training error as a result of splitting on x_l in T . For a linear combination of B trees, the squared relative importance of x_l is simply averaged over the trees:

$$\mathcal{I}_l^2 = \frac{1}{B} \sum_{b=1}^B \mathcal{I}_l^2(T_b). \quad (2.4.26)$$

Averaging results in \mathcal{I}_l^2 provides a far more reliable estimator of the squared relative importance of x_l than does $\mathcal{I}_l^2(T)$. For this reason, variable importance is more routinely estimated for additive tree expansion models such as bagging, RFs and boosting than for single trees. The *relative importance* of x_l is simply the square root of \mathcal{I}_l^2 . However, the relative importances of the input variables are usually scaled so that they, for example, sum to 100.

Another measure of variable importance used in bagging and RF models involves a permutation test. To determine the importance of x_l in a tree T

in the forest, T 's OOB observations are sent down T , and their prediction errors are found. The observed values of x_l are then permuted (or shuffled) amongst these OOB observations, after which they are again sent down T . The difference in their prediction errors before and after the permutation of their x_l values gives an indication of the usefulness of x_l in predicting the response in T . This reduction in prediction accuracy is then averaged over all B trees to provide an indication of the predictive power of x_l in the forest.

Apart from variable importance, the influence of the different input variables on the response in bagging, RFs and boosted regression tree models can be investigated by examining *partial dependence plots*. A partial dependence plot of the predicted response \hat{y} on x_l shows, for each value in the domain of x_l , the average value of \hat{y} for that value of x_l . In other words, it displays \hat{y} after averaging \hat{y} over the values of the other input variables, and therefore shows the average value of the predicted response for each value of the variable of interest.

2.4.9 Measuring model fit: squared-error loss

The aim of each modelling algorithm is to model the response as accurately as possible subject to the model constraints, given the information contained in the training data set. The question then naturally arises of how to measure the fit of a model to the data. Due to mathematical tractability, by far the most commonly used *loss function* $L[Y, \hat{f}(\mathbf{x}_p)]$ in regression problems is *squared-error loss*

$$L[Y, \hat{f}(\mathbf{x}_p)] = [Y - \hat{f}(\mathbf{x}_p)]^2, \quad (2.4.27)$$

where Y denotes the true response and $\hat{f}(\mathbf{x}_p)$ the model prediction. When using squared-error loss, the prediction that minimises the prediction error of an observation \mathbf{x}_p for a quantitative response under the model restrictions is the conditional mean $\hat{Y} = \hat{f}(\mathbf{x}_p) = E_{Y|\mathbf{X}_p}[Y | \mathbf{X}_p = \mathbf{x}_p]$ (compare with (2.4.2) on page 19, which does not involve squared-error loss). This conditional mean is estimated from the training data.

Squared-error loss measures the *square* of the difference between the true response y and the model prediction $\hat{y} = \hat{f}(\mathbf{x}_p)$. A small difference between y and \hat{y} will result in a small loss. However, $L[Y, \hat{f}(\mathbf{x}_p)]$ increases *quadratically* with a *linear* increase in the distance between y and \hat{y} , with the repercussion

that outlier observations are penalised disproportionately to their divergence from their model predictions. This is advantageous if the goal is to model the whole process (including outlier values) as accurately as possible. However, if the data contain observations that are considered less important to be able to predict accurately, such as rarely occurring observations and inaccurate observations, then measuring model fit using squared-error loss can seriously hinder the modelling process, resulting in a model that sacrifices accuracy amongst the commonly occurring observations in order to accommodate more unusual observations.

Alternative measures of model fit are available for regression problems that are more robust to outlier observations, including *absolute-error loss*

$$L[Y, \hat{f}(\mathbf{x}_p)] = |Y - \hat{f}(\mathbf{x}_p)|. \quad (2.4.28)$$

Reporting model fit in terms of absolute-error loss is more convenient for the end user, since it is in the same units as the response variable. Unfortunately, absolute-error loss has the major drawback that it is not differentiable, which makes optimisation far more challenging. Hence, squared-error loss remains the most popular loss function in regression.

2.4.10 Selecting model parameters via cross-validation

Most model parameters, such as the intercept and slope coefficients in MLR (Section 2.4.2.2 on page 22) and the variables and cutpoints in regression trees (Section 2.4.4.2 on page 36), are selected to fit the training data as closely as possible i.e. to minimise the training error. However, certain parameters cannot be optimised directly on the training data set. For example, when choosing the optimum tree depth for a pruned regression tree (Section 2.4.4.3 on page 37), the deepest tree will always fit the training data most closely and therefore minimise the training error. It, however, does not necessarily follow that the deepest tree will also predict future observations accurately i.e. will minimise the generalisation error. Indeed, the deepest tree is likely to overfit the training data, and will therefore tend to predict observations from the larger population less accurately than some of its shallower counterparts. Other examples of parameters that regulate the bias-variance trade-off of the model (see Section 2.4.11 on page 59) include model size in MLR, the tuning parameter λ in the lasso (Section 2.4.3 on page 31), the number of input

variables to be considered at each split m in RFs (Section 2.4.6 on page 48) and the shrinkage parameter ν in boosting (Section 2.4.7 on page 49).

These *complexity parameters* need to be optimised in another way to obtain a good generalisation error. Ideally an independent set of observations should be used for this purpose. For example, the nested set of pruned regression trees $\{T_0, T_1, T_2, \dots\}$ would be fit to the training data set, and the pruned tree achieving the lowest prediction error on the independent data set would be chosen as the optimum regression tree model. However, data are usually limited, and all of the available observations are therefore needed for training the model.

One option is to adjust the training error in order to make it a less biased estimator of the generalisation error. Statistics that follow this approach are discussed in Section 2.4.2.6 on page 28. However, these statistics require a knowledge of model size, which is not always obvious.

An alternative strategy that directly estimates the expected generalisation error is *K-fold cross-validation* (*K-fold CV*). Let α represent the complexity parameter of interest for the model. *K-fold CV* involves estimating the generalisation error from the training set by calculating the prediction error on observations not used for fitting the model. It proceeds as follows: The training observations are randomly partitioned into $K < n$ groups or *folds*, with each fold containing $\pm n/K$ training observations. For each of the candidate values of α , the model is fit K times to a data set containing $K - 1$ of the folds combined, while determining the prediction error on the left-out k^{th} fold (with a different group of observations being used as the *held-out fold* each time). This provides K prediction errors for the value of α being evaluated, one from each of the K folds. Since each of the K folds contains different training observations, these prediction errors will be quite different from one another. The K prediction errors are averaged to reach a more stable estimator of the generalisation error for that particular value of α . After following the same procedure for all of the candidate values, the α value that achieves the smallest average prediction error is then chosen as the optimum value for α . Since each prediction error is calculated using observations not used for fitting the model, the prediction errors obtained from *K-fold CV* are valid estimators of the generalisation error. K is typically chosen to be 5 or 10, since these values have been found to strike a good balance between the bias and

the variance of the resulting estimators.

CV is therefore a good way of estimating the generalisation error when data are limited, since the same data set can be used for both model fitting and model selection. For a particular value of α , K models are fit, each to a data set not much smaller than the training set (about $n - n/K = n \cdot (K - 1)/K$ in size). Moreover, all of the training observations are used as a test observation during the process. Consequently, CV makes maximum use of the data to estimate the generalisation error. If CV is being used to determine the optimal flexibility of the model, then the exact generalisation errors for the different values of the complexity parameter are not actually of that much interest anyway—primary interest rather lies in the complexity parameter value that minimises the generalisation error, irrespective of the actual value of the minimum. CV makes very few assumptions regarding the true model, and is therefore a technique that is applicable to diverse modelling techniques.

2.4.10.1 Performing cross-validation in R

There are a number of packages available in R for optimising the complexity parameter of various modelling techniques using CV. Two of these packages that are used in this study are the `caret` package (Kuhn, 2015a) and the `e1071` package (Meyer *et al.*, 2014). Both of these packages can implement 10-fold CV.

2.4.11 The generalisation error of a model

The statistical modelling techniques described above—MLR, the lasso, regression trees, bagging, RFs and boosting—all fit prespecified relationships to the observations in the training data set in order to model the quantitative response variable. Modelling algorithms tend to fit the model that minimises the squared difference between the model predictions and the training responses i.e. minimises the RSS_{train} or a function thereof. One is, however, usually not especially interested in predicting the training observations accurately i.e. minimising the *training error*. The training observations have already occurred, and it is therefore of less interest to be able to predict the training responses than it is to predict the response of future observations. Hence, a more appro-

appropriate test for choosing the best model is to determine the model that achieves the lowest error when predicting *unseen observations* i.e. observations that the model was not trained on. As explained in the previous section, due to random noise present in all data sets, including the training data set, a model that minimises the training error is unlikely to minimise the error achieved on independent observations—the *generalisation error* (also known as the *expected prediction error* or EPE). More than one data set is therefore needed. The random observations are therefore usually partitioned amongst a training set and a *validation set*. The statistical models are fit to the observations in the training set, and their relative generalisation errors are estimated using the independent validation set for model selection. A third data set—the *test set*—can also be used in order to provide an unbiased estimator of the generalisation error of the best model.

Since the training, validation and test sets are used for different purposes in the modelling process, they provide different information regarding the fit of the model. A high training and high validation error is indicative of a poorly fitting model, which can be the result of applying a technique that makes inappropriate assumptions regarding the underlying relationship and/or an insufficient number of training observations for estimating the model parameters. On the other hand, a low training error does not necessarily indicate a good model. Since every data set contains a noise component, a model that is fit too closely to the training data will follow the noise too closely and will therefore not generalise well to unseen cases (see Section 2.4.1 on page 18). Consequently, a low training error can be the result of overfitting, and the validation error is more informative here. A low training and a high validation error suggests that the model has been overfit to the training data, resulting in it performing poorly on unseen cases, while both low training and low validation errors suggest a good model—a model that will in general predict the response of observations reasonably accurately.

While the training and validation errors together provide an indication of the model that is likely to predict the response variable most accurately, they do not tend to provide good estimators of how accurately the best model is likely to predict the response. Since the responses of the observations that happen to make up the validation set were predicted most accurately by this model, the validation error is biased in favour of this model. The test set,

which contains a third set of random observations, was not used to either train the model or choose the best model, and is therefore not biased in favour of any particular model. It will therefore provide a more accurate estimator of the prediction error of the best model.

For the regression model in (2.4.1) on page 19 with $E(\varepsilon) = 0$ and $\text{Var}(\varepsilon) = \sigma_\varepsilon^2$, the expected prediction error of an unseen case \mathbf{x}_0 using squared-error loss is given by

$$\begin{aligned} \text{EPE}(\mathbf{x}_0) &= E \left\{ \left[Y_0 - \hat{Y}_0 \right]^2 \right\} = E \left\{ \left[Y_0 - \hat{f}(\mathbf{x}_0) \right]^2 \right\} \\ &= \sigma_\varepsilon^2 + \left\{ \text{Bias} \left[\hat{f}(\mathbf{x}_0) \right] \right\}^2 + \text{Var} \left[\hat{f}(\mathbf{x}_0) \right]. \end{aligned} \quad (2.4.29)$$

Consequently, the accuracy with which the response of an unseen observation \mathbf{x}_0 can be predicted from a model \hat{f} is determined by the model's bias and variance, as well as by the *irreducible error* σ_ε^2 , which is the variance in the response that cannot be explained by the input variables included in the model. Hence, due to the ever-present irreducible error (which is caused by measurement error and unobserved influences on the response), the prediction error of future observations can never be reduced to zero, even if f were known.

In contrast, the *bias* of $\hat{f}(\mathbf{x}_0)$ —the difference between the expected value of the estimated function $E[\hat{f}(\mathbf{x}_0)]$ and true function f —and the *variance* of $\hat{f}(\mathbf{x}_0)$ —a measure of the spread of $\hat{f}(\mathbf{x}_0)$ around its mean—can be controlled. In general, all else being equal, a more flexible model will have less bias but greater variance, while a more rigid model will have less variance but more bias. Hence, EPE can be minimised for a particular model type by regulating the bias-variance trade-off to choose the flexibility (*model complexity*) that will minimise the combined bias and variance of the model. This is achieved in the lasso by selecting the tuning parameter that finds the optimum balance between minimising the training error and reducing the variance of the model coefficient estimators, and in the tree-based methods by selecting the optimum tree depth and other parameters that control the flexibility of the tree model. These parameters cannot be chosen directly from the training data set, since the most flexible model will always minimise the training error as a result of overfitting to the training data, resulting in a model that will generalise poorly. The parameters are therefore chosen using independent observations or using a method such as CV (see Section 2.4.10 on page 57).

In general, the more observations present in the training data set, the more information available for modelling. More complex (and therefore less biased) models can therefore be fit to the data without incurring more variance. Thus, the most effective way to increase the predictive power of a specific model type given a particular response and set of input variables is to increase the number of random observations available for modelling. However, given enough observations, a more effective way to increase predictive power is usually to include more informative input variables in the model.

2.4.12 Variable selection

Statistical modelling involves finding a useful approximation \hat{f} to the true relationship f between the response and the input variables in (2.4.1) on page 19 using modelling techniques such as those outlined in this chapter. Geometrically, this can be thought of as fitting a response surface \hat{f} above the space spanned by the p input variables in Euclidean space. From this perspective, modelling takes place in the *input-output space*, which contains the estimated function \hat{f} hovering over the space consisting of all the input variable value combinations occurring in the population of interest—the *input space*. The information necessary for estimating the functional form of f is contained in the data points or observations in the training data set, each of which constitute an independent observation of specific input variable values and the resulting response. The input variable values of the n observations determine their positions in the input space, and the response value of each data point contributes to estimation of the height of f above the input space in that area. The training observations together form the *training hull*.

Function approximation therefore takes place in the input-output space, which is a $(p+1)$ -D space (if all of the input variables are continuous variables) consisting of the n training observations clustered within a space spanned by the p input variables and the response variable. The response variable spans one dimension, and it is the number of input variables that determines the dimension of the input-output space. Consequently, adding more training observations to a modelling analysis involves adding extra observations to the $(p+1)$ -D input-output space, whereas adding an extra input variable entails adding a whole extra dimension to the input-output space in which function approximation occurs. Thus, the more input variables included in the model,

the higher will be the dimensionality of the input-output space. An increase in dimensionality in turn causes the volume of the input-output space to increase exponentially, causing the distances between the observations to increase and the density of observations to plummet in the training hull. Since the response surface is estimated from the observed values of the response at different input values, a lower density of observations will result in a less accurate model. Hence, a linear increase in input variables requires an exponential increase in observations in order to maintain model accuracy at a constant level. In addition to a larger volume and greater distances, high dimensionality also results in a greater proportion of an object's volume being located close to the object's surface rather than towards its interior. Consequently, an additional problem caused by an increase in dimensionality in statistical modelling is that a greater proportion of the observations are located close to the surface of the training hull as opposed to more towards its interior. This means that unseen observations are also more likely to be situated closer to an edge of the training hull rather than to any of the training points. Since extrapolation is far less accurate than interpolation, prediction accuracy for unseen observations also falls as a result. The drawbacks of including numerous input variables in a statistical analysis involving relatively few training observations i.e. the drawbacks of a low n/p ratio, are collectively referred to as the “*curse of dimensionality*” (CoD). Parametric models mitigate the CoD by imposing heavy restrictions on the relationships allowed between the response and the input variables, making parametric models far more powerful than nonparametric models in high-dimensional problems.

In addition to the problems directly associated with including many input variables in an analysis outlined in the previous paragraph, a large number of input variables also makes phenomena such as multicollinearity more likely. *Multicollinearity* refers to the presence of groups of strongly correlated input variables in a data set. This tends to lead to the model coefficient estimators having higher variances due to the increased difficulty that the modelling algorithm faces in distinguishing the effects of the individual input variables on the response (also see Section 2.4.2.2 on page 22). Furthermore, the inclusion of *noisy variables*—variables that are only weakly dependent on or independent of the response and are therefore uninformative for predicting the response—adds to the noise in the training data set, and is therefore likely to lead to

inferior models.

In summary, the inclusion of too many variables in a statistical analysis tends to result in several problems culminating in less accurate modelling, including dimensionality problems and multicollinearity. A balance therefore has to be struck between the usefulness of each input variable and the problems that will be caused by adding extra dimensions to the analysis. It is advisable to use only the most informative input variables when developing statistical models. This has the additional advantage of making the resulting model easier to understand.

Variable selection is therefore an important step in any statistical analysis involving more than a few input variables. Diverse methods have been proposed over the years to identify the most informative input variables prior to an analysis. The lasso (Section 2.4.3 on page 31) and tree-based methods (Sections 2.4.4 to 2.4.7 on pages 34–49) naturally perform variable selection. Another approach is best subset selection, described next.

2.4.12.1 Best subset selection

Best subset selection is a relatively old variable selection technique that is applicable to a number of modelling techniques. It fits models containing all 2^p possible subsets of the p input variables, and chooses the best model from amongst these. It achieves this by selecting the model of each size $d \in \{1, 2, \dots, p\}$ that scores the lowest training error, and then chooses the best model overall amongst the $p + 1$ model sizes (the null model and the p model sizes including one or more input variables) using techniques such as CV (Section 2.4.10 on page 57), the adjusted R^2 , the C_p statistic or BIC (Section 2.4.2.6 on page 28).

Best subset selection makes it possible to assess models containing all possible subsets of the p input variables. However, a drawback, especially when a large number of input variables is involved, is that there is a chance of a model being selected that does well on the held-out training observations during CV as a result of the model fitting the noise unique to the training data set well (rather than the signal). Such a model would necessarily predict general observations from the population poorly. Best subset selection in the presence of numerous input variables is a highly flexible technique that can suffer from high variance. Another more obvious disadvantage is that the number of pos-

sible subsets grows exponentially as p increases, making best subset selection impractical in the case of numerous input variables (more than 40).

Best subset selection can be performed in R using the `leaps` package (Lumley, 2009).

In summary, in regression the response is modelled as a static function of input variables that are suspected of influencing the response. The resulting models can be used as an objective basis for understanding the influences of the input variables on the response, and for predicting the response of future observations. The nature of the observations sourced for the modelling process determines the applicability of the models: Models developed from random observations of the process are generally applicable to predicting future observations, while models based on the most recent observations of a changing system are likely to predict observations in the immediate future most accurately. Since each model type makes different assumptions regarding the underlying relationship, modelling the process using different techniques enables the process to be investigated from different angles, resulting in a deeper understanding of relationships.

2.5 Aims of this study

The system currently used on the tomato farm for anticipating harvest quantities is simply based on weekly averages of the harvest quantities obtained over recent years. Two sigmoidal curves are generated—one for the summer and another for the winter—and the quantity of tomatoes that will be picked in any given week is estimated by reading off the height of the appropriate curve for that week. This system provides an indication of future harvest quantities based on previous experience. However, the resulting estimates are not very accurate, especially in the presence of abnormal weather conditions such as those encountered in El Niño years. A method that predicts harvest quantities based on conditions that have had a direct impact on the development of the crop plants is likely to produce more accurate predictions.

The current study was therefore formulated in order to:

- investigate the roles that various planting, harvest and weather variables play in determining the harvest of a tomato crop,

- explore the utility of different statistical techniques for modelling harvest quantities, and
- develop statistical models for accurately predicting crop harvest from weather variables for this tomato farm.

The following chapter provides a detailed description of the data sets used in pursuing these objectives. Chapters 4 to 6 then focus on applying interpretive statistical models to untangle the effects of the various weather variables on the harvest density of tomato crops. Predictive statistical models are also applied in order to develop models that will generate accurate predictions of harvest density.

Chapter 3

Data description and exploratory analyses

The analyses presented in this document are based on data from three sources. The responses as well as some of the predictor variables come from the crop data set, which is described in the next section. However, many of the predictors are weather variables. These were obtained from two different sources, and are outlined in the following section. Analyses of the merged harvest and weather data sets are presented in the final section.

3.1 The tomato crop data set

The records of field-grown crops planted between 2008 and 2015 were obtained from a South African tomato farm in November of 2015. This farm is part of a farming conglomerate that produces tomatoes for both local and overseas markets, and therefore grows tomatoes on a large scale (well over 1 000 ha/year). Although this farm has been producing tomatoes commercially for several decades, agricultural practices have changed over time in response to advances in technology and knowledge. The period 2008–2015 was therefore chosen to minimise the amount of variance in the yield (and therefore in the harvest quantities) due to changes in cultivation practices and tomato cultivars.

3.1.1 Crop care on the tomato farm in brief

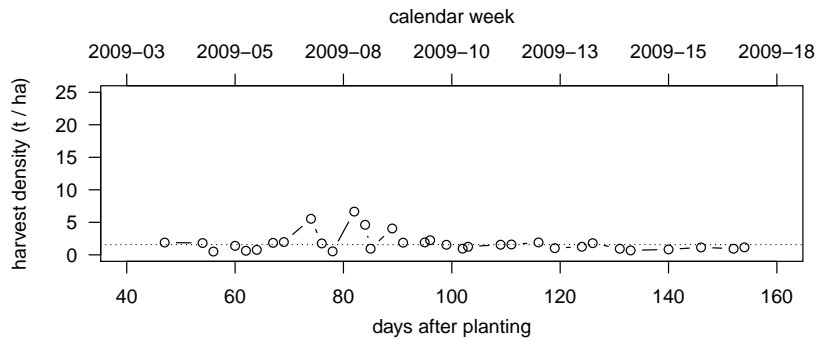
Tomato plants grown on the farm belong to one of a few commonly grown commercial tomato cultivars of the same type. These cultivars are closely related to one another, and genetic differences amongst the crop plants are not expected to introduce much variance into the analyses.

The tomato plants start out life in a nursery. The seeds are sown in seed trays and the plants are nurtured in greenhouses under carefully controlled conditions for the first 4 to 6 weeks (depending on the weather) of their lives. From the nursery the seedlings are planted out into the open field. Tomatoes planted in a particular field in a particular year constitute a *field crop*. Fields are up to about 60 ha in size, and to facilitate the management of the tomato crops the fields are partitioned into 1 to 27 blocks of up to about 5 ha each. Blocks constitute the lowest level at which the crops are managed on the farm, and most of the farm's data records are therefore in terms of the *block crops*. The harvest quantities of the block crops were therefore modelled in this study.

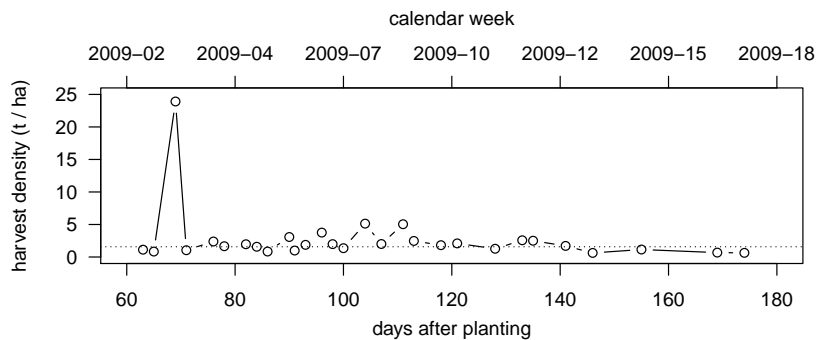
The plants are trained up poles for support, and are pruned back when they reach the top of the support structures. The height of the poles in the fields therefore limits the size of the mature plants, which in turn limits the duration of the field-grown tomato crops and their cumulative harvests. Moisture levels in the soil are closely monitored, and irrigation is adjusted accordingly. Hence the crops are not dependent on rainfall patterns for their survival. The crops are fertilised on a regular basis, and are treated at the first sight of pests or diseases. Consequently, cultivation practices are kept relatively constant across crops, and are therefore unlikely to be responsible for large variations in yield.

3.1.2 Harvesting on the tomato farm

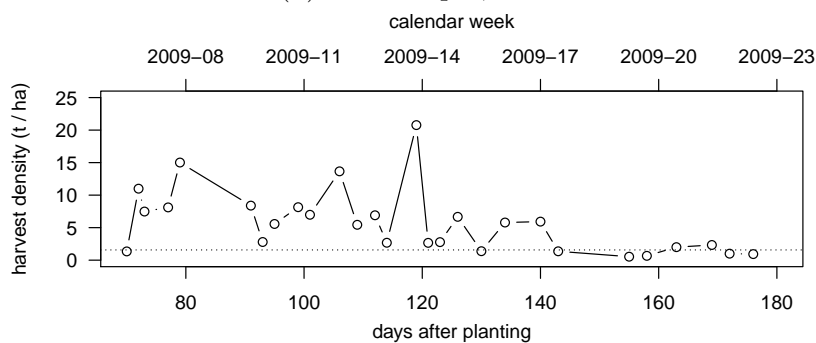
Tomato plants tend to mature faster in warmer conditions, and summer crops can therefore be harvested sooner than winter crops. The crops start producing fruit anywhere from 39 to 116 days after planting out into the fields i.e. the crops have *growing periods* spanning about 6 to 17 weeks, after which the crops are harvested on a regular basis (Figures 3.1 to 3.3 on pages 69–71). The staff harvests tomatoes for up to 137 days i.e. the crops' *harvest periods* are up to about 20 weeks long, after which the field is ploughed up and prepared for the



(a) Field crop 3; Block 5B

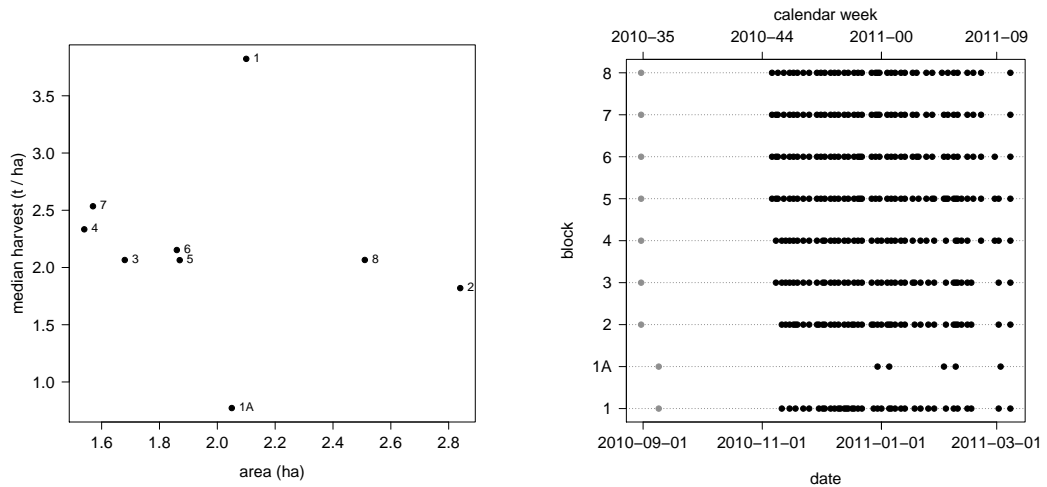


(b) Field crop 3; Block 7A

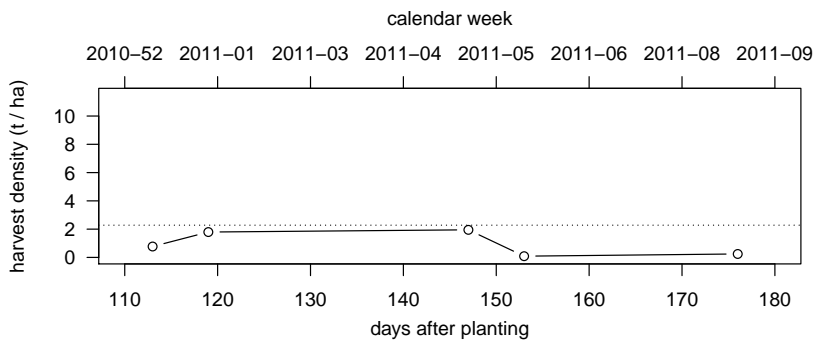


(c) Field crop 3; Block 16B

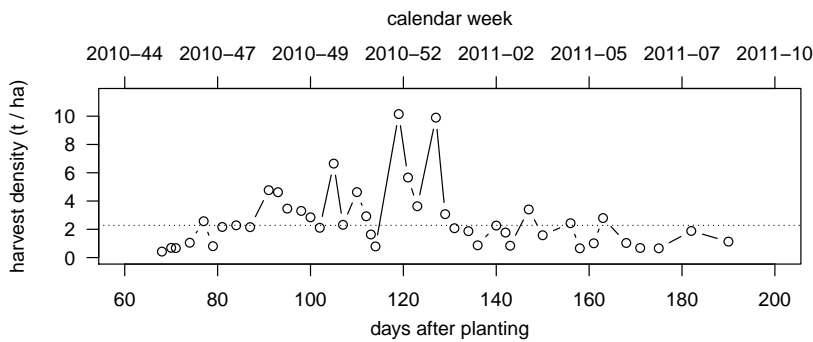
Figure 3.1: Harvest events of 3 of the 27 block crops in Field crop 3. In each case, harvest density (tonnes/hectare) is shown on the vertical axis, while days after planting and calendar week are shown on the bottom and top axes, respectively. The median harvest density of the field crop is indicated by the horizontal dotted line. Block 7A was excluded from the modelling analyses due to the unrealistically high harvest peak towards the beginning of its harvest time series. Most of the harvest events for Block 16B are well above this field crop's median harvest density, which led to Block 16B also being excluded (see Section 3.1.3 on page 72 for the criteria applied to clean the crop data). Block 5B was retained, and is shown here for comparative purposes.



(a) median harvest densities of block crops in Field crop 20 (b) planting (grey) and harvest (black) dates of Field crop 20

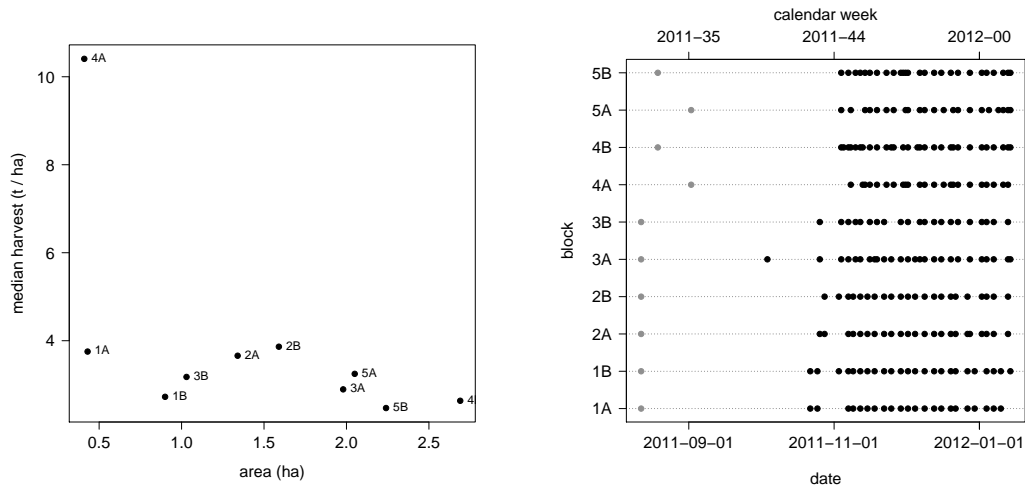


(c) Field crop 20; Block 1A

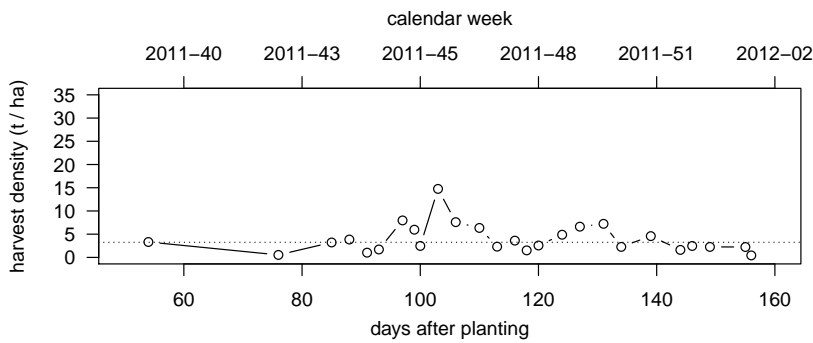


(d) Field crop 20; Block 6

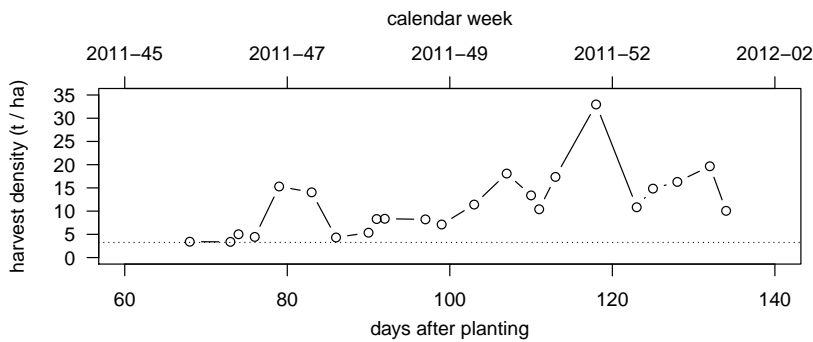
Figure 3.2: Planting and harvest events of 2 of the 9 block crops in Field crop 20. Block 1A was excluded from the modelling analyses because its harvest densities fall below its field crop median, it has unusually long inter-harvest intervals, and it has very few harvest events. Block 6 was retained, and is shown here for comparative purposes.



(a) median harvest densities of block crops in Field crop 32 (b) planting (grey) and harvest (black) dates of Field crop 32



(c) Field crop 32; Block 3A



(d) Field crop 32; Block 4A

Figure 3.3: Planting and harvest events of 2 of the 10 block crops in Field crop 32. Block 3A was excluded due to the unusual timing of its first harvest event, while Block 4A was excluded due to its exceptionally high harvest density values.

next crop. In total, crops last for about 16 to 32 weeks from planting of the seedlings to the last harvest date i.e. *crop duration* is up to about 8 months.

The field crops are harvested in blocks: A picking team typically starts at one end of a field, and works its way linearly through the field to the other end. Consequently, it can take several days (or a few weeks towards the end of the harvest period because of less regular visits to the field) for picking teams to make their way through all of the blocks in a field. The crops tend to produce the most fruit during the first 2 to 7 weeks of the harvest period, during which the block crops are harvested on a weekly or bi-weekly basis. After this initial harvest period, yield slows down, as does the regularity of the harvest events. Having said this, there is a lot of variation in tomato quantity and harvest frequency over the harvest period (see Figures 3.1 to 3.3). This variation can be ascribed in part to the weather: A cold snap during the harvest period can result in far fewer tomatoes being harvested over subsequent days, while a sudden warm spell tends to speed up ripening considerably, resulting in substantial loss of produce if the fruit is not picked and sold quickly over the next few days. The weather can also have more long-lasting effects: Weather conditions during the growing period affect plant development, and adverse weather conditions can result in the tomato plants failing to reach their full yield potential. Even the prevalence of pests and diseases is, at least in part, driven by the weather (De Bruyn, pers. comm.).

Hence, due to the relative uniformity of the cultivation practices exercised on the tomato farm, the chief source of variance in the success (in terms of harvest quantities) of different crops is expected to be caused by the weather. This project was therefore initiated in order to model a key aspect of variation in crop harvest quantities—the weather.

3.1.3 Data set description and cleaning

The original crop data set contains planting and harvest information of 943 field-grown block crops (constituting 96 field crops (grown in 56 different fields)) cultivated from 2008 to 2015 on the tomato farm. The variables present in this data set are listed in Table 3.1 on the next page. Since the areas of some of the blocks differ from year to year, it is assumed that the fields remain constant but that a field is repartitioned into blocks during the planting of each field crop.

In addition to those listed in Table 3.1, the following variables were added:

Table 3.1: Variables in the tomato farm’s crop data set. The values of the variables (displayed in the third column) reflect the range of values in the final (cleaned) data set. Refer to Figures 3.4a to 3.14a on pages 76–79 for an idea of the value ranges in the original data.

variable	description	type (and values)	units/format
Project	Identification number assigned to one or more concomitant field crops.	integer	-
Field	Field in which the block crop was grown.	categorical; 56 different field names	-
Block	Part of the field in which the crop was grown.	categorical; 1–27 blocks per field	-
Ha	Area of the block crop.	real number; 0.18–4.56	ha
PIDate	Date on which the block crop was planted.	date; 2008/09/24–2015/05/20	yyyy/mm/dd
PIDensity	The number of seedlings planted in the block per unit area.	categorical; 11 500 or 13 300	plants/ha
HarvestDate	Date on which tomatoes were harvested from the block crop.	date; 2008/11/26–2015/11/09	yyyy/mm/dd
Ton	Mass of tomatoes harvested on HarvestDate . This variable is referred to as <i>harvest mass</i> in this document.	real number; 0.0205–48.5839	t
TonPerHa	Mass of tomatoes harvested per unit area of the block crop i.e. Ton/Ha . This variable is referred to as <i>harvest density</i> in this document.	real number; 0.0082–35.2683	t/ha

PIMonth month of the year in which the seedlings were planted into the block (value type: integer; value range: 3–12),

PIWeek calendar week in which the block crop was planted (value type: integer; value range: 10–51),

GrowD number of days between the planting of the seedlings into the block and the first harvest date (value type: integer; value range: 39–116),

HarvD number of days between the first and last harvest dates of the block crop (value type: integer; value range: 43–137),

NHarv total number of times (i.e. total number of dates) that tomatoes were harvested from the block crop (value type: integer; value range: 18–48), and

TotalD number of days between the planting of the seedlings into the block and the last harvest date i.e. the sum of **GrowD** and **HarvD** (value type: integer; value range: 109–223).

An initial examination of the crop data set revealed 29 block crop records all having missing harvest date, harvest mass and harvest density entries. Since the statistical techniques used in the subsequent modelling analyses are all supervised methods, the absence of response values resulted in these records being excluded.

In collaboration with an agricultural expert on the tomato farm, the remaining 915 block crop records were examined in detail, both to gain an understanding of the data and to identify suspicious-looking entries. Several telephonic conversations culminated in the formulation of the following criteria for detecting possibly erroneous crop records:

Unusually high harvest density for field crop As stated previously, picking teams work their way linearly through blocks in a field during the course of a day. Since the smallest blocks tend to be located around the edges of the fields, picking teams often end the day in one of the smaller blocks of the field. This sometimes results in tomatoes harvested in the larger blocks of the field being ascribed to one of the smaller blocks. For this reason, the smallest block crops that have exceptionally high median harvest densities (t/ha) compared to their field crops were excluded, e.g. Block 16B of Field crop 3 (Figure 3.1c on page 69) and Block 4A of Field crop 32 (Figures 3.3a & 3.3d on page 71).

Unusually low harvest density for field crop Due to the inconsistencies in harvest quantity allocation mentioned above, block crops with harvest densities far lower than those of the other block crops in the field were also excluded, e.g. Block 1A of Field crop 20 (Figure 3.2a & 3.2c on page 70).

Unusually long interharvest period Block crops having unusually long intervals between successive harvest events were excluded, e.g. Block 1A of Field crop 20 (Figure 3.2 b–c on page 70).

Outlier harvest dates Block crops having a harvest event outside the harvest periods of concomitant crops in the field were excluded, e.g. Block 3A of Field crop 32 (Figure 3.3 b–c on page 71).

Unusually high value in harvest time series Block crops with an exceptionally high harvest density (t/ha) relative to the other harvest densities of that block crop were excluded, e.g. Block 7A of Field crop 3 (Figure 3.1b on page 69).

Unusually few harvest events Block crops having very few harvest events were excluded, e.g. Block 1A of Field crop 20 (Figure 3.2c on page 70). This included the crops that had not finished being harvested at the time that the data set was obtained from the tomato farm.

Many of the inconsistencies in the crop records listed above could have been circumvented by modelling field crops instead of block crops. However, a field crop is typically planted over several weeks, and different blocks in a field are harvested on different days (see Figure 3.2b on page 70 and Figure 3.3b on page 71), making predictors such as the length of the growing period and beginning and end dates for the crop’s weather time series variables difficult to fix. Block crops (rather than field crop) were therefore modelled.

For comparative purposes, box-and-whisker plots of the quantitative crop variables in the data set used in the modelling analyses (the *final* or *cleaned* crop data set) are depicted alongside their *original* counterparts in Figures 3.4 to 3.14 on pages 76–79 below. The box-and-whisker plots were drawn using the `Boxplot` function in the `car` package (Fox and Weisberg, 2011), which is a wrapper function providing additional options for the standard `boxplot` function in the `graphics` package in R (R Core Team, 2014). The box-and-whisker plots show the 25th (the lower hinge), 50th (the median) and 75th (the upper hinge) percentiles, as well as outliers (the default settings of this function depict values located more than 1.5 times the box’s length away from the box’s border as outliers).

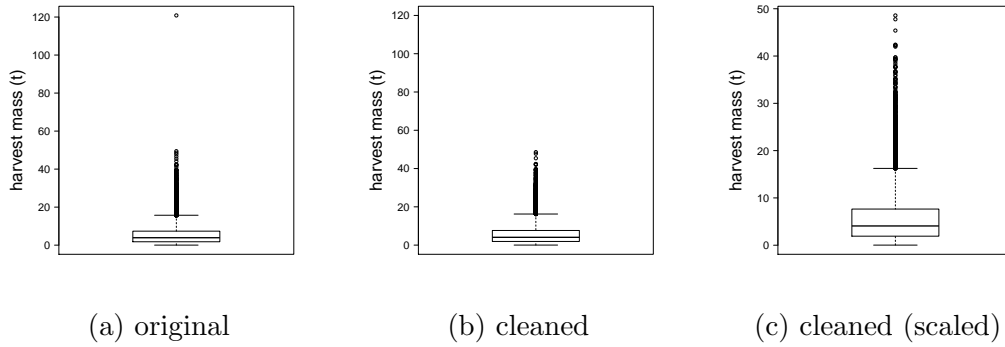


Figure 3.4: Original and cleaned distributions of the separate harvest masses of the block crops.

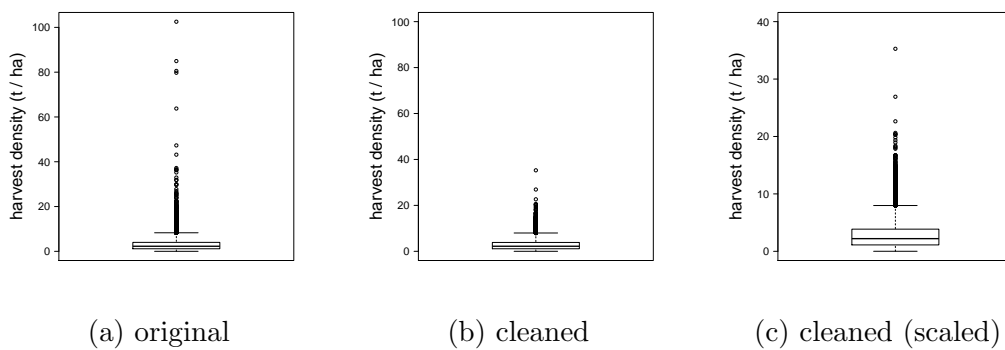


Figure 3.5: Original and cleaned distributions of the separate harvest densities of the block crops.

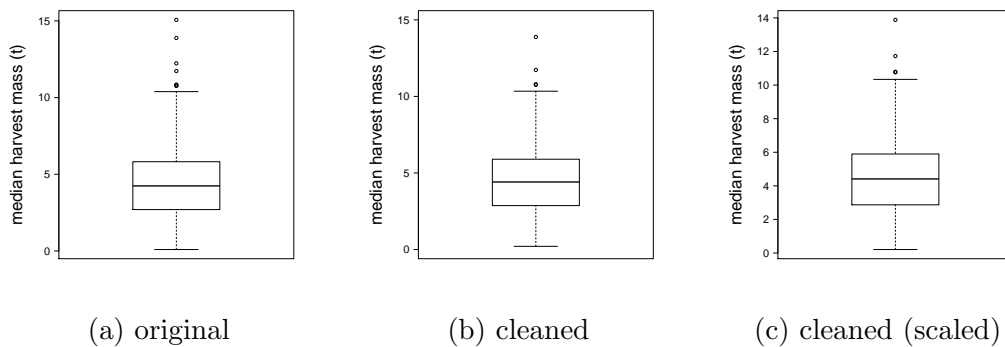


Figure 3.6: Original and cleaned distributions of the median harvest masses of the block crops.

The effects of the data-cleaning process are especially evident in the harvest density distributions (Figure 3.5, Figure 3.7 on the following page and Figure 3.9 on the next page), with Figures 3.5a, 3.7a and 3.9a containing extreme outliers that are absent from Figures 3.5b, 3.7b and 3.9b. The differ-

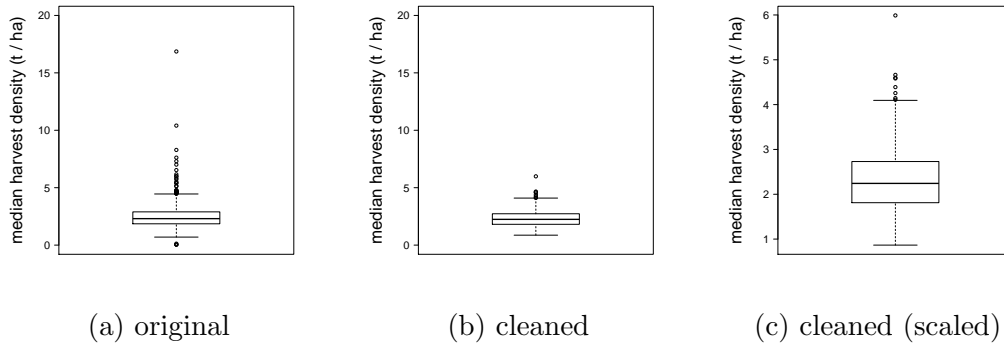


Figure 3.7: Original and cleaned distributions of the median harvest densities of the block crops.

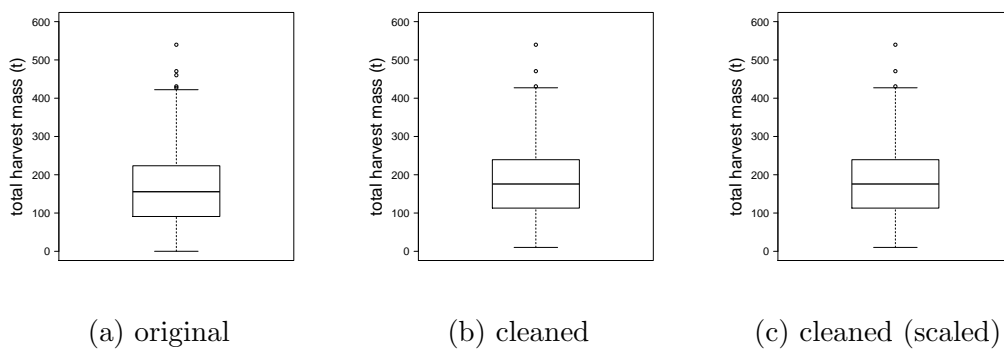


Figure 3.8: Original and cleaned distributions of the total harvest masses of the block crops.

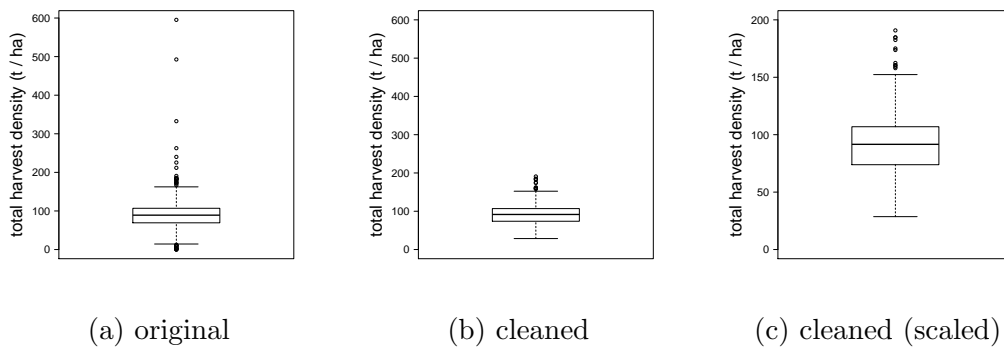


Figure 3.9: Original and cleaned distributions of the total harvest densities of the block crops.

ences between the original and cleaned harvest density distributions are mainly due to the exclusion of block crops having exceptionally high median harvest densities compared to their field crops.

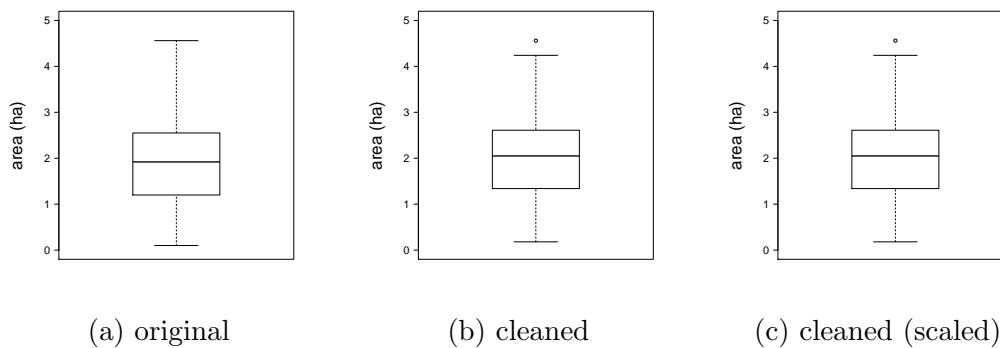


Figure 3.10: Original and cleaned distributions of the areas of the block crops.

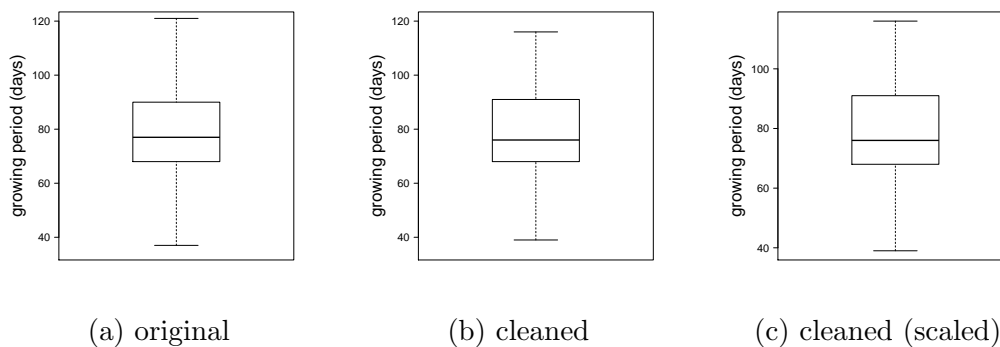


Figure 3.11: Original and cleaned distributions of the lengths of the growing periods of the block crops.

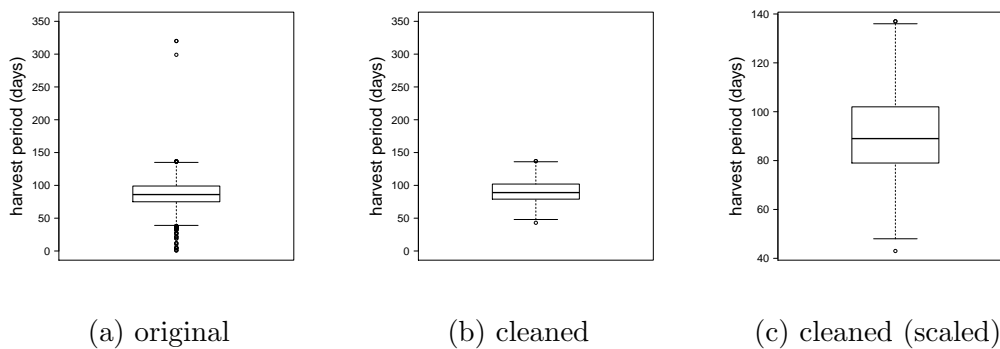


Figure 3.12: Original and cleaned distributions of the lengths of the harvest periods of the block crops.

3.1.4 The training, validation and test data sets

Applying the criteria outlined in the previous section to the crop data set brought the number of crop records down to a total of 738 block crops (representing 85 field crops). For the modelling analyses, the 738 observations were

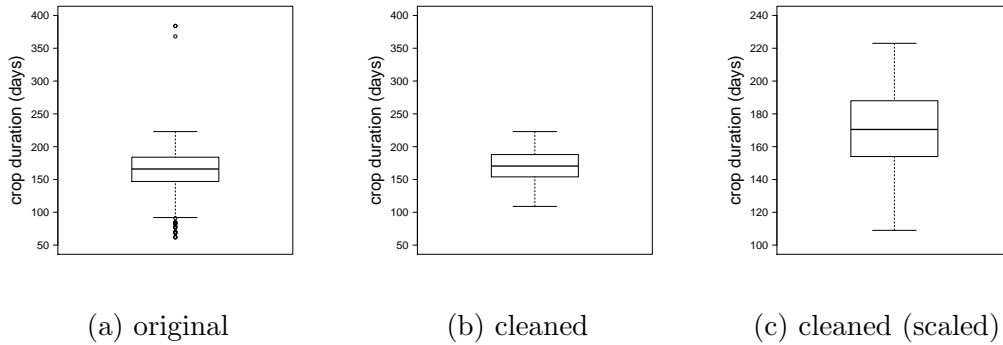


Figure 3.13: Original and cleaned distributions of the crop durations of the block crops.

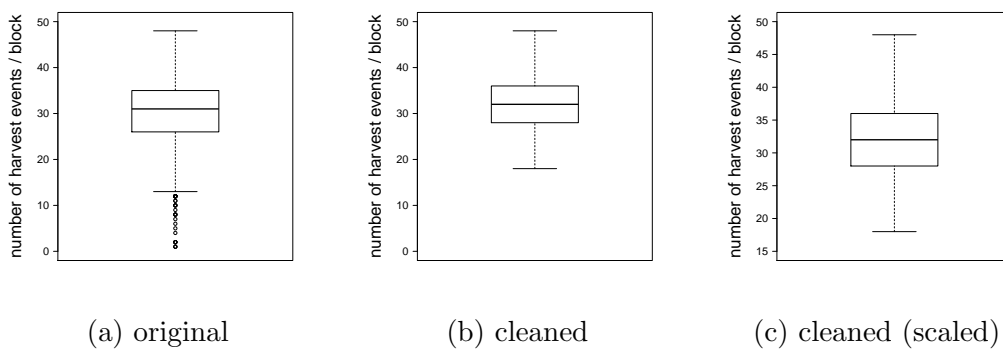


Figure 3.14: Original and cleaned distributions of the number of harvest events of the block crops.

randomly partitioned amongst a training (600 observations), a validation (88 observations) and a test (50 observations) set. These data sets were kept constant for all of the analyses in the remainder of this chapter and in subsequent chapters (Chapters 4 to 6).

Exploratory analyses, parameter optimisation and model fitting were performed using the training data set, while the accuracies of the different models were compared using the unseen observations in the validation set. The exploratory analyses were carried out only using the training observations, since some of the decisions regarding how to proceed with the modelling analyses were based on conclusions drawn from the exploratory analyses. The accuracy of the statistical model that achieved the lowest prediction error on the validation set was assessed using the test set in order to provide an independent and unbiased estimator of the prediction errors resulting from using the selected model (see Section 2.4.11 on page 59).

The R code for preparing the training, validation and test data sets is presented in the first part of Section A.1 on page 204.

3.1.5 Variables in the crop data set

3.1.5.1 Harvest quantities: harvest mass and density

The data set provided by the tomato farm (see Table 3.1 on page 73) contains information on *harvest quantities* i.e. the quantity of tomatoes harvested by the picking teams, in terms of *harvest mass* (Ton) and *harvest density* (TonPerHa). The aim of this study was therefore to develop a statistical model for predicting harvest quantity from crop and weather variables. However, harvest quantity can be seen as an approximation of crop *yield* i.e. the quantity of tomatoes produced by the crop plants, since most of the fruit is probably harvested. Hence, the models in this thesis can also be seen as models for crop yield.

Figure 3.15 displays the distributions in the training data set of the mass of tomatoes harvested from the block crops during individual harvest events (Figure 3.15a), as well as the median (Figure 3.15b) and total (Figure 3.15c) mass of tomatoes harvested over a block crop's harvest period. Figure 3.16 on the following page shows the corresponding harvest densities.

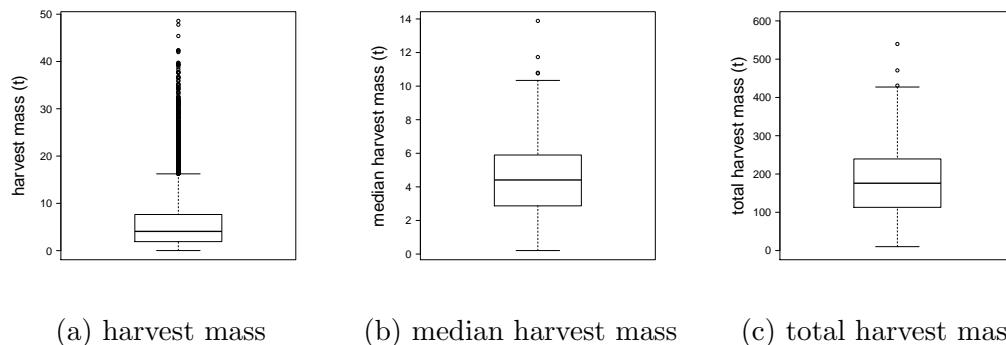


Figure 3.15: Distributions of the harvest masses of the training block crops.

Figures 3.15a and 3.16a show that the distribution of tomatoes collected during individual harvest events is skewed to the right, with the vast majority of harvest events involving small quantities of tomatoes. In contrast, the median and total harvest masses and densities (Figures 3.15 b–c and 3.16 b–c) have far more symmetric distributions, with only a few outlier values in each case.

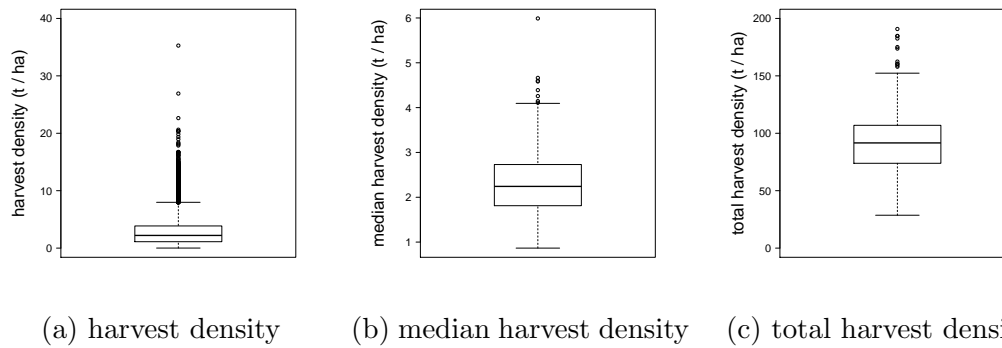


Figure 3.16: Distributions of the harvest densities of the training block crops.

Since harvest densities are more comparable between blocks than are harvest masses, median (`MedTonPerHa`; Figure 3.16b) and total (`TotTonPerHa`; Figure 3.16c) harvest density were chosen as the responses in the modelling algorithms of the next few chapters. The median is a *robust* statistic with a *breakdown point* of 50%, and modelling median harvest density should therefore make the models less sensitive to outlier values in the harvest time series of the training observations. Robustness is a desirable quality in practical applications, since the accuracy of the data can seldom (if ever) be assured beyond doubt. Moreover, it is often more useful to maximise the prediction accuracy of the most commonly occurring cases. However, although the median has many advantages, total harvest density was also modelled, since it was assumed that this quantity might be more useful to the farmer. Most of the remaining figures in this chapter therefore show harvest density (rather than harvest mass).

Figures 3.17 to 3.18 on the next page depict the median and total harvest densities in the training data obtained from the different fields from 2008 to 2015. There appears to be differences in productivity amongst the fields, which could be caused by factors such as differences in soil type and aspect.¹ This suggests that including the field name (`Field` in Table 3.1 on page 73) in the modelling analyses would help to unravel relationships. However, since incorporating a categorical variable into a statistical model such as MLR involves the estimation of a separate coefficient for all but one of the variable's distinct

¹The *aspect* of a field situated on a slope refers to the direction in which the field is facing. In the southern hemisphere, northern-facing fields receive much more sunlight than do southern-facing fields. For this reason northern-facing fields tend to achieve higher crop yields in the southern hemisphere than do their southern-facing counterparts, especially in milder climates.

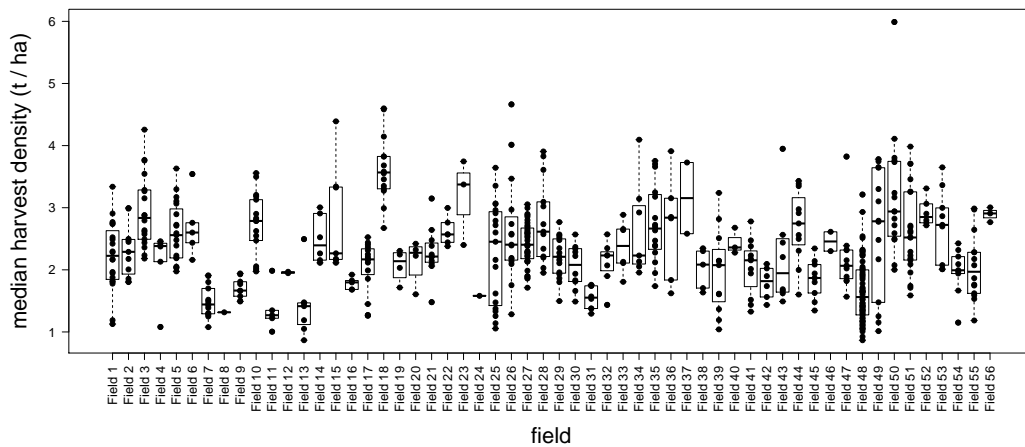


Figure 3.17: Median harvest densities of the fields in the training data.

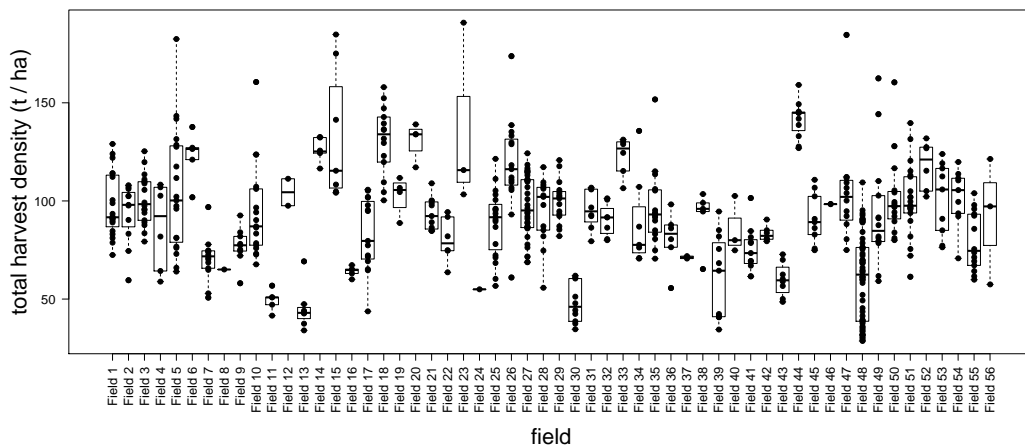


Figure 3.18: Total harvest densities of the fields in the training data.

values, additional caution needs to be exercised when weighing up the benefits of including categorical variables (see Section 2.4.2.7 on page 30). `Field` is a qualitative variable containing 56 distinct field names, each field represented by a median of 9 block crops in the training set, and including this variable is therefore likely to cause dimensionality problems in the analyses (see Section 2.4.12 on page 62). `Field` was therefore not used in the modelling analyses.

3.1.5.2 Block crop area

Figures 3.19 to 3.20 on the next page display the relationships between harvest mass, harvest density and block crop area in the training data. As can be

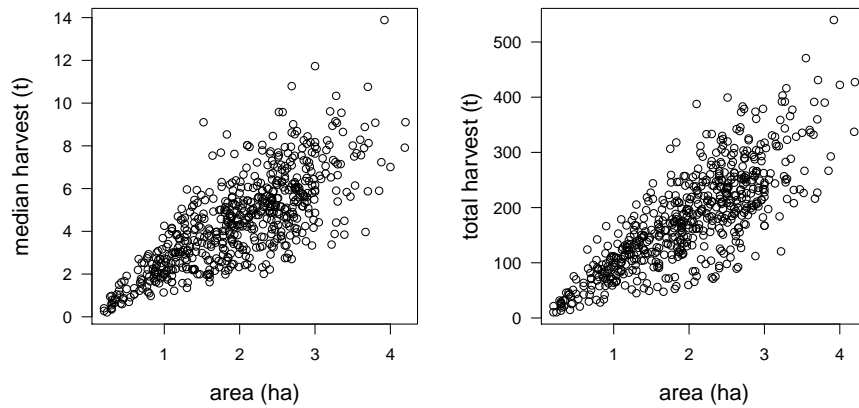


Figure 3.19: Harvest mass versus area for the training block crops.

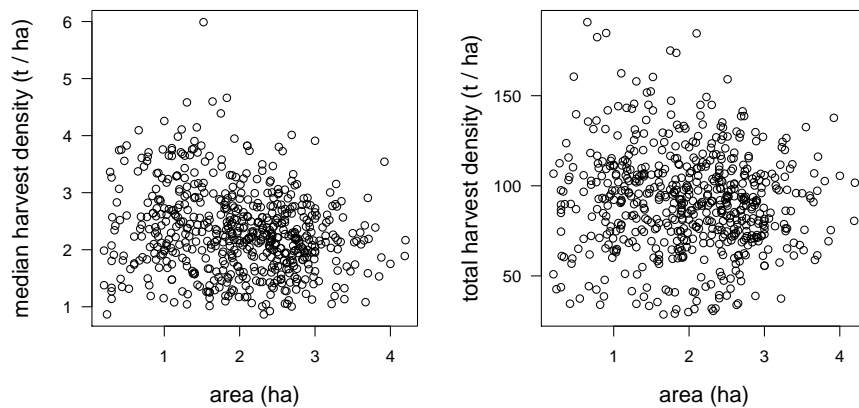


Figure 3.20: Harvest density versus area for the training block crops.

expected, a larger mass of tomatoes tends to be harvested from larger block crops (Figure 3.19), while correcting for area largely removes the dependence between harvest quantity and area (Figure 3.20).

3.1.5.3 Planting density

Figure 3.21 on the following page displays harvest density against planting density, a qualitative variable with two values. The relationship between these two variables appears to be rather ambiguous, depending on the summary statistic analysed: While a greater planting density is associated with higher total harvest densities (right-hand plot), there seems to be no relationship (or perhaps even a slightly negative relationship) between median harvest density and planting density (left-hand plot). This puzzling pattern, whereby total

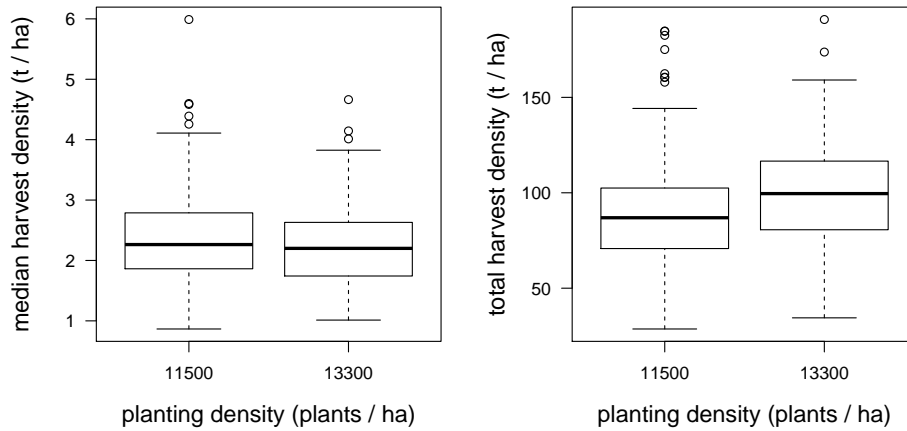


Figure 3.21: Harvest density versus planting density for the training block crops.

harvest density shows more intuitive relationships with the predictor variables than does median harvest density, is encountered throughout this study. An explanation for this is proposed towards the end of Chapter 6.

3.1.5.4 Planting time and lengths of the crop periods

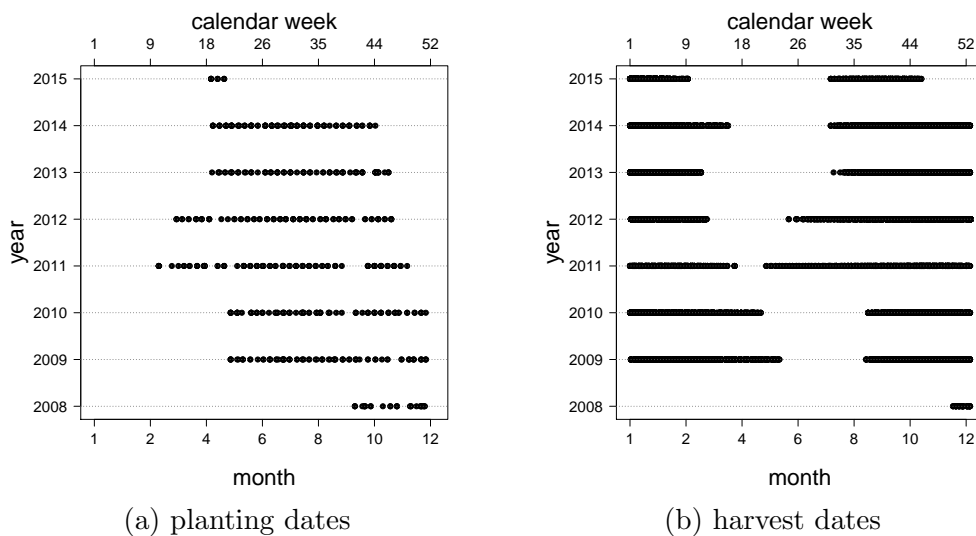


Figure 3.22: Planting and harvest dates for the training block crops.

As can be seen in Figure 3.22, the block crops are planted towards the middle of the year, and harvested at the end of the year and beginning of the

subsequent year on the tomato farm. An examination of Figures 3.23 to 3.24 indicates that this planting strategy maximises harvest density.

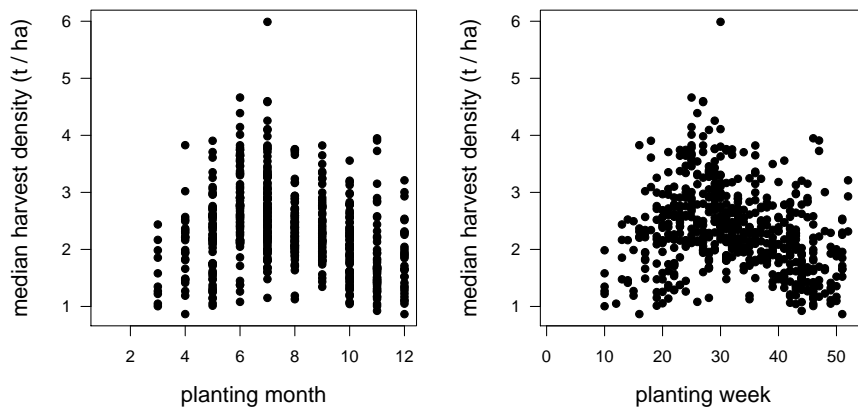


Figure 3.23: Median harvest density of the training block crops for different planting times.

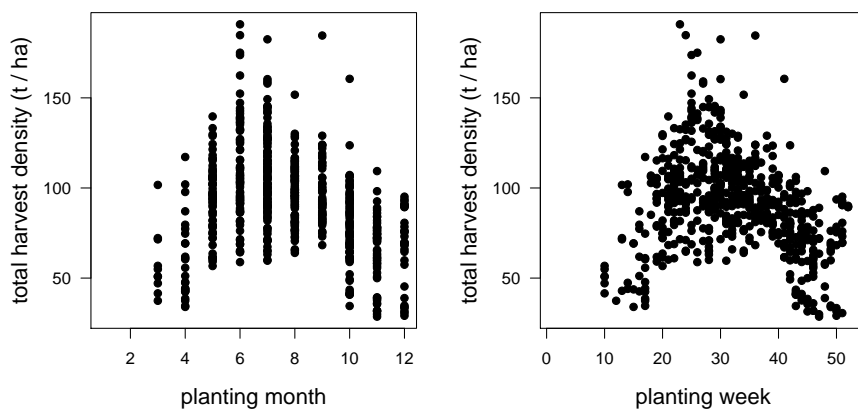


Figure 3.24: Total harvest density of the training block crops for different planting times.

However, planting in the middle of the year also has some drawbacks, in that it results in longer growing periods (Figure 3.25 on the next page). Hence, both the length of the growing period and harvest quantity are climate dependent, and the optimum planting time depends on the relative merits of maximising output by planting in the middle of the year versus minimising costs (in terms of irrigation, fertiliser, labour, etc.) by planting towards the beginning or end of the year.

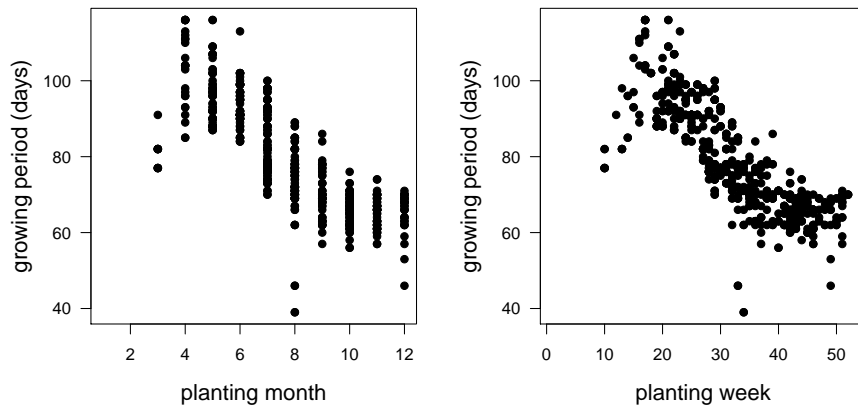


Figure 3.25: Growing period of the training block crops for different planting times.

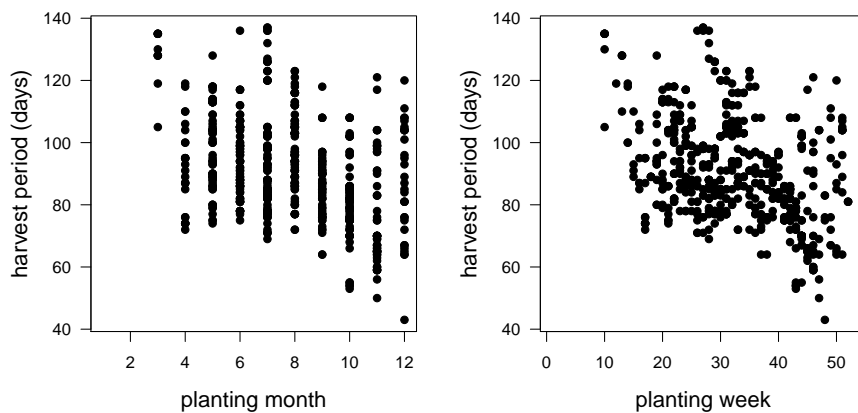


Figure 3.26: Harvest period of the training block crops for different planting times.

Although harvest period shows similar patterns to growing period for different planting times (Figure 3.26), the relationship between harvest period and planting time is less clear-cut. This could be due to the fact that, whereas growing period is largely out of a farmer's control—under ideal growing conditions (enough sunlight, water, fertiliser, etc.) the weather and the tomato cultivar largely determine how quickly plants reach maturity—the farmer has more control over the length of the harvest period in that he can decide for how long a crop is productive enough to continue harvesting its fruit. In other words, the weaker relationship between harvest period and planting time (as opposed to growing period and planting time) is probably the result of more (independent) factors determining the length of the harvest period.

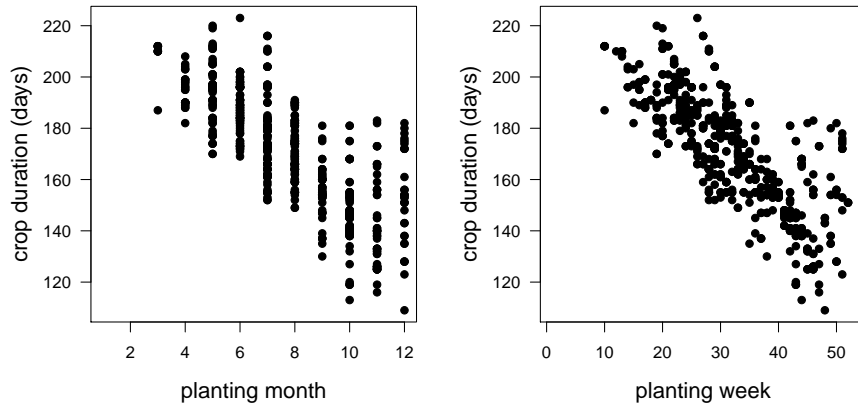


Figure 3.27: Crop duration of the training block crops for different planting times.

The dependence of growing (Figure 3.25 on the previous page) and harvest (Figure 3.26 on the preceding page) period on planting time in turn results in crop duration having a strong relationship with planting time (Figure 3.27).

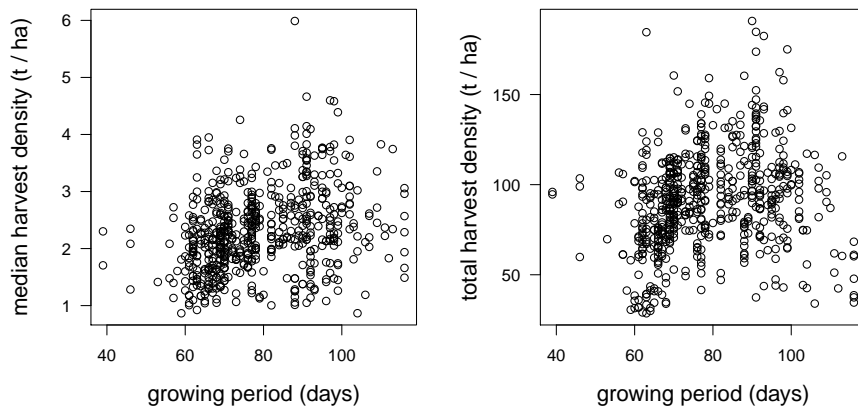


Figure 3.28: Harvest density versus the length of the growing period for the training block crops.

Figures 3.28 to 3.30 on pages 87–88 show median (left) and total (right) harvest density against the growing (Figure 3.28) and harvest (Figure 3.29) periods and crop duration (Figure 3.30). Figure 3.29 shows that total harvest density tends to increase with harvest period (right-hand plot in Figure 3.29), which is probably why this harvest quantity also increases with crop duration (Figure 3.30). A longer harvest period can be the result of the farmer continuing to harvest produce for a longer time due to economic reasons, e.g. a high

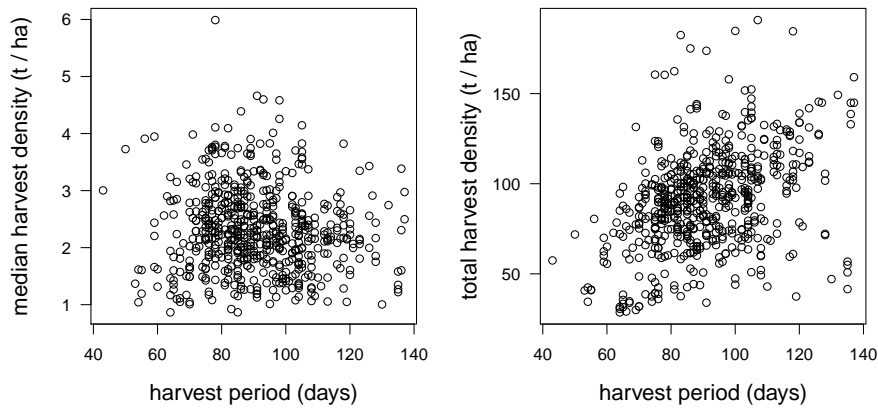


Figure 3.29: Harvest density versus the length of the harvest period for the training block crops.

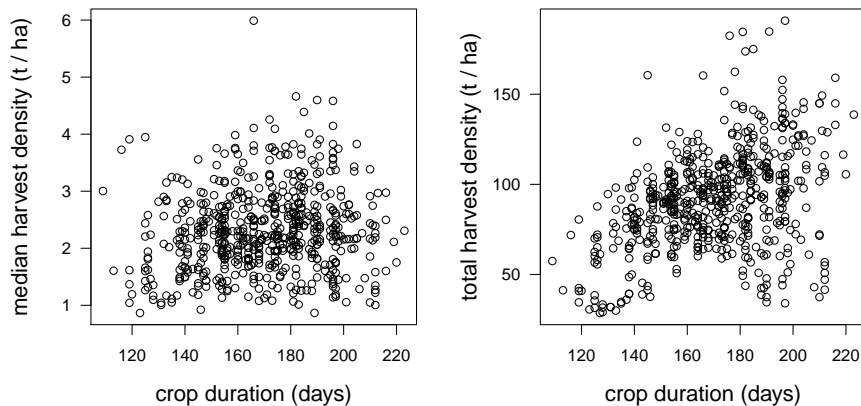


Figure 3.30: Harvest density versus the total duration of the training block crops.

tomato price will motivate the farmer to pick a greater proportion of the total yield. However, planting a crop towards the middle of the year also tends to result in a longer harvest period (see Figure 3.26 on page 86). The farmer picking tomatoes over a longer period due to favourable economic conditions will reap a larger cumulative harvest. Similarly, the farmer harvesting tomatoes over a longer period due to mid-year planting will also accumulate a higher total harvest density (see Figure 3.24 on page 85). Hence, irrespective of the reason for the longer harvest period, total harvest density should increase with the length of the harvest period.

In contrast to total harvest density, median harvest density shows a far weaker or no relationship with harvest period and crop duration (left-hand

plots in Figures 3.29 to 3.30 on the preceding page). There is, however, no reason to expect median harvest density to increase by extending the harvest period. On the contrary, longer harvest periods may even lead to median harvest density decreasing, since the larger harvest densities obtained early in the harvest season during the crop's most productive period will be offset by numerous smaller harvest densities towards the end of an extended harvest period. Indeed, Figure 3.31 shows that the number of harvest events (an alternative measure of harvest activity) has a positive relationship with total harvest density (right-hand plot), but a slightly negative relationship with median harvest density (left-hand plot).

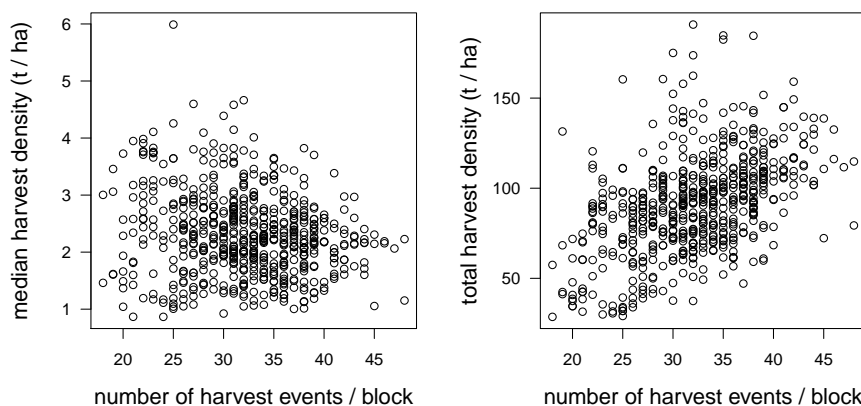


Figure 3.31: Harvest density versus the number of harvest events for the training block crops.

The relationship between harvest density and the length of the growing period is less clear (Figure 3.28 on page 87). This is probably because the time of year associated with the largest harvest densities—around week 30 (Figures 3.23 to 3.24 on page 85)—is associated with intermediate growing period lengths (Figure 3.25 on page 86).

3.1.6 Comparing crop variables

From the plots shown in Figures 3.17 to 3.31 on pages 82–89, it appears that median and total harvest density share slightly different relationships with some of the planting and harvest variables. Figure 3.32 on the next page

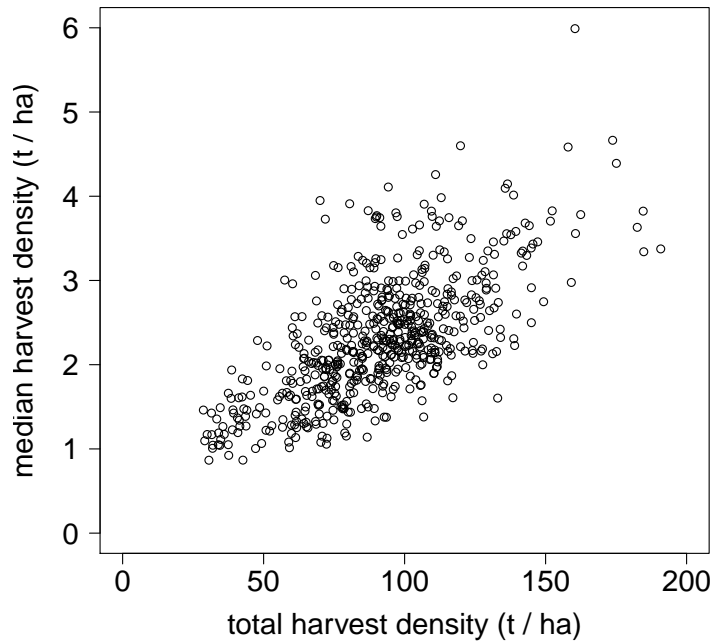


Figure 3.32: Scatterplot of median versus total harvest density in the training data set.

shows median and total harvest density plotted against each other. As expected, these two quantities are positively correlated with each other. There is, however, quite a lot of variance in this relationship, showing that all crops with higher total harvest densities do not necessarily have higher median harvest densities. The relationship between these two quantities is dependent on how the crop is harvested, as explained in the previous section.

Table 3.2: Correlation coefficients for the numerical crop variables in the training data set. Values greater than 0.5 or less than -0.5 are highlighted.

	TotTonPerHa	Ha	PIMonth	PIWeek	GrowD	NHarv	HarvD
MedTonPerHa	0.66	-0.20	-0.21	-0.21	0.29	-0.18	-0.09
TotTonPerHa		-0.12	-0.24	-0.25	0.18	0.49	0.37
Ha			0.23	0.22	-0.24	0.19	0.00
PIMonth				0.99	-0.82	-0.11	-0.38
PIWeek					-0.82	-0.12	-0.39
GrowD						-0.10	0.09
NHarv							0.69

Some of the planting and harvest variables measure similar quantities, and

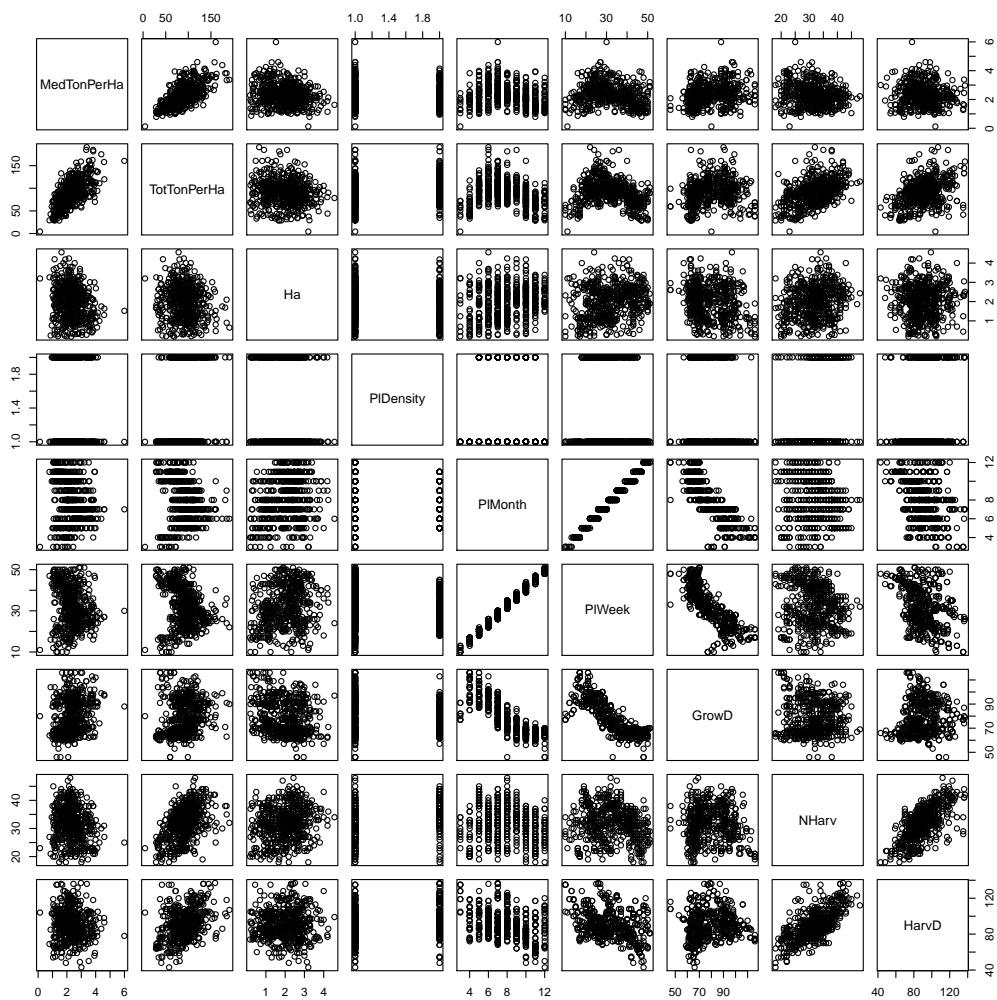


Figure 3.33: Scatterplots of median and total harvest density against some of the crop variables in the training data set.

can therefore be considered for exclusion. Since crop duration (`TotalD`) is simply the sum of the growing (`GrowD`) and harvest (`HarvD`) periods of a block crop, this variable is redundant and is therefore not included in the modelling analyses.

Figure 3.33 shows scatterplots of the variables from the crop data set, and Table 3.2 on the previous page shows the corresponding correlation coefficients for the numerical crop variables in the training data set.

It is evident from Figure 3.33 and Table 3.2 that some of the predictor variables are highly correlated, most obviously the quantitative variables month (`PlMonth`) and week (`PlWeek`) of planting ($r = 0.99$). `PlWeek` provides more

information regarding planting time than does `PlMonth`, and `PlMonth` was therefore omitted from the modelling analyses. The number of harvest events (`NHarv`) and the length of the harvest period (`HarvD`) constitute another group with a strong linear relationship ($r = 0.69$). Since `NHarv` is a more direct measure of “harvest effort” than is `HarvD`, `NHarv` was retained for subsequent analyses.

Although, `PlWeek` and `GrowD` also share a strong linear correlation ($r = -0.82$), the relationship between these two variables is nonlinear. Since all of the analyses in Chapters 4 to 6 involve a variable selection component (best subset selection was performed prior to MLR, and variable selection is an integral part of the lasso and the tree-based methods), both of these variables can be input into the analyses without jeopardising the stability of the resulting models.

3.2 The weather and evapotranspiration data set

3.2.1 Data set description and cleaning

Daily readings amounting to 2952 records were obtained from a weather station belonging to the South African Agricultural Research Council (ARC) close to the tomato farm for the period 2008/04/22–2016/04/05 (Figure 3.34 on the following page). The ARC also provided hourly readings from the same weather station for the period 2013/01/01–2016/02/29, which were used in the data-cleaning process. In addition to the weather data from the ARC, 294 ten-daily (or *dekad*) relative evapotranspiration (RE) values with a spatial resolution of 3 km spanning the period 2008/01/01–2016/02/21 were obtained from the Dutch company Environmental Analysis and Remote Sensing Earth Environment Monitoring B.V. (EARS-E2M). These RE values were estimated by EARS-E2M from weather variables obtained from satellite data for the GPS co-ordinates of the tomato farm. Hourly weather readings from a South African Weather Service (SAWS) weather station situated 12.7 km away in an adjacent valley to the tomato farm were also acquired. However, the closer proximity of the ARC weather station in the same valley as the tomato farm prompted the decision to use the ARC (rather than the SAWS) weather data



Figure 3.34: Relief map showing the location of the ARC weather station relative to the tomato farm. The ARC weather station is in the same valley, 51 m higher in altitude and 4.13 km away from the tomato farm. The map was created using GPS Visualizer (Schneider, 2016) using a map layer obtained from OpenCycleMap (Thunderforest, 2016).

to approximate the weather conditions on the tomato farm. The ARC and EARS-E2M weather variables are given in Table 3.3 on the next page.

The ARC weather variables contained some suspicious-looking values. Plots of the ARC daily weather time series (Figures 3.35a to 3.39a on pages 95–99) were therefore compared with time series plots of the ARC hourly and SAWS hourly data (not shown) in order to identify unrealistic readings for omission. Dates in the original ARC daily weather data set containing outlier values for one or more of the weather variables were entirely removed. The omission of all of the weather variables' readings for the dates containing any unrealistic values left a total of 2 816 records in the final (cleaned) data set (Figures 3.35b to 3.39b).

The RE variable from EARS-E2M (Figure 3.40 on page 100) contains no outlier values, and this variable was therefore used in its original form.

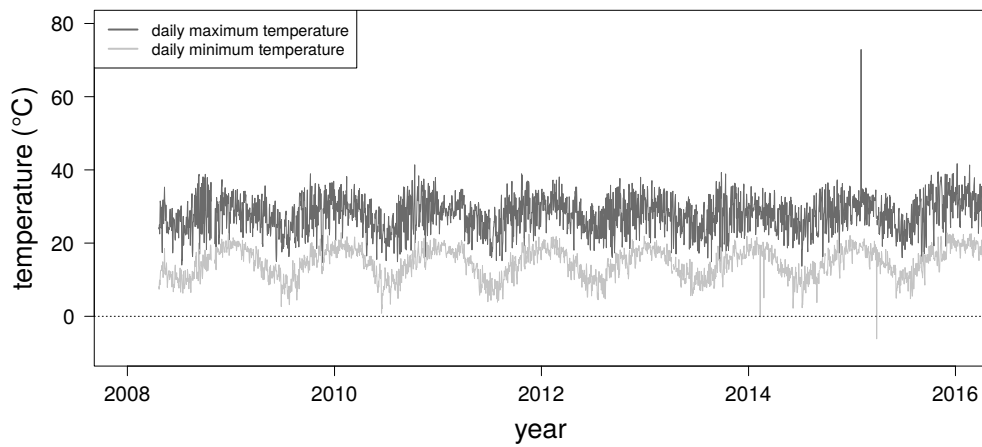
Table 3.3: Time series variables in the weather data set. The values of the variables (displayed in the third column) reflect the range of values in the cleaned weather data set. Refer to Figures 3.35a to 3.39a on pages 95–99 for an idea of the value ranges in the original weather data.

variable	description	type (and values)	units
Weather time series provided by the ARC (2008/04/22–2016/04/05):			
maxTemp	daily maximum temperature	real number; 13.68–41.70	°C
minTemp	daily minimum temperature	real number; 0.84–22.86	°C
maxRH	daily maximum relative humidity	real number; 24.40–100	%
minRH	daily minimum relative humidity	real number; 0.32–89.48	%
avgWindS	daily average wind speed	real number; 0.19–3.13	m/s
totalRain	daily total rainfall	real number; 0–82.04	mm/d
totalET0	daily total ET ₀	real number; 0–8.74	mm/d
Weather time series provided by EARS-E2M (2008/01/01–2016/02/21):			
ReLET	dekad (every 10 days) RE	integer; 0–100	%

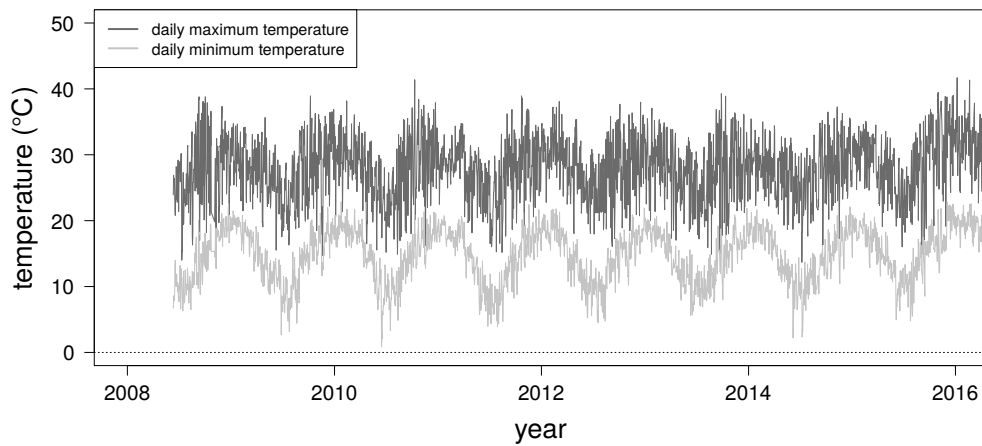
3.2.2 Variables in the weather data set

The time series plots in Figures 3.35b to 3.39b on pages 95–99 and Figure 3.40 on page 100 show that the weather variables follow annual cycles in response to the changing seasons. To get a better idea of the weather that the tomato crops are typically exposed to on the farm during the course of a year, the data values in the weather time series were plotted against calendar week. The resulting plots are shown in Figures 3.41 to 3.42 on pages 101–102.

Figures 3.41 to 3.42 show that the tomato farm vicinity is characterised by cold, dry winters and hot, wet summers. The daily total rainfall measurements over the period 2008/04/22–2016/04/05 recorded by the ARC weather station range from 0 mm/d to 82.04 mm/d (Table 3.3), or 0 mm/h to 3.42 mm/h, which can be described as “light” to “moderate” rain (Met Office, 2011). Similar to rainfall, RH (Figure 3.41 c–d) and evapotranspiration (ET; Figure 3.42 c–d) both fall during the winter months and rise in the summer. The moist air from the tropics and Indian Ocean that reaches this area in the summer (Tyson and Preston-Whyte, 2000) probably causes both the higher rainfall and higher humidity levels. Having said this, daily total rainfall has a symmetric distribution, reaching its minimum at around week 27 (Figure 3.42b), whereas



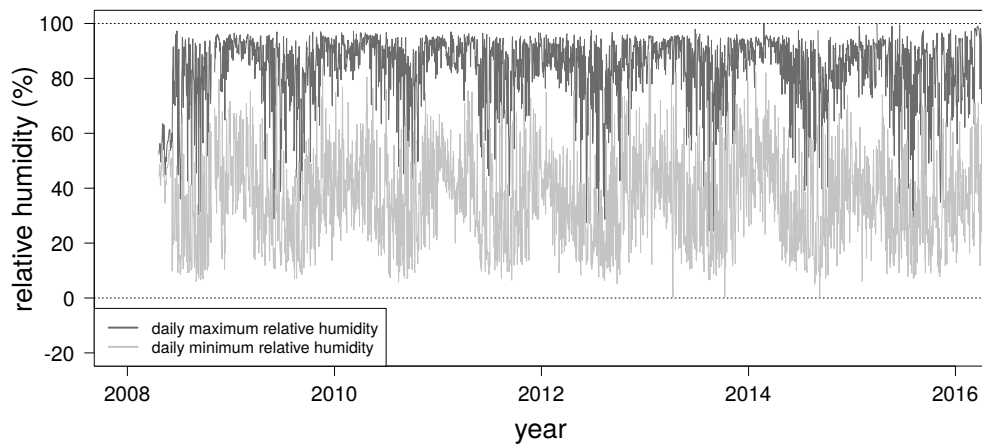
(a) original



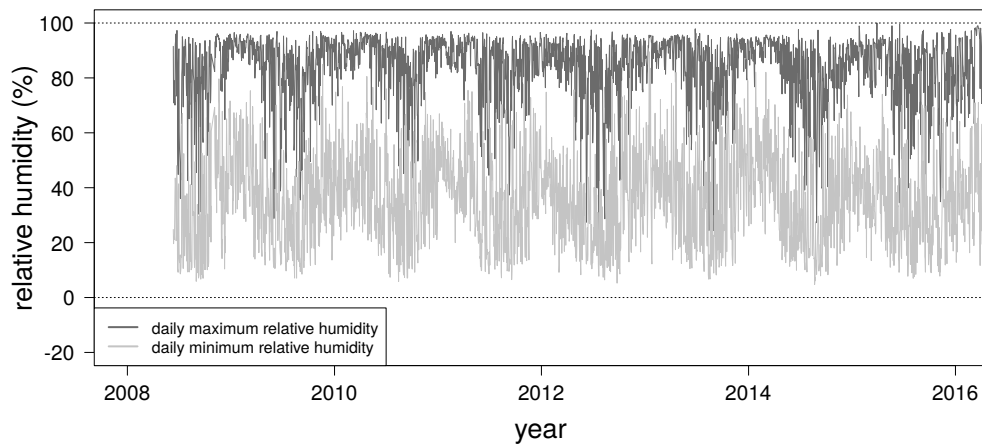
(b) cleaned (scaled)

Figure 3.35: (a) Original and (b) cleaned time series of the daily maximum and minimum temperatures recorded between 2008 and 2015 by the ARC's weather station. Dotted reference lines mark 0°C .

daily maximum and minimum RH have less symmetric distributions, each reaching their minima around week 35 (Figure 3.41 c–d). Consequently, despite their similar patterns overall, there appears to be a few weeks lag between rainfall and RH. Hence, humidity and rainfall patterns are probably mostly driven by the same processes, although there are also individual influences on each variable. The lower solar radiation levels (not shown), temperature, rainfall and RH in the winter probably result in less evaporation from the soil. These factors probably also contribute to slower metabolic rates (including those of photosynthesis and cellular respiration), resulting in less transpiration



(a) original

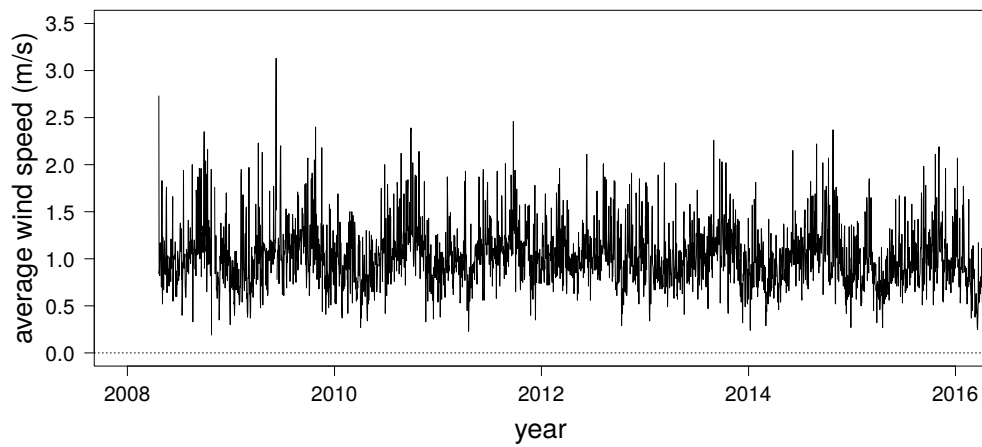


(b) cleaned (scaled)

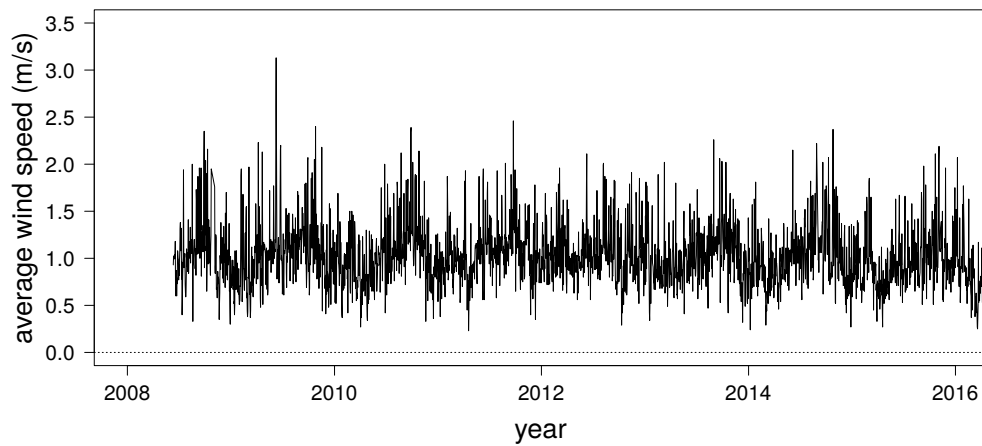
Figure 3.36: (a) Original and (b) cleaned time series of the daily maximum and minimum relative humidity values recorded between 2008 and 2015 by the ARC's weather station. Dotted reference lines mark 0% and 100%, respectively.

from the crop plants. Hence, ET_0 and RE also fall in the winter.

Figure 3.42a on page 102 shows that daily average wind speed also has an asymmetric distribution, peaking in the spring time after week 40 and reaching its minimum during the summer months. That being said, there is not much difference in wind speed across the seasons: The daily average wind speeds recorded between 2008/04/22–2016/04/05 by the ARC weather station range from 0.19 m/s to 3.13 m/s (Table 3.3 on page 94). According to the Beaufort Wind Force Scale (Met Office, 2010; Royal Meteorological Society, 2016)—a



(a) original



(b) cleaned (scaled)

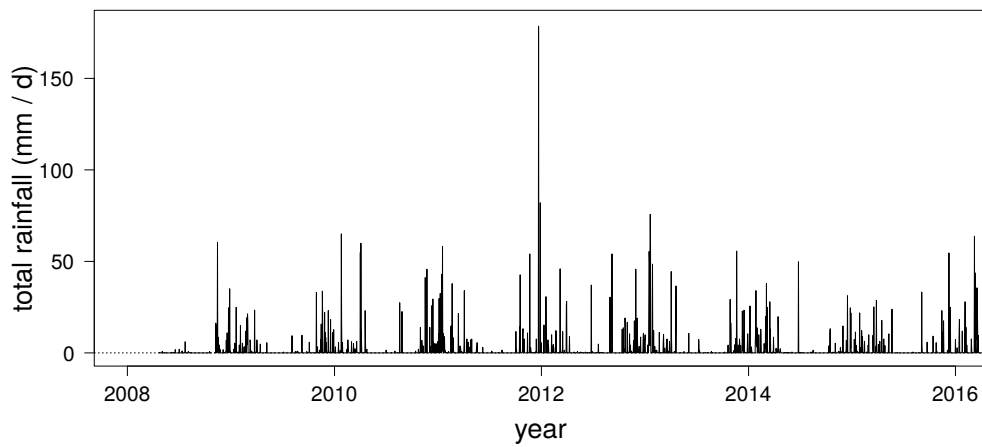
Figure 3.37: (a) Original and (b) cleaned time series of the daily average wind speed values recorded between 2008 and 2015 by the ARC’s weather station. Dotted reference lines mark 0 m/s.

popular system for relating wind speed to observed conditions—daily average wind speeds in the tomato farm vicinity range in force from “calm” (Beaufort number 0: < 0.3 m/s) to a “light breeze” (Beaufort number 2: 1.6 m/s to 3.3 m/s), categories that can, respectively, be described as

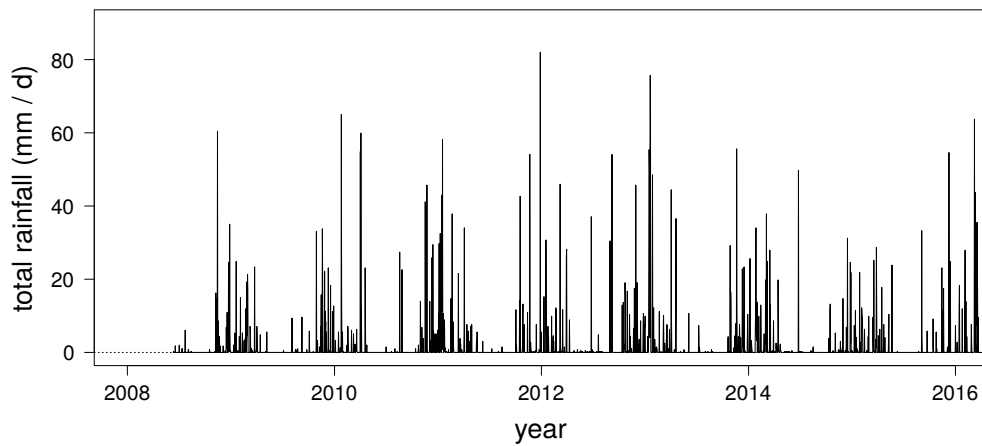
“Smoke rises vertically.”

and

“Wind felt on face; leaves rustle; wind vane moved by wind.”



(a) original

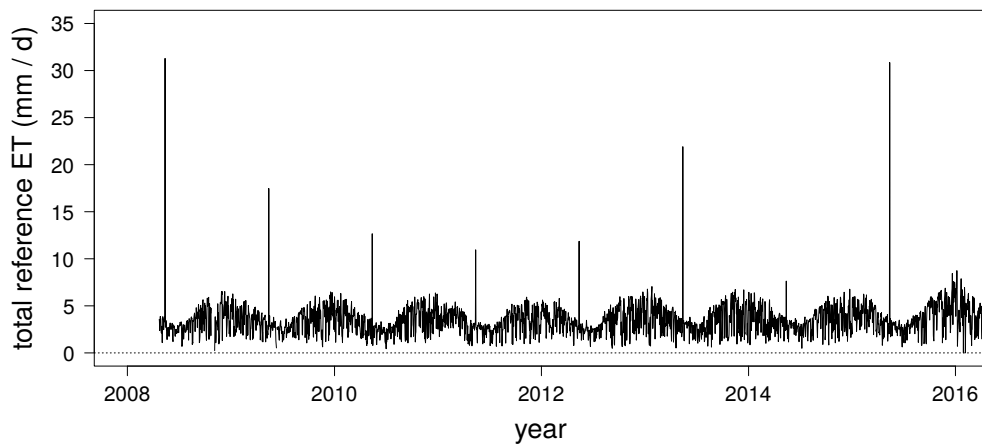


(b) cleaned (scaled)

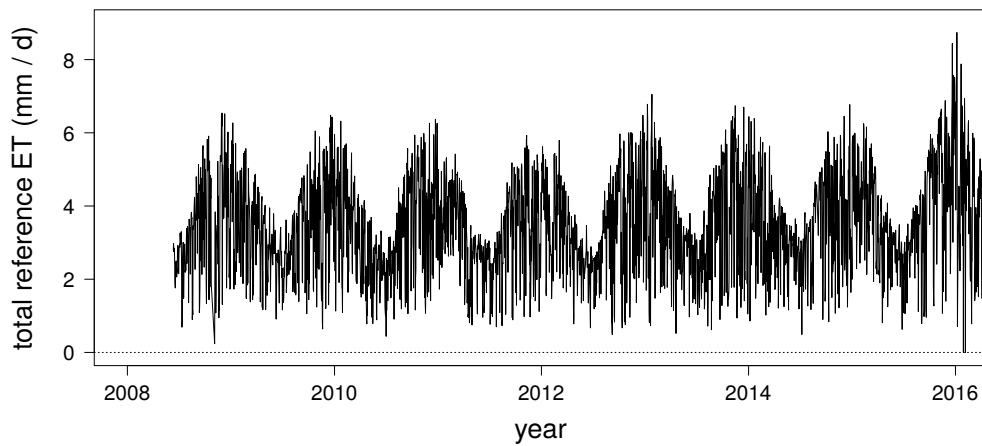
Figure 3.38: (a) Original and (b) cleaned time series of the daily total rainfall values recorded between 2008 and 2015 by the ARC's weather station. Dotted reference lines mark 0 mm.

Consequently, daily average wind speeds do not rise above a “light breeze”, and the farm's tomato crops probably do not suffer from wind damage very often.

Although the weather variables are in the form of time series, the statistical learning theory (SLT) techniques applied in this study only accept cross-sectional (or numerical) input variables. The weather variables therefore had to be summarised before being used for modelling harvest densities. To this end, the minimum and/or maximum, as well as the average, median and total of the time series values recorded during the growing period (i.e. from



(a) original



(b) cleaned (scaled)

Figure 3.39: (a) Original and (b) cleaned time series of the daily total reference evapotranspiration values recorded between 2008 and 2015 by the ARC's weather station. Dotted reference lines mark 0 mm.

planting to the first harvest event) of each block crop in the training data set were computed for each weather variable. Note that because the weather data comes from a single source (the weather data set), block crops sharing the same growing period (the same planting date and first harvest date) will share identical values for the summary statistics of their weather variables. Very few weather records were excluded from the ARC weather data set during the data-cleaning process (about 95% of dates for which weather readings were provided were retained; see Section 3.2.1 on page 92), and these summary statistics were therefore calculated from daily weather records representing

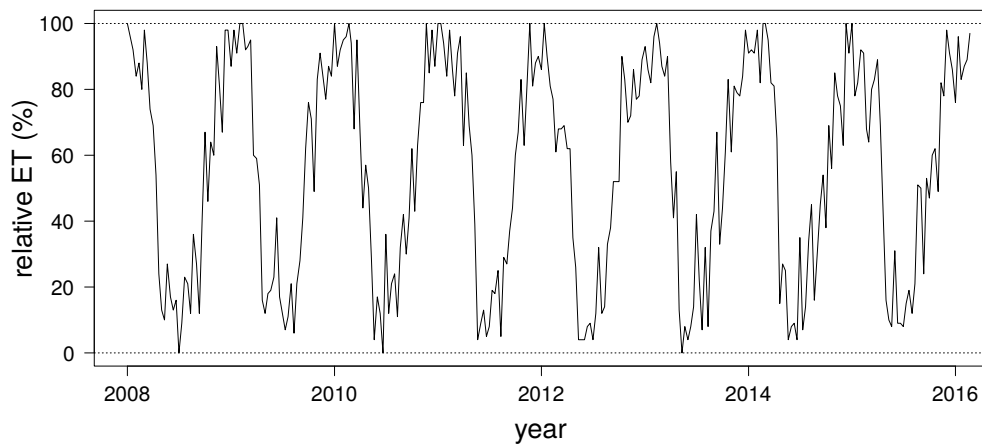
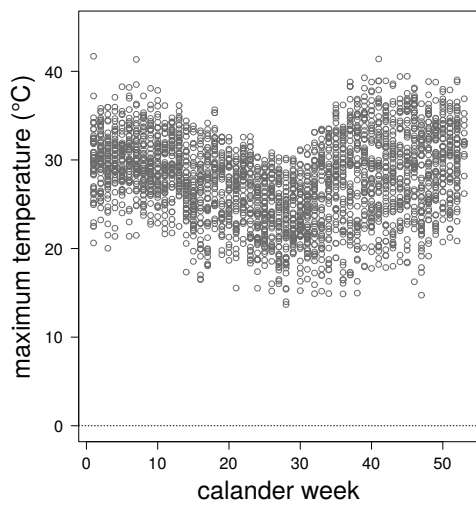


Figure 3.40: Time series of the dekad relative evapotranspiration values estimated from satellite data recorded between 2008 and 2015 by EARS-E2M for the tomato farm’s GPS co-ordinates. Dotted reference lines mark 0 % and 100 %, respectively.

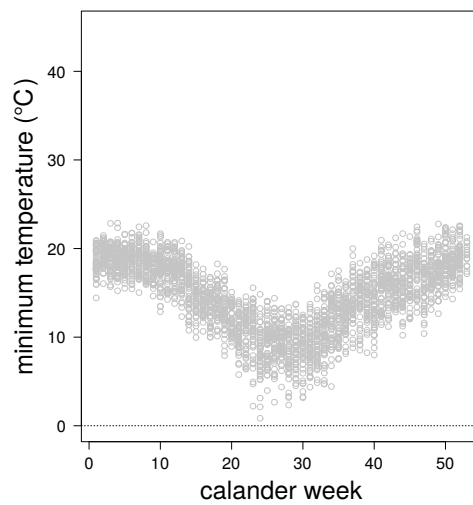
well over 80 % of the days in each block crop’s growing period. For each of the weather variables, the summary statistic having the strongest linear correlations with the two response variables median and total harvest density based on the training data was chosen as the input for that weather variable in the subsequent modelling analyses.

Distributions and scatterplots of the maximum, average, median and total summary statistics for the daily maximum temperature and RH are displayed in Figure 3.43 on page 103 and Figure 3.45 on page 105, respectively, while distributions and scatterplots of the minimum, average, median and total for the daily minimum temperature and RH in the training data set are shown in Figure 3.44 on page 104 and Figure 3.46 on page 106. Distributions and scatterplots of the maximum, minimum, average, median and total for daily average wind speed, daily total rainfall, daily total ET_0 and dekad RE for the block crops are shown in Figures 3.47 to 3.50 on pages 107–110. The corresponding correlation coefficients between the summary statistics of each weather variable and median and total harvest density are presented in Tables 3.4 to 3.11 on pages 111–113.

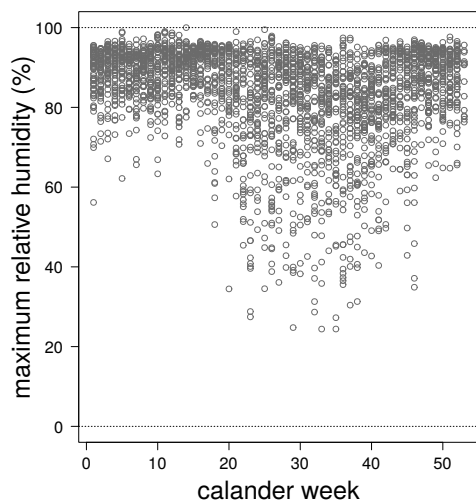
For most of the weather variables, the correlations of the averages and medians of the weather readings with the two response variables for the block crops are very similar to each other (see Figures 3.43 to 3.47 on pages 103–107,



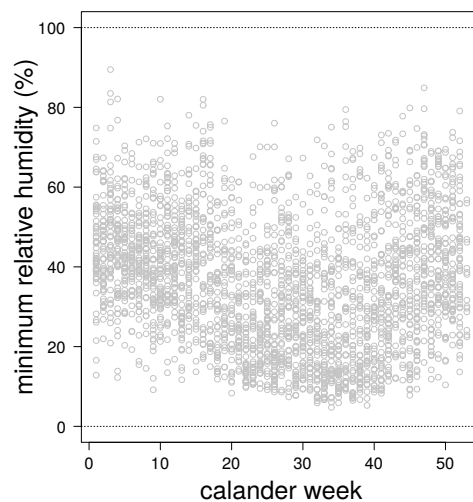
(a) daily maximum temperature



(b) daily minimum temperature



(c) daily maximum relative humidity



(d) daily minimum relative humidity

Figure 3.41: Daily maximum and minimum temperature and relative humidity throughout the year. Dotted reference lines mark 0 °C (top figures) and 0 % and 100 % (bottom figures).

Figures 3.49 to 3.50 on pages 109–110, Tables 3.4 to 3.8 on pages 111–112 and Tables 3.10 to 3.11 on page 113), suggesting that there are not many outlier values perturbing the averages. However, rainfall is the exception to this rule: Since it does not rain on most days, even in the summer, both the minimum and median rainfall values for the vast majority of the block crops is 0 mm/d (Figure 3.48 on page 108). The absence of variation in the minimum and median rainfall levels over the growing periods of the block crops rules these

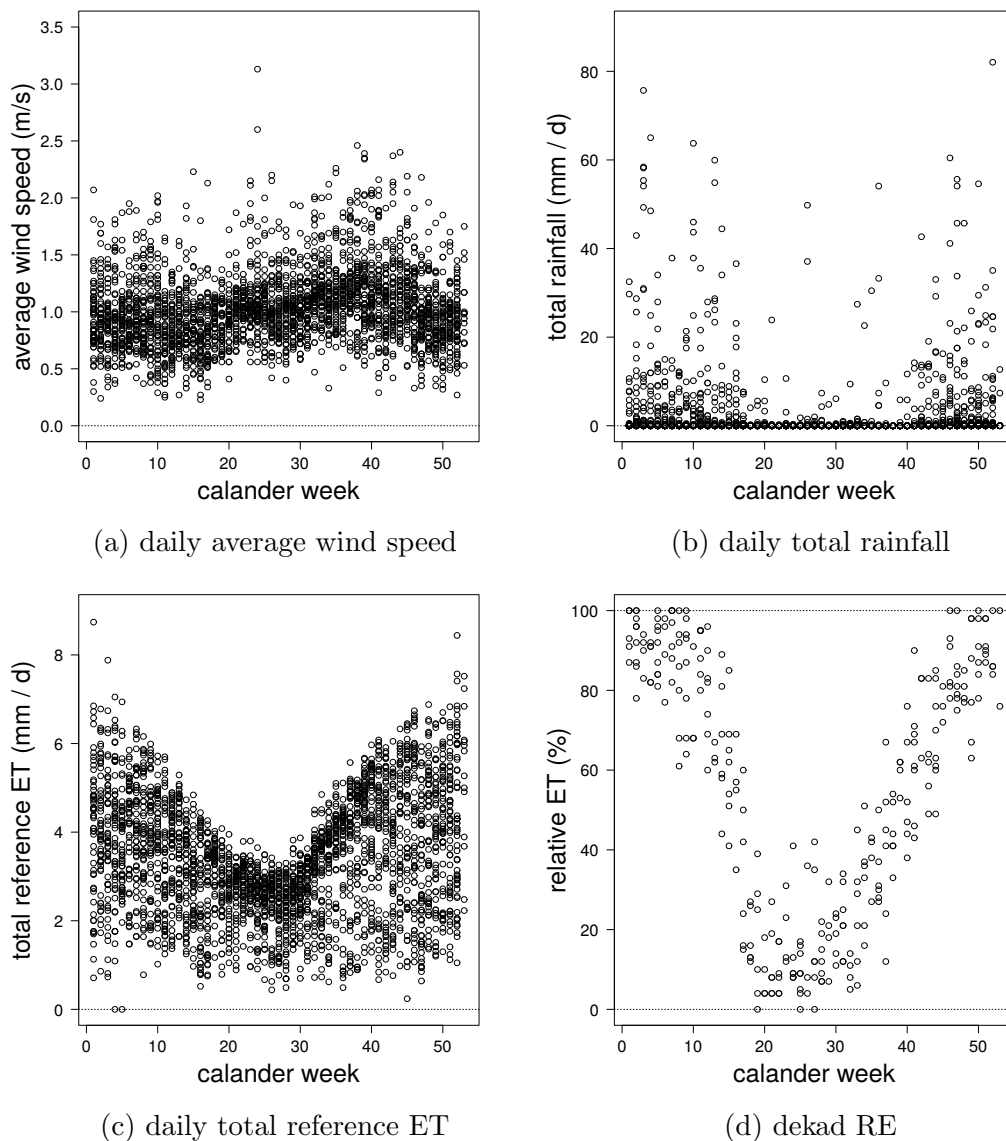


Figure 3.42: Daily average wind speed, daily total rainfall, daily total reference evapotranspiration and dekad relative evapotranspiration throughout the year. Dotted reference lines mark (a) 0 m/s, (b) 0 mm/d, (c) 0 mm/d and (d) 0% and 100%. ET: evapotranspiration; RE: relative evapotranspiration.

two summary statistics out as informative inputs of rainfall for the statistical models. A comparison of maximum, average and total rainfall over the block crops' growing periods reveals the average rainfall levels as being the most strongly correlated with the two responses, closely followed by total rainfall (Table 3.9 on page 112). The average was therefore chosen as the summary statistic for daily total rainfall.

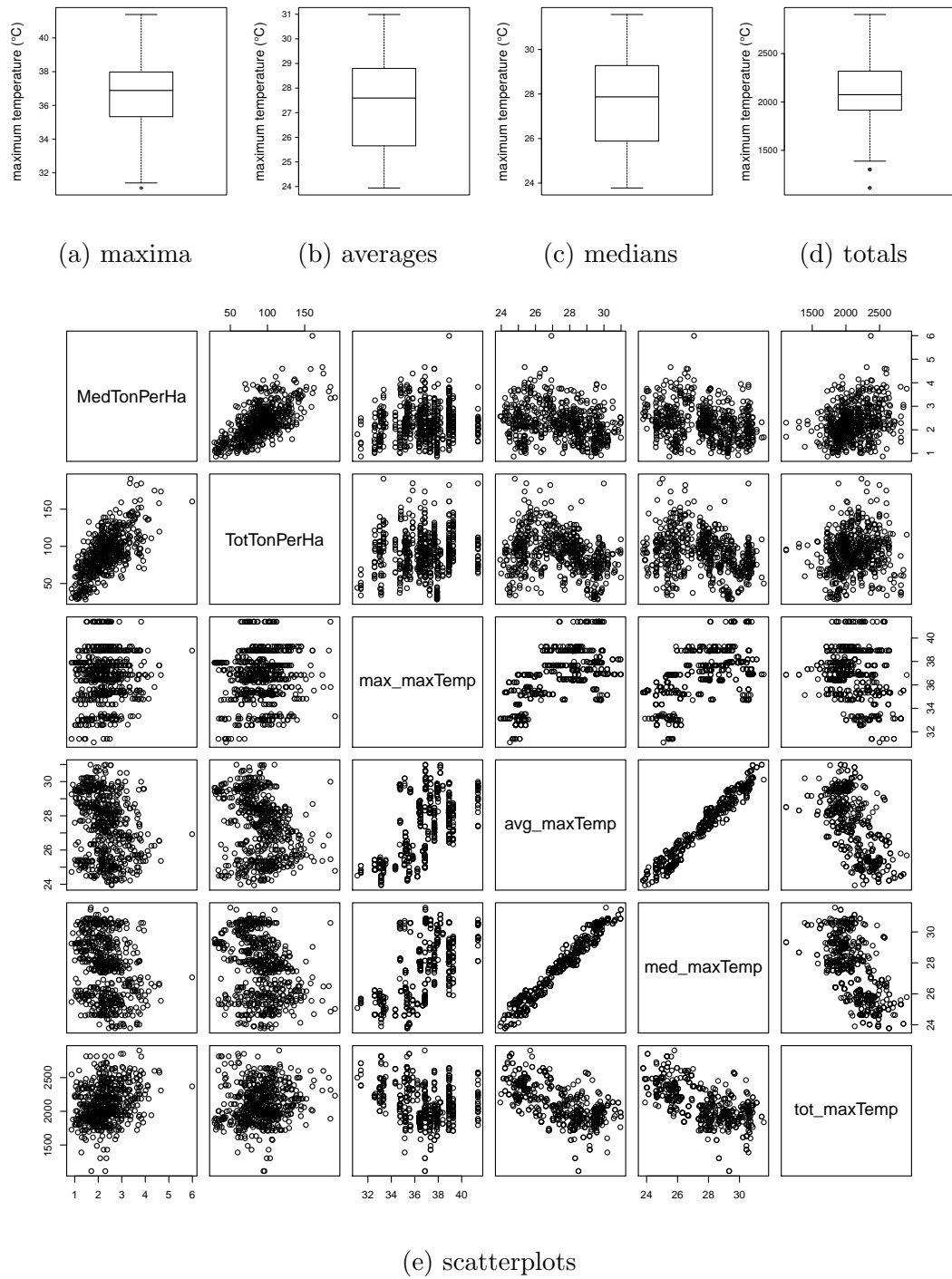


Figure 3.43: The maximum, average, median and total summary statistics of the daily maximum temperatures experienced by the training block crops.

Although the correlations of the averages and medians of the weather readings taken during the growing periods of the block crops with the two responses

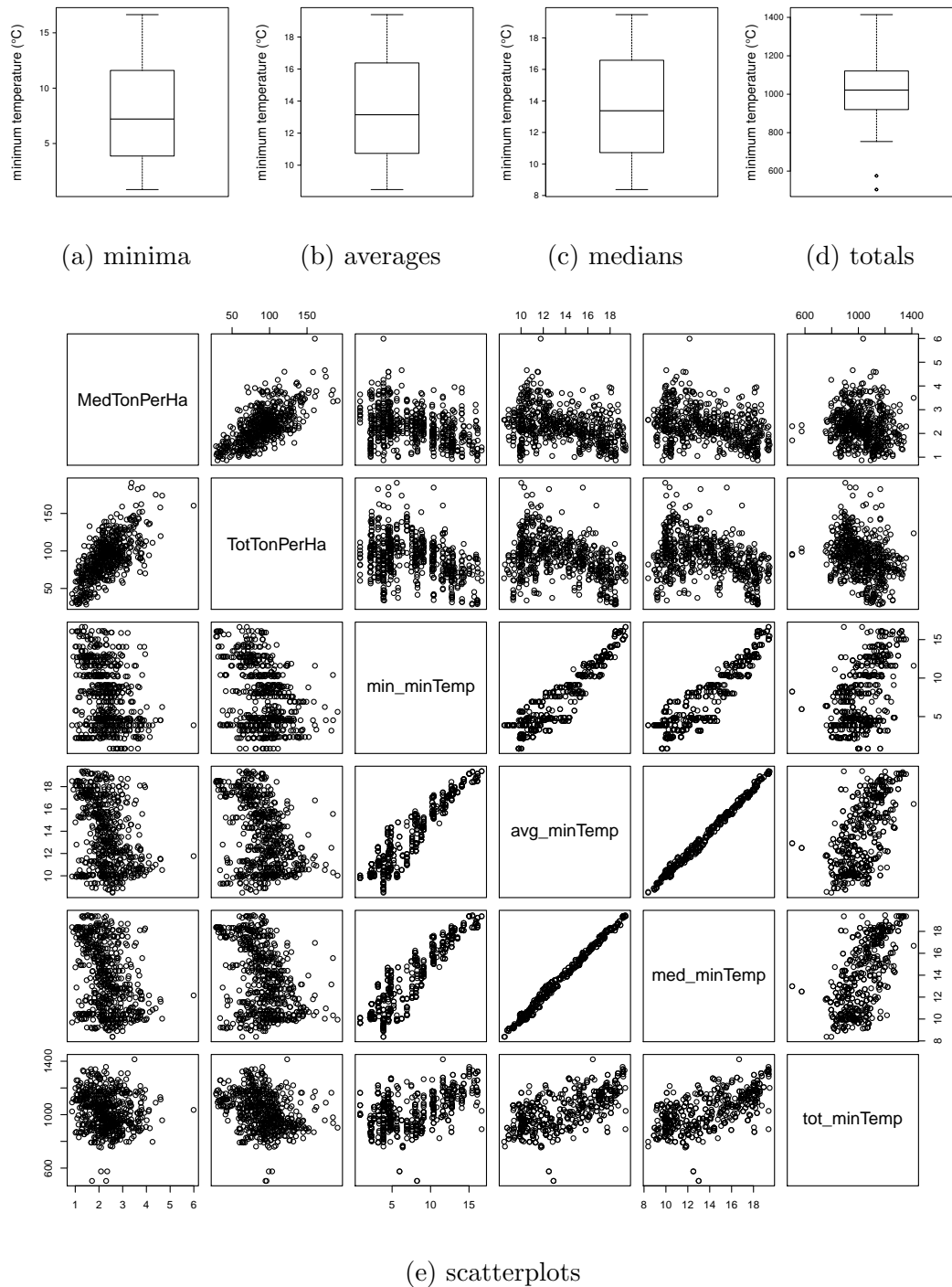


Figure 3.44: The minimum, average, median and total summary statistics of the daily minimum temperatures experienced by the training block crops.

in the training data set are very similar for all of the weather variables except rainfall, the median is slightly more strongly correlated with the two responses

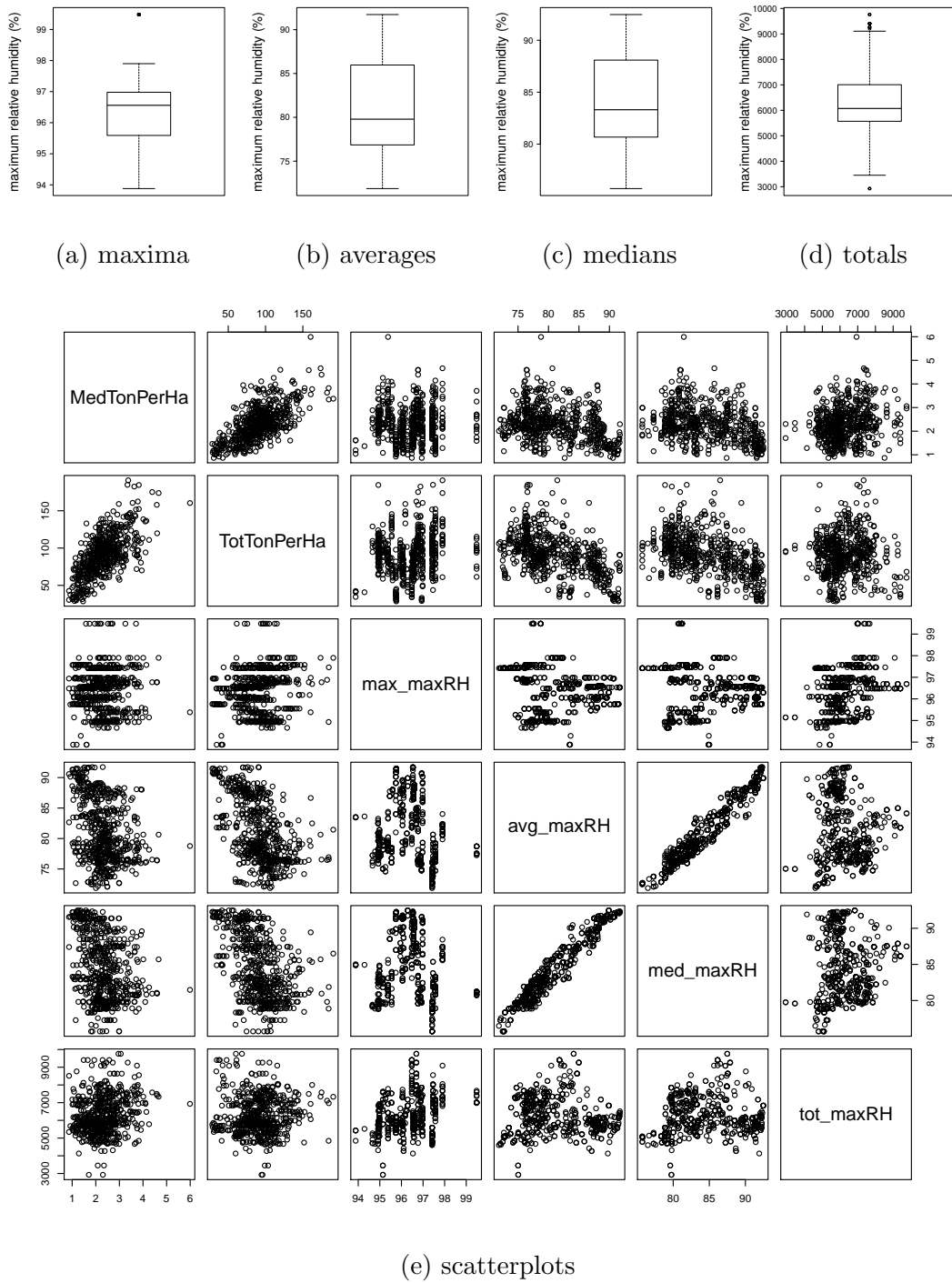


Figure 3.45: The maximum, average, median and total summary statistics of the maximum relative humidity experienced by the training block crops.

in the case of daily maximum temperature (Figure 3.43 on page 103 and Table 3.4 on page 111), while the average tends to have slightly higher corre-

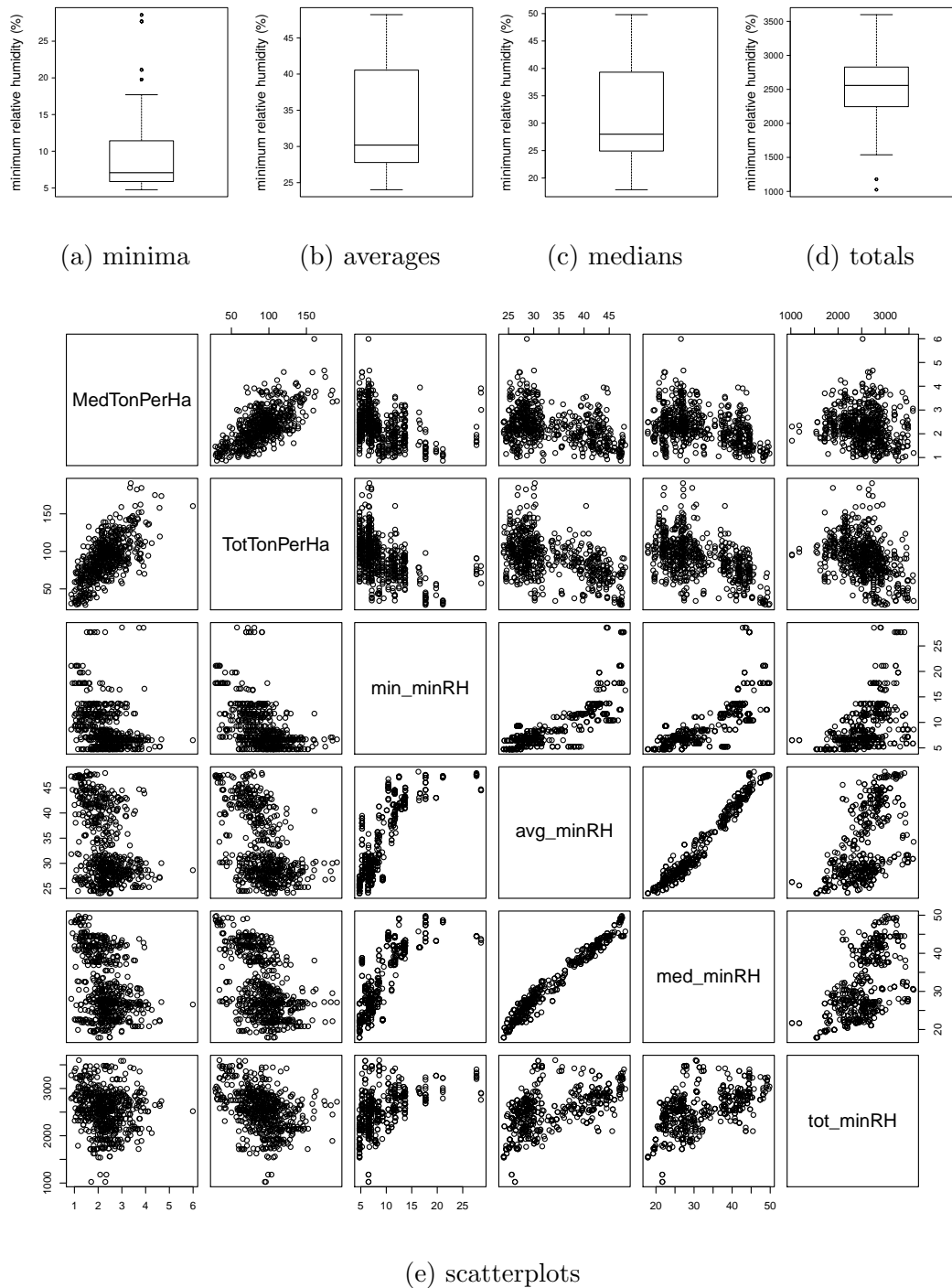
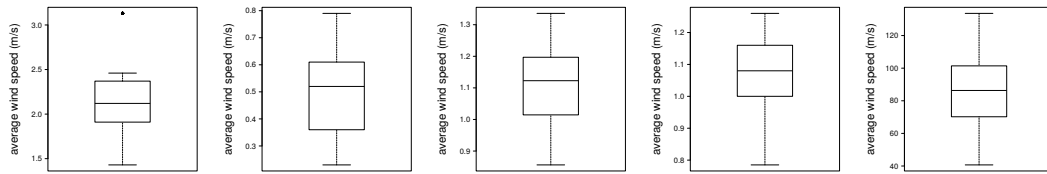
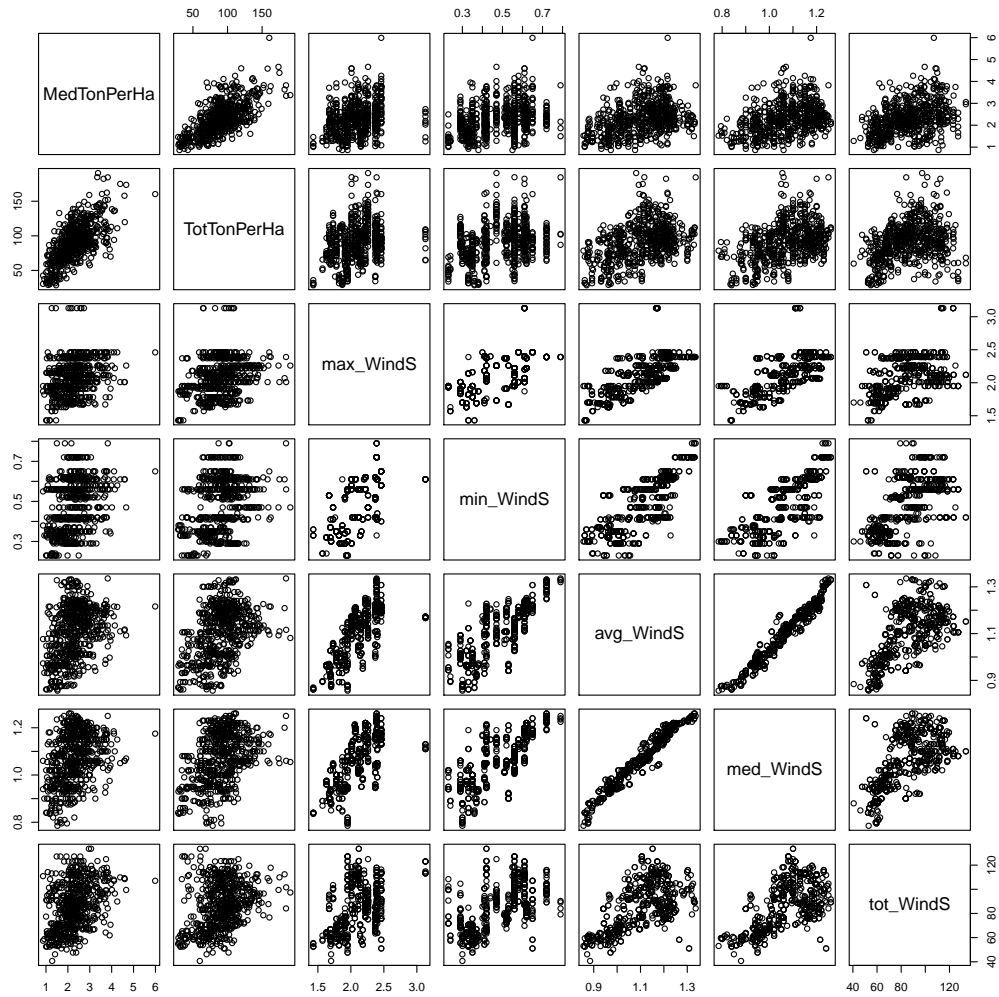


Figure 3.46: The minimum, average, median and total summary statistics of the minimum relative humidity experienced by the training block crops.

lations with the responses in the case of daily maximum and minimum RH and daily average wind speed (Figures 3.45 to 3.47 on pages 105–107 and Ta-



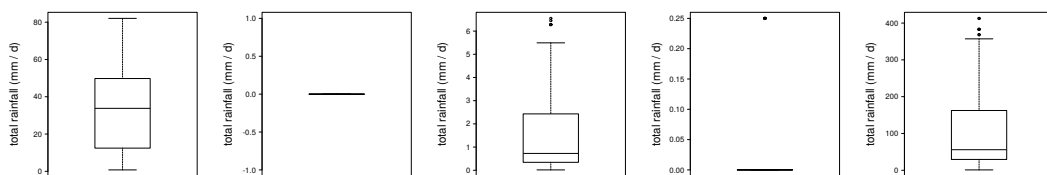
(a) maxima (b) minima (c) averages (d) medians (e) totals



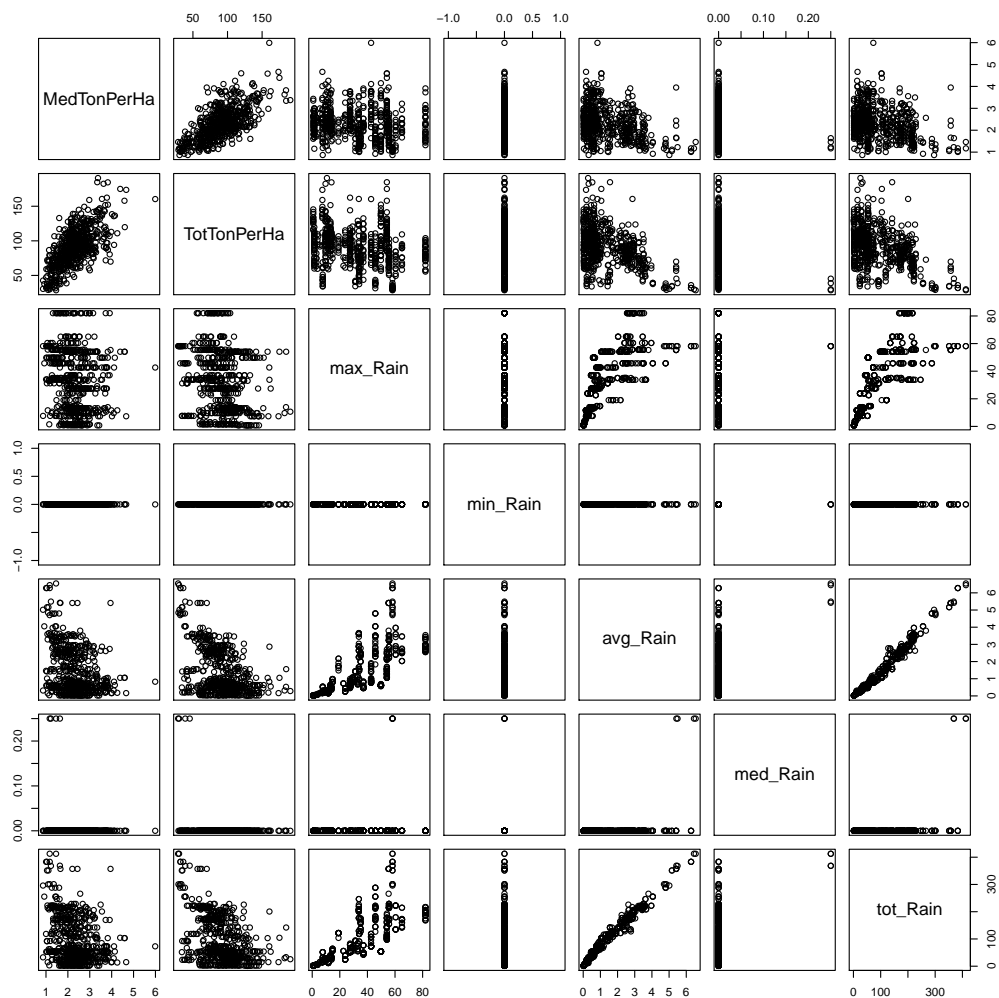
(f) scatterplots

Figure 3.47: The maximum, minimum, average, median and total summary statistics of the daily average wind speed experienced by the training block crops.

bles 3.6 to 3.8 on pages 111–112). The average and median daily minimum temperatures have identical correlations with the two responses (Figure 3.44 on page 104 and Table 3.5 on page 111), and the average was arbitrarily cho-



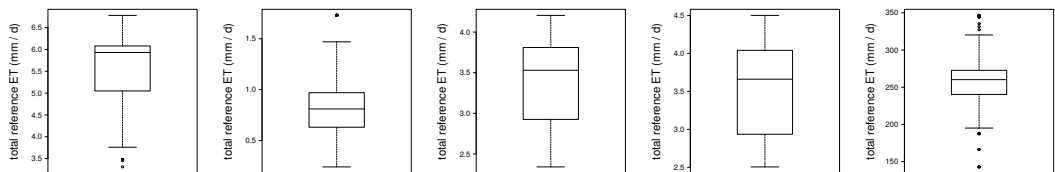
(a) maxima (b) minima (c) averages (d) medians (e) totals



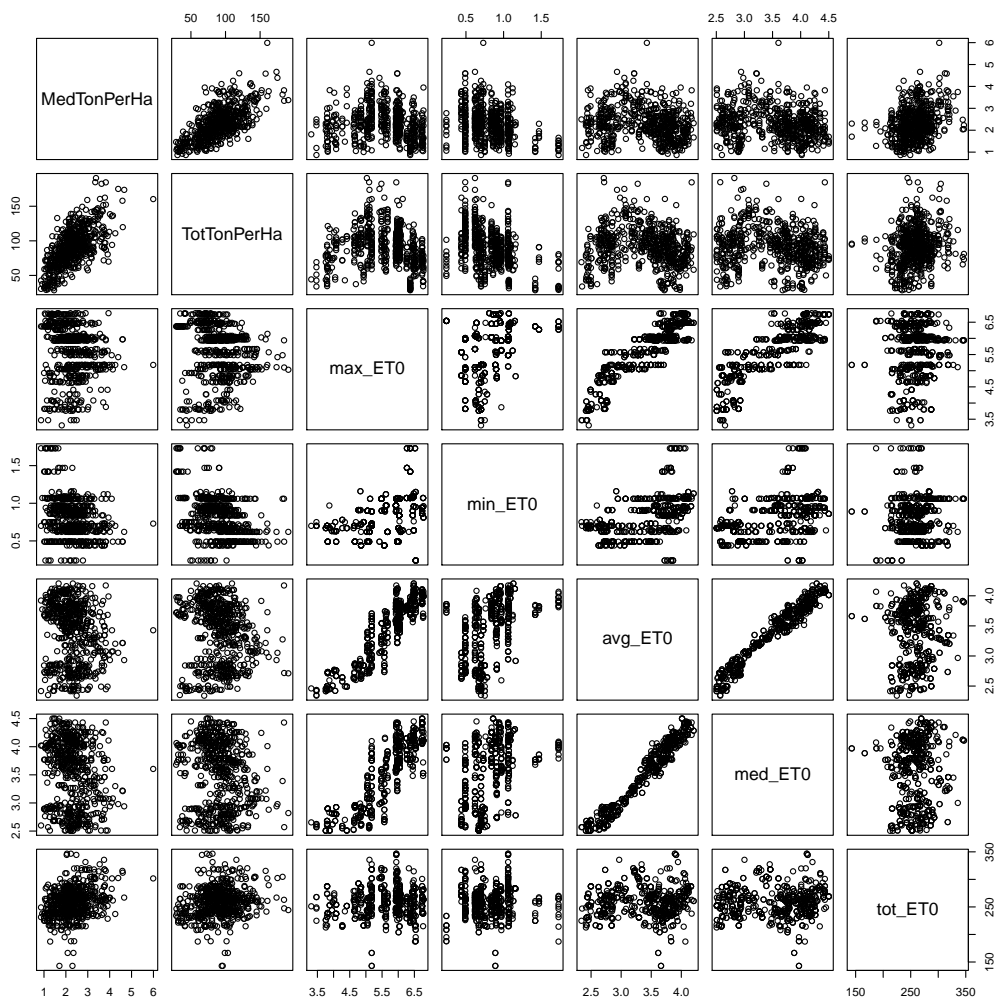
(f) scatterplots

Figure 3.48: The maximum, minimum, average, median and total summary statistics of the daily total rainfall experienced by the training block crops.

sen for this variable. Finally, in the case of the two ET weather variables, the minimum (rather than one of the two central tendency measures) of the readings over the block crops' growing periods tends to be the most strongly correlated with the two responses (Figures 3.49 to 3.50 on pages 109–110 and



(a) maxima (b) minima (c) averages (d) medians (e) totals

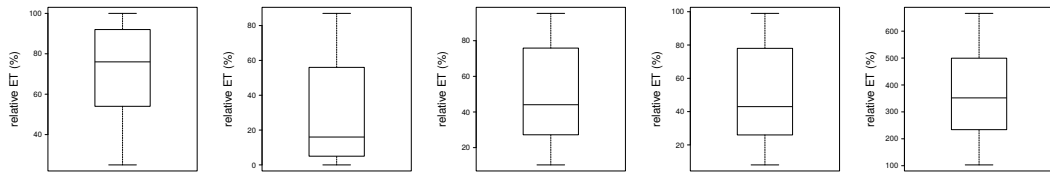


(f) scatterplots

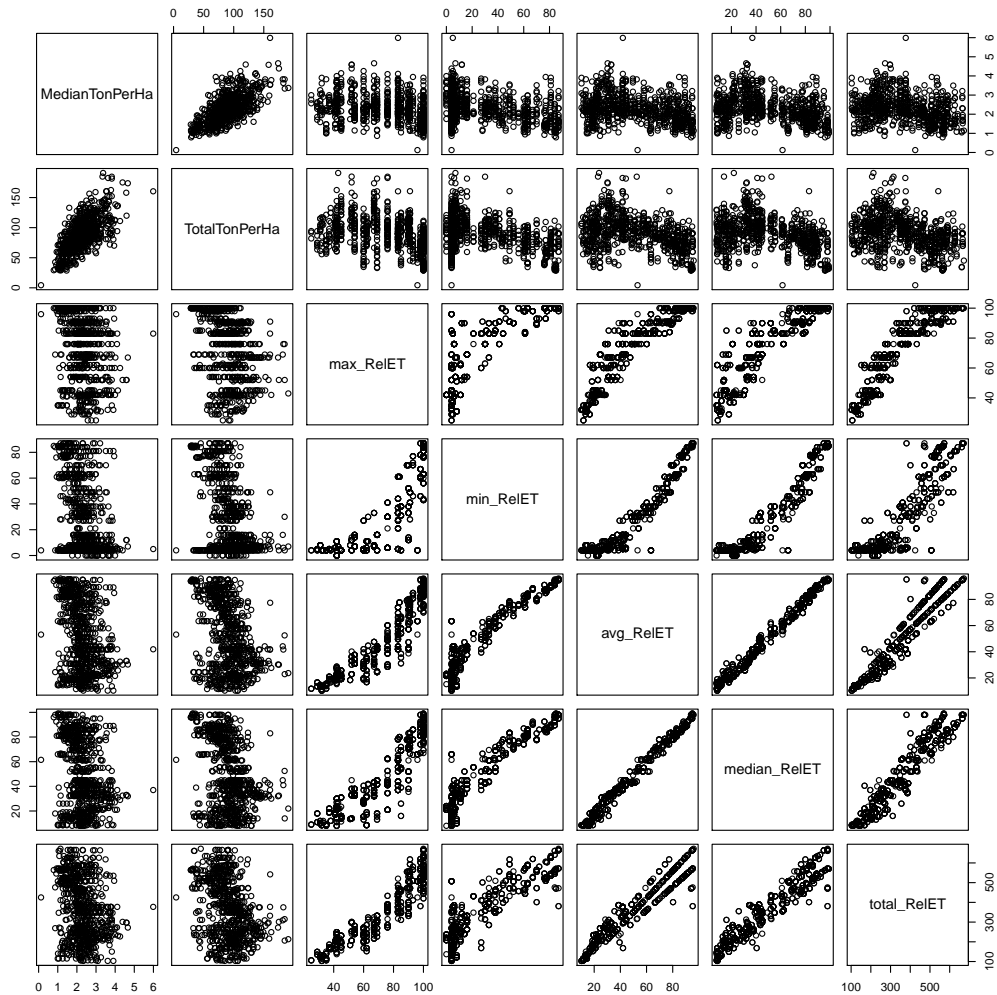
Figure 3.49: The maximum, minimum, average, median and total summary statistics of the daily total reference evapotranspiration experienced by the training block crops.

Tables 3.10 to 3.11 on page 113).

Hence, the median of the daily maximum temperatures, the averages of the daily minimum temperature, the daily maximum and minimum RH values, the



(a) maxima (b) minima (c) averages (d) medians (e) totals



(f) scatterplots

Figure 3.50: The maximum, minimum, average, median and total summary statistics of the dekad relative evapotranspiration experienced by the training block crops.

daily average wind speed and the daily total rainfall and the minima of the daily total ET_0 and dekad RE were selected as the summary statistics for these weather variables.

Table 3.4: Correlation coefficients for the daily maximum temperature summary statistics and harvest density in the training data set. The rows containing the correlations of the summary statistics with harvest density are light grey, and the summary statistics having the highest correlations with harvest density are dark grey.

	max_maxTemp	avg_maxTemp	med_maxTemp	tot_maxTemp
MedTonPerHa	0.09	-0.26	-0.27	0.28
TotTonPerHa	0.08	-0.28	-0.28	0.13
max_maxTemp		0.62	0.59	-0.27
avg_maxTemp			0.98	-0.62
med_maxTemp				-0.62

Table 3.5: Correlation coefficients for the daily minimum temperature summary statistics and harvest density in the training data set. The two harvest density rows are light grey, and the summary statistics most highly correlated with harvest density are dark grey.

	min_minTemp	avg_minTemp	med_minTemp	tot_minTemp
MedTonPerHa	-0.33	-0.33	-0.33	-0.13
TotTonPerHa	-0.35	-0.38	-0.38	-0.31
min_minTemp		0.94	0.92	0.53
avg_minTemp			1.00	0.62
med_minTemp				0.63

Table 3.6: Correlation coefficients for the daily maximum relative humidity summary statistics and harvest density in the training data set. The two harvest density rows are light grey, and the summary statistics most highly correlated with harvest density are dark grey.

	max_maxRH	avg_maxRH	med_maxRH	tot_maxRH
MedTonPerHa	0.01	-0.33	-0.32	0.20
TotTonPerHa	0.18	-0.51	-0.48	0.02
max_maxRH		-0.18	-0.10	0.27
avg_maxRH			0.97	-0.05
med_maxRH				0.06

The relationships between the selected summary statistic of each weather variable and median and total harvest density in the training data are shown in Figures 3.51 to 3.58 on pages 114–117. The weather variable summary statistics that appear to have the strongest *linear* relationships with median

Table 3.7: Correlation coefficients for the daily minimum relative humidity summary statistics and harvest density in the training data set. The two harvest density rows are light grey, and the summary statistics most highly correlated with harvest density are dark grey.

	min_minRH	avg_minRH	med_minRH	tot_minRH
MedTonPerHa	-0.33	-0.38	-0.37	-0.13
TotTonPerHa	-0.49	-0.51	-0.51	-0.38
min_minRH		0.82	0.79	0.49
avg_minRH			0.98	0.55
med_minRH				0.55

Table 3.8: Correlation coefficients for the daily average wind speed summary statistics and harvest density in the training data set. The two harvest density rows are light grey, and the summary statistics most highly correlated with harvest density are dark grey.

	max_WindS	min_WindS	avg_WindS	med_WindS	tot_WindS
MedTonPerHa	0.25	0.31	0.36	0.35	0.39
TotTonPerHa	0.28	0.31	0.41	0.40	0.31
max_WindS		0.61	0.73	0.68	0.44
min_WindS			0.79	0.75	0.59
avg_WindS				0.98	0.59
med_WindS					0.58

Table 3.9: Correlation coefficients for the daily total rainfall summary statistics and harvest density in the training data set. The two harvest density rows are light grey, and the summary statistics most highly correlated with harvest density are dark grey.

	max_Rain	avg_Rain	tot_Rain
MedTonPerHa	-0.12	-0.34	-0.30
TotTonPerHa	-0.22	-0.47	-0.44
max_Rain		0.72	0.75
avg_Rain			0.99

(MedTonPerHa) and total (TotTonPerHa) harvest density are the averages of the daily maximum RH ($r(\text{avg_maxRH}, \text{MedTonPerHa}) = -0.33$; $r(\text{avg_maxRH}, \text{TotTonPerHa}) = -0.51$; Table 3.6 on the preceding page; Figure 3.53 on page 115) and daily minimum RH ($r(\text{avg_minRH}, \text{MedTonPerHa}) = -0.38$; $r(\text{avg_minRH}, \text{TotTonPerHa}) = -0.51$; Table 3.7; Figure 3.54 on page 115) readings.

Table 3.10: Correlation coefficients for the daily total reference evapotranspiration summary statistics and harvest density in the training data set. The two harvest density rows are light grey, and the summary statistics most highly correlated with harvest density are dark grey.

	max_ET0	min_ET0	avg_ET0	med_ET0	tot_ET0
MedTonPerHa	-0.13	-0.30	-0.16	-0.18	0.31
TotTonPerHa	-0.10	-0.37	-0.13	-0.13	0.19
max_ET0		0.39	0.90	0.85	0.09
min_ET0			0.48	0.42	0.04
avg_ET0				0.98	0.05
med_ET0					0.05

Table 3.11: Correlation coefficients for the dekad relative evapotranspiration summary statistics and harvest density in the training data set. The two harvest density rows are light grey, and the summary statistics most highly correlated with harvest density are dark grey.

	max_ReET	min_ReET	avg_ReET	med_ReET	tot_ReET
MedTonPerHa	-0.26	-0.34	-0.31	-0.30	-0.26
TotTonPerHa	-0.34	-0.43	-0.39	-0.37	-0.35
max_ReET		0.80	0.92	0.90	0.93
min_ReET			0.95	0.94	0.88
avg_ReET				0.99	0.96
med_ReET					0.95

The positive linear relationship between harvest density and daily average wind speed may seem counterintuitive at first, in that it appears that higher wind speeds are conducive to higher harvest densities. However, Figure 3.11c on page 78, Figures 3.23 to 3.24 on page 85 and Figure 3.42a on page 102 together provide an alternative explanation for the patterns evident in Figure 3.55: Figures 3.23 to 3.24 show that harvest density is maximised on the farm by planting a tomato crop towards the middle of the year (around weeks 20 to 35). The tomato crops' growing periods range from 39 to 116 days (Figure 3.11c on page 78), suggesting that many of the crops planted around mid-year are still in their developmental phases during the windiest season of the year, which is around week 40 (Figure 3.42a). Hence, it could be that, rather than strong winds actually *inducing* high harvest densities, it is rather the case that the time of the year in which conditions are most conducive to high yields *happens to co-incide with* the windiest time of the year. In other

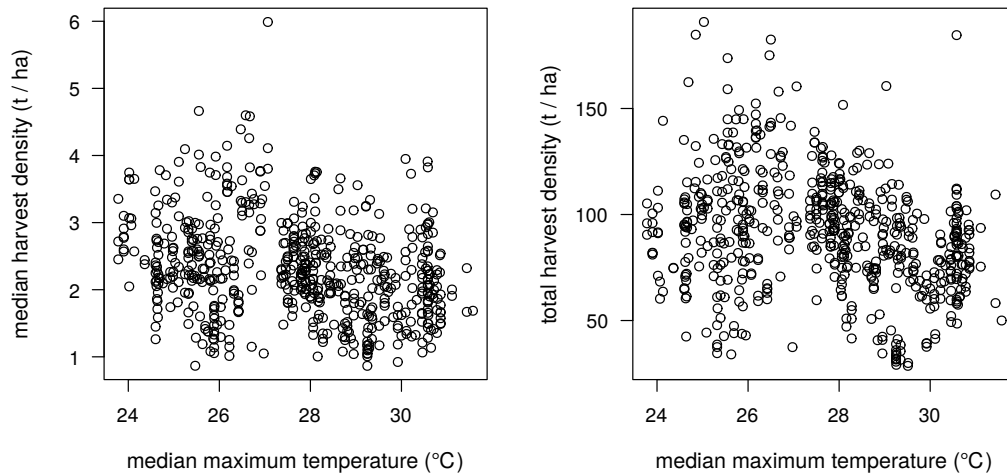


Figure 3.51: Harvest density versus the median of the daily maximum temperatures over the growing periods of the training block crops.

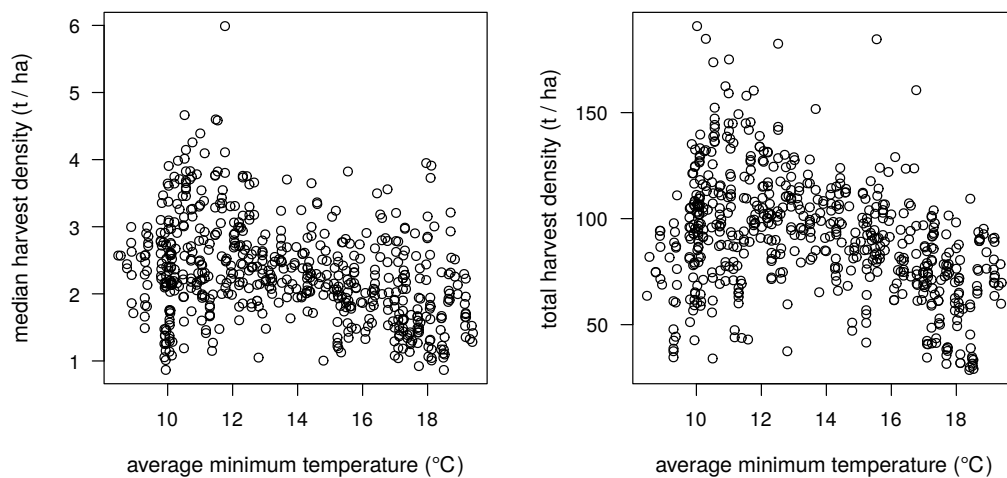


Figure 3.52: Harvest density versus the average of the daily minimum temperatures over the growing periods of the training block crops.

words, wind speed and harvest density probably share common driving forces that make these variables correlated with one another (rather than dependent on one another) within the range of daily average wind speeds present in the ARC data set. Since daily average wind speeds do not typically rise much above 3 m/s (Table 3.3 on page 94), which can be described as a “light breeze”

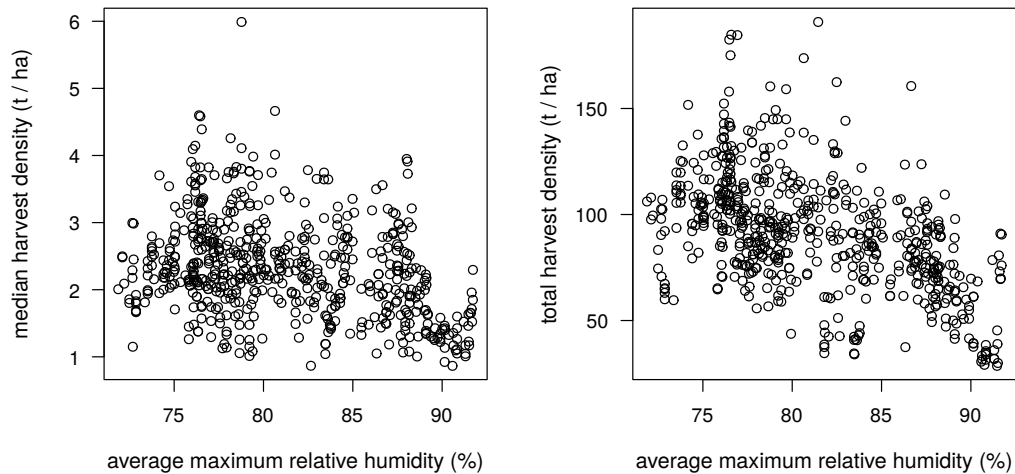


Figure 3.53: Harvest density versus the average of the daily maximum relative humidity readings over the growing periods of the training block crops.

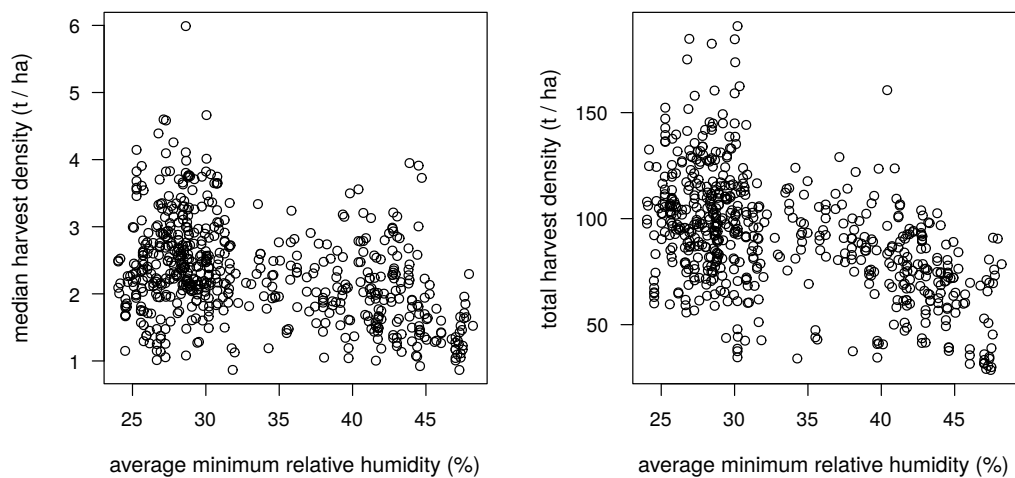


Figure 3.54: Harvest density versus the average of the daily minimum relative humidity readings over the growing periods of the training block crops.

(Met Office, 2010; Royal Meteorological Society, 2016), the positive correlation between daily average wind speed and harvest density is not tempered (much) by high wind speeds causing extensive crop damage. One would, however, not expect the positive linear relationship in Figure 3.55 to hold for wind speeds much higher than those recorded in the ARC weather data set.

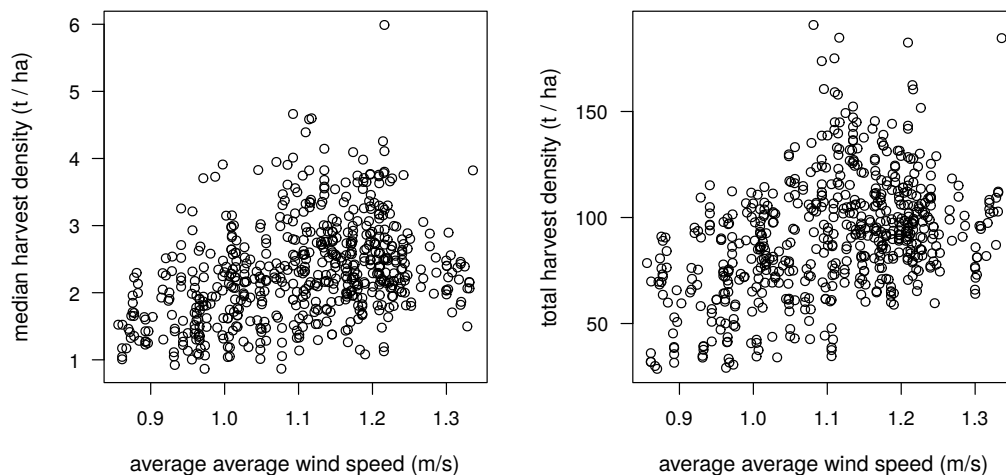


Figure 3.55: Harvest density versus the average of the daily average wind speed readings over the growing periods of the training block crops.

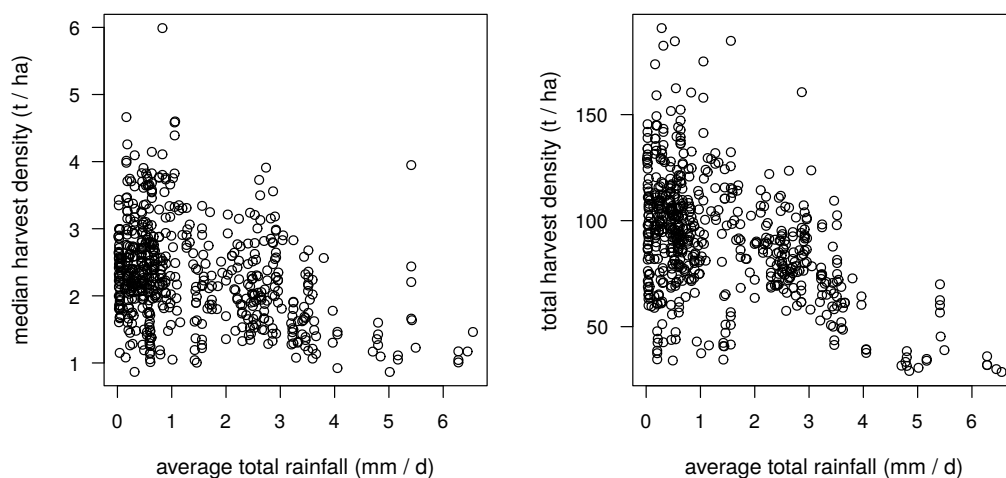


Figure 3.56: Harvest density versus the average of the daily total rainfall readings over the growing periods of the training block crops.

As is the case with daily average wind speed, daily maximum and minimum RH also have asymmetric distributions across the weeks of the year, each reaching their minima at around week 35 (rather than in the middle of the year; Figure 3.41 c–d on page 101). A comparison between Figures 3.23 to 3.24 on page 85 and Figure 3.41 c–d reveals that harvest density tends to

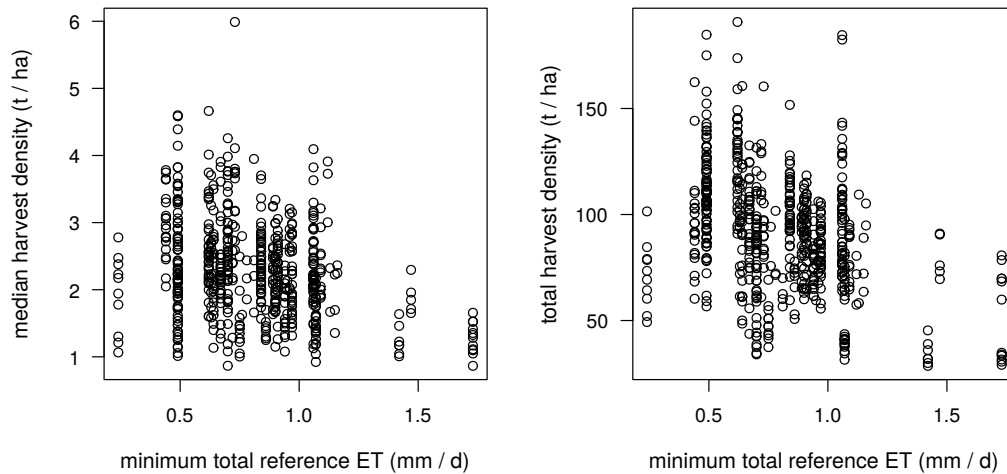


Figure 3.57: Harvest density versus the minimum of the daily total reference evapotranspiration readings over the growing periods of the training block crops.

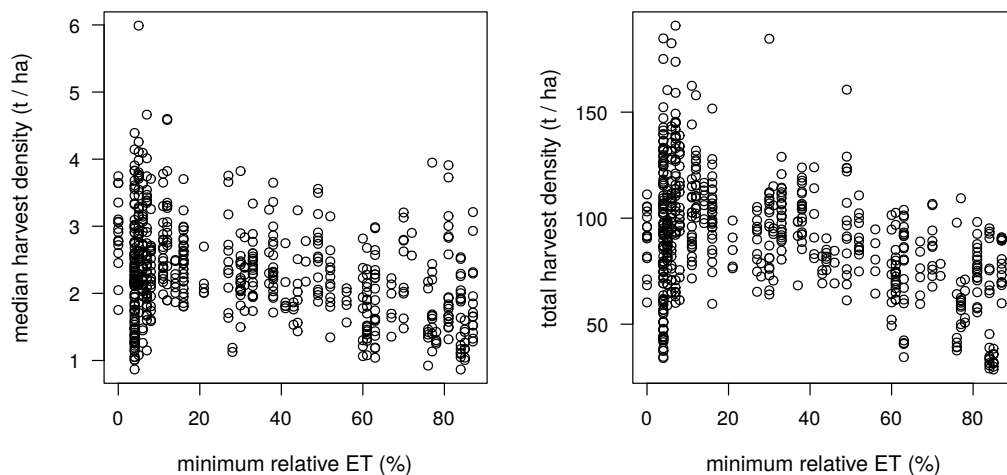


Figure 3.58: Harvest density versus the minimum of the dekad relative evapotranspiration readings over the growing periods of the training block crops.

increase as RH decreases and vice versa, explaining the negative relationships in Figures 3.53 to 3.54 on page 115.

The relationships between harvest density and the averages of the daily maximum RH, daily minimum RH and daily average wind speed readings are

relatively straightforward to interpret, since daily average wind speed reaches a maximum and daily maximum and minimum RH reach their minima during the time of the year when many of the most successful crops are in their growth phases. However, this is not the case for the other weather variables: Daily maximum and minimum temperature (Figure 3.41 a–b on page 101), daily total rainfall, daily total ET_0 and dekad RE (Figure 3.42 b–d on page 102) all peak in the summer and plummet in the winter, suggesting that intermediate levels of these variables should maximise harvest density. This indeed appears to be the case, since harvest density shares quadratic relationships with most of the weather variables in Figures 3.51 to 3.52 on page 114 and Figures 3.56 to 3.58 on pages 116–117.

Similarities amongst the weather variables, most evident in Figures 3.41 to 3.42 on pages 101–102, suggest high levels of correlations. In the next section, scatterplots and correlation coefficients of the weather variables are presented in order to identify groups of highly correlated variables.

3.2.3 Comparing the weather variables

Figure 3.59 on the next page displays scatterplots of the selected summary statistics of the weather variables against one another and the two responses, while Table 3.12 on page 122 provides the corresponding correlation coefficients.

The medians of the daily maximum temperatures (`med_maxTemp`) and the averages of the daily minimum temperatures (`avg_minTemp`) are highly correlated with each other ($r = 0.92$), as are the averages of the daily maximum RH readings (`avg_maxRH`) and of the daily minimum RH readings (`avg_minRH`; $r = 0.94$). Of the two temperature summary statistics, `avg_minTemp` is more strongly correlated with harvest density than is `med_maxTemp`, and `avg_minTemp` (rather than `med_maxTemp`) was therefore retained for the modelling analyses. Although `avg_minRH` is more strongly correlated with the two responses than is `avg_maxRH`, `avg_maxRH` has a more linear relationship with harvest density than does `avg_minRH` (compare Figure 3.53 on page 115 with Figure 3.54 on page 115). Since linear statistical modelling algorithms will be applied to the data in Chapter 4, `avg_maxRH` (rather than `avg_minRH`) was retained for the subsequent analyses.

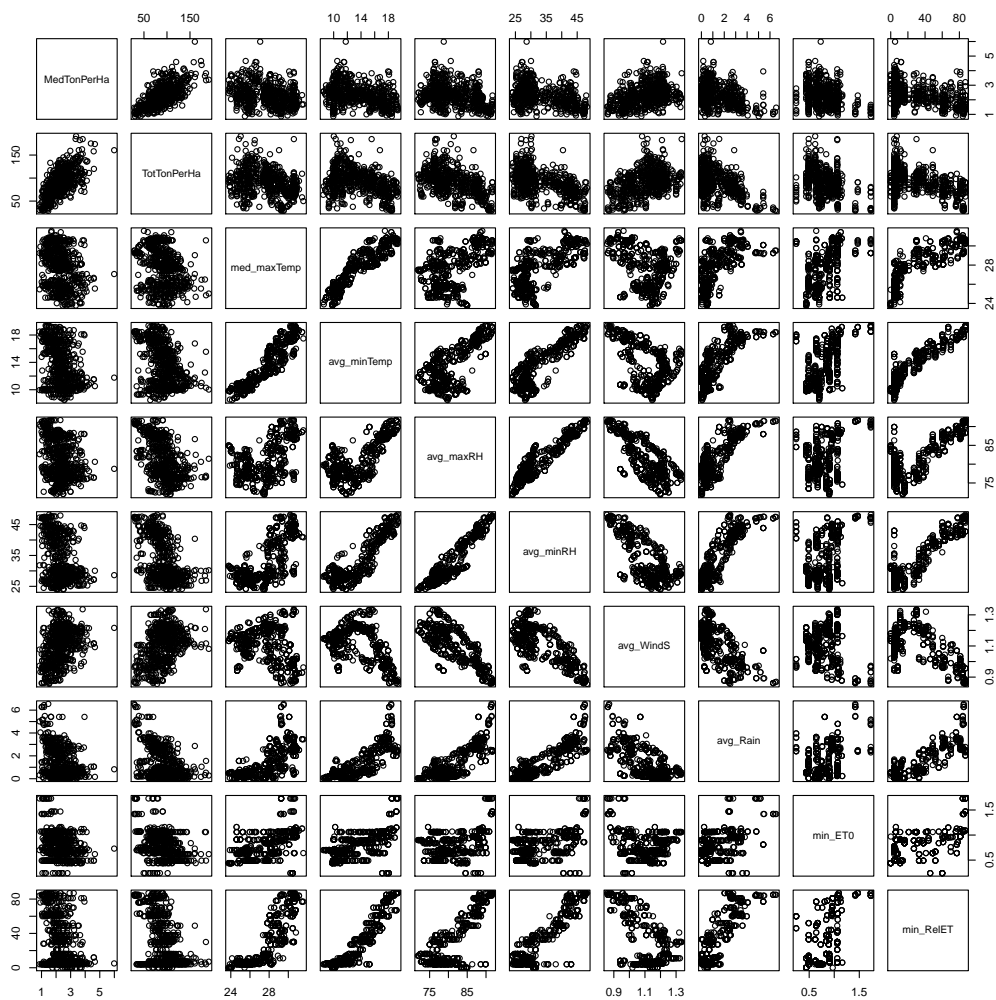


Figure 3.59: Scatterplots of the training set summary statistics of the ARC and EARS-E2M weather time series.

3.3 The crop and weather variables combined

3.3.1 Variables in the crop and weather data set

Excluding `med_maxTemp` and `avg_minRH` and adding the crop variables to the summary statistics of the weather time series in the training data set resulted in the scatterplots in Figure 3.60 on the next page and the correlation coefficients in Table 3.13 on page 123. Figure 3.60 and Table 3.13 show that there are still strongly correlated variables present in the crop and weather data set, which will be addressed by variable selection in subsequent chapters. Table 3.14 on page 124 presents the variables that will be used as inputs in

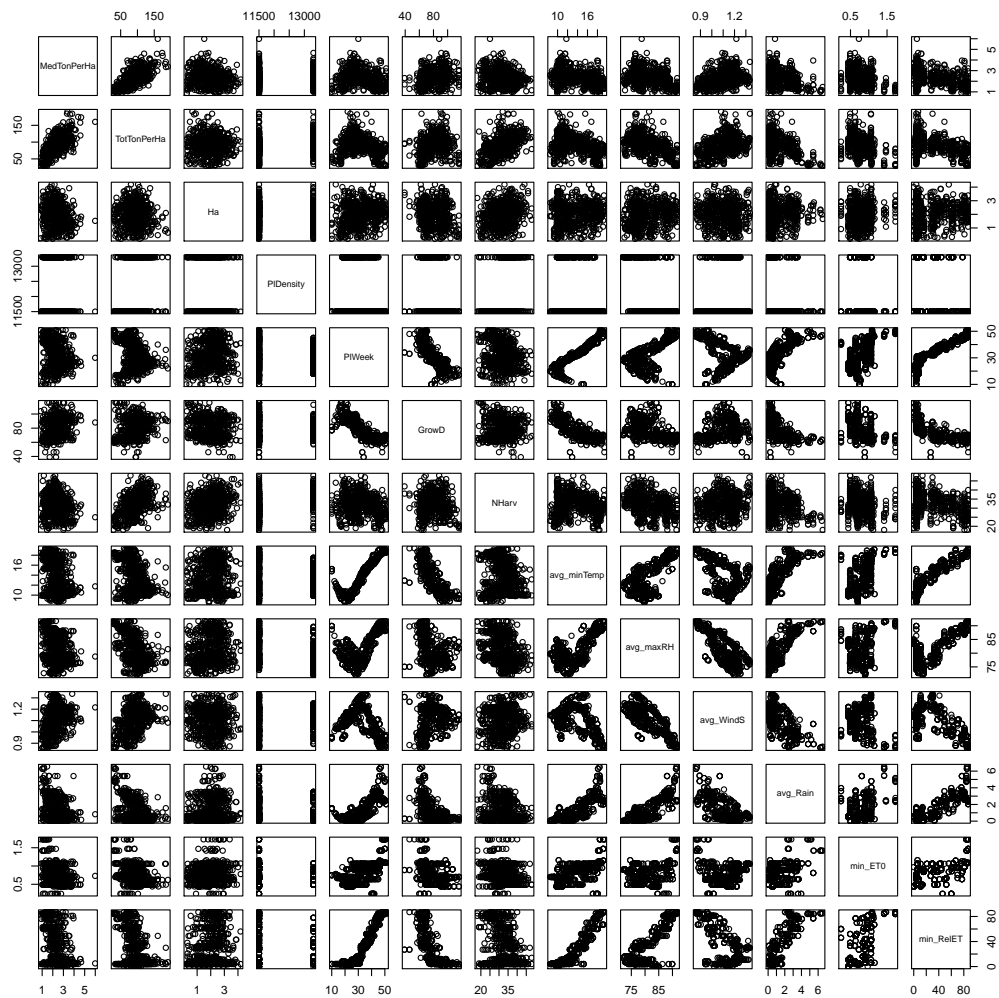


Figure 3.60: Scatterplots of the training set crop variables and summary statistics of the weather time series.

the modelling analyses of Chapters 4 to 6, and Table 3.15 on page 125 lists the model responses.

The exploratory analyses in this chapter have uncovered several relationships amongst the variables. However, scatterplots and correlation coefficients provide a rather myopic view of relationships. Correlation coefficients only indicate the strength of the *linear* relationship between variable *pairs*. Scatterplots show both linear and nonlinear relationships, but can only include two or three variables. SLT modelling methods, which are the focus of the next few chapters, address some of these limitations by allowing multiple input variables in the models. Some of the techniques, such as the tree-based techniques, go

further by allowing interaction terms amongst the input variables.

An examination of the relationships between harvest density and the crop and weather variables in Figure 3.60 (these relationships can be seen more clearly in Figures 3.20 to 3.21 on pages 83–84, Figures 3.23 to 3.24 on page 85, Figure 3.28 on page 87, Figure 3.31 on page 89, Figures 3.52 to 3.53 on pages 114–115 and Figures 3.55 to 3.58 on pages 116–117) reveals that the predictors have either linear—planting density (Figure 3.21), number of harvest events (Figure 3.31), average maximum RH (Figure 3.53) and average of the daily average wind speed readings (Figure 3.55)—or quadratic—planting time (Figures 3.23–3.24), average daily minimum temperature (Figure 3.52), minimum daily total ET_0 (Figure 3.57) and minimum dekad RE (Figure 3.58)—relationships with the responses. The relationships between harvest density and block area (Figure 3.20), growing period length (Figure 3.28) and average total rainfall (Figure 3.56) are, however, less clear, but do not appear to require models other than a linear or quadratic model. Hence, while most of the variables can be input into the modelling algorithms in their linear forms, the quadratic forms of planting time, average daily minimum temperature, minimum daily total ET_0 and minimum dekad RE would also have to be input into the linear models in order to capture the complexity of their relationships with harvest density.

This would involve the addition of several additional input variables to some of the modelling analyses, which could cause dimensionality problems (see Section 2.4.12 on page 62). Variable selection is therefore implemented in all of the analyses in Chapters 4 to 6. Variable selection should also help to mitigate problems caused by multicollinearity amongst the input variables.

In the next few chapters, the training, validation and test data sets are used to model total and median harvest density from planting, harvest and weather variables. The next chapter, Chapter 4, covers the application of multiple linear regression and methods based on the MLR coefficients to model harvest density.

Table 3.14: Input variables for modelling tomato harvest density. The value ranges of the linear terms (third column) reflect those in the training data set.

variable	description	type and values	units
Crop variables from the tomato farm (2008/09/24–2015/11/09):			
Ha	Area of the block crop.	real number; 0.20–4.20	ha
PIDensity	The number of seedlings planted in the block per unit area.	categorical; 11 500 or 13 300	plants/ha
PIWeek; PIWeek_2	Week of the year in which the block crop was planted. PIWeek_2 = (PIWeek)².	integer; 10–51 (linear term)	calendar weeks (linear)
GrowD	Number of days between the planting of the seedlings into the block and the first harvest date.	integer; 39–116	d
NHarv	Number of days on which tomatoes were harvested from the block crop.	integer; 18–48	d
Summary statistics of weather time series from the ARC (2008/04/22–2016/04/05):			
avg_minTemp; avg_minTemp_2	Average of the daily minimum temperature readings for the block crops. avg_minTemp_2 = (avg_minTemp)².	real number; 8.48–19.40 (linear term)	°C (linear term)
avg_maxRH	Average of the daily maximum relative humidity readings for the block crops.	real number; 71.87–91.72	%
avg_WindS	Average of the daily average wind speed readings for the block crops.	real number; 0.86–1.34	m/s
avg_Rain	Average of the daily total rainfall readings for the block crops.	real number; 0.02–6.55	mm/d
min_ET0; min_ET0_2	Minima of the daily total ET ₀ readings for the block crops. min_ET0_2 = (min_ET0)².	real number; 0.24–1.73 (linear term)	mm/d (linear term)
Summary statistics of weather time series from EARS-E2M (2008/01/01–2016/02/21):			
min_RelET; min_RelET_2	Minima of the dekad RE readings for the block crops. min_RelET_2 = (min_RelET)².	real number; 0–87 (linear term)	% (linear term)

Table 3.15: Variables modelled by the tomato harvest models. The value ranges (third column) reflect those in the training data set.

variable	description	type and values	units
MedTonPerHa	Median harvest density i.e. the median quantity harvested from the block crop over its harvest period, divided by its area.	real number; 0.87–5.99	t/ha
TotTonPerHa	Total harvest density i.e. the cumulative harvest of the block crop over its harvest period, divided by its area.	real number; 28.63–190.80	t/ha

Chapter 4

The linear model and related methods

4.1 Introduction

The exploratory analyses of the previous chapter provide indications of relationships between pairs of variables. An advantage of statistical modelling techniques such as multiple linear regression over these simpler analyses is that they can incorporate several input variables simultaneously. This enables relationships between the individual input variables and the response to be explored while taking the influence of other input variables on the response into account. Moreover, the modelling techniques each make specific assumptions about the nature of the relationships between the input variables and the response. If the assumptions of a particular technique happen to approximate the true relationship well, then that technique is likely to yield an accurate model. Consequently, applying statistical modelling algorithms to a problem provides the opportunity to understand and predict the relationships between the input variables and the response more accurately.

This chapter focuses on the application of multiple linear regression and the lasso to modelling harvest density, while Chapters 5 to 6 cover tree-based modelling algorithms. To get an idea of how well each technique models harvest quantities, the prediction accuracy of the modelling techniques are compared to that of the null model, which is discussed next.

4.2 The null model

4.2.1 Introduction and methods

The *null model* does not involve any input variables, but rather simply predicts the response of all observations using the mean of the training responses. As such, the null model is one of the simplest models that can be fit to the training data, and is therefore a good “baseline” model against which the fit of more sophisticated models can be measured.

The R code for preparing the training, validation and test data sets for modelling and the code for fitting the null model to median and total harvest density are presented in Sections A.1 on page 204 and A.2 on page 207, respectively.

4.2.2 Results

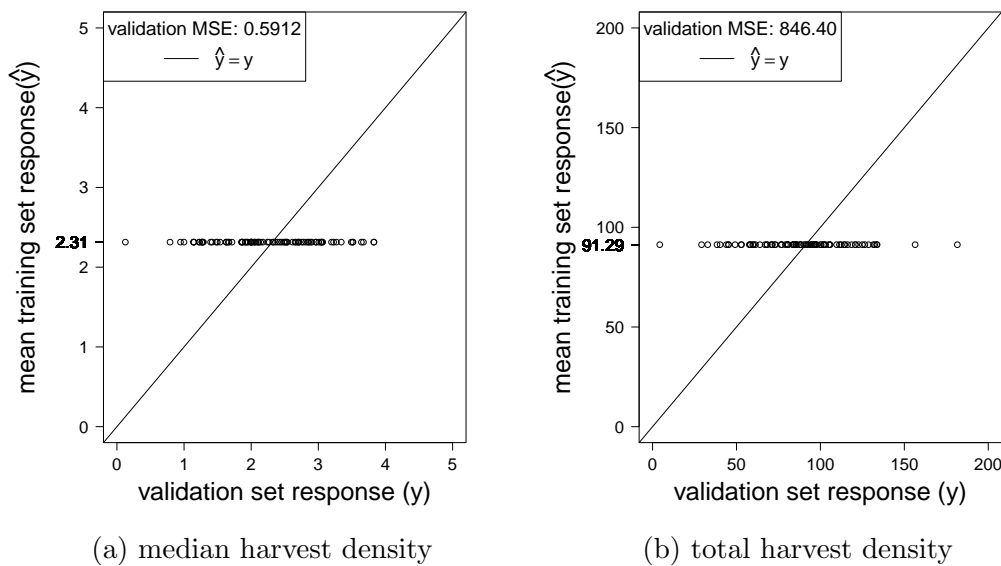


Figure 4.1: Model predictions versus validation responses for the null models of (a) median and (b) total harvest density. The 45° line in each plot indicates the positions that the points would hold if there were perfect correlation between the validation set responses and the model predictions.

Figure 4.1 shows the responses in the validation set plotted against the responses predicted by the null model. The mean of the median harvest density

(MedTonPerHa) values in the training set is 2.3143 t/ha (Figure 4.1a) and that of total harvest density (TotTonPerHa) is 91.2931 t/ha (Figure 4.1b), and the null models therefore simply predict the median and total harvest densities of unseen cases using these two mean values. This results in validation MSE values of $MSE_{\text{validate}} = 0.5912$ and $MSE_{\text{validate}} = 846.40$ for median and total harvest density, respectively.

4.3 Multiple linear regression

4.3.1 Introduction

Unlike the null model, MLR and other modelling techniques incorporate the input variables into their structures, producing a function that predicts the mean of the response in the case of regression based on the values of the input variables with an additive error model and squared-error loss (see (2.4.1) on page 19 and (2.4.27) on page 56).

Linear models, which model the response as a linear function of the input variable coefficients, have a number of advantages over other modelling techniques. The simple structure of the MLR model makes it easy to interpret, providing rough insights into the relative contributions of the different input variables towards the response. In addition, their rigid structural assumptions make them robust to noise. They should therefore find similar relationships even in the presence of a low signal-to-noise ratio, provided that the underlying process is linear. Moreover, many algorithms are based on the linear model, and if the process being modelled is approximately linear over the range of values of primary interest, then one should be able to find a technique that models the process accurately. However, linearity between the input variables and the response is more the exception than the rule in the vast majority of natural processes. Nevertheless, the MLR model often provides a reasonably good fit to the data. This technique is explored in this section.

4.3.2 Methods

MLR models containing three different subsets of the 15 input variables listed in Table 3.14 on page 124 were fit to the training data:

- the eleven linear terms,
- the eleven linear terms and four quadratic terms, and
- the linear and quadratic terms selected using best subset selection (BSS)

in R. Best subsets selection was performed using the `leaps` package. The R code for BSS as well as for fitting the MLR models is provided in Section A.3 on page 208.

4.3.3 Results

Figure 4.2 on the following page and Figure 4.4 on page 132 display R^2 , adj. R^2 , Mallows' C_p statistic and Bayesian information criterion (BIC) for median and total harvest density MLR models of different sizes, and Figures 4.3 on page 131 and 4.5 on page 133 show the model terms included for each model size for the different statistics. The top left-hand plot in Figures 4.2 and 4.4 show the R^2 values for the different model sizes. Since R^2 measures the proportion of the variance in the response explained by the model and since all of the input variables are to a certain extent correlated with the two response variables, R^2 is a nondecreasing function with respect to model complexity. The R^2 curve levels off with an increase in model complexity, because the best models of the smaller sizes are likely to contain only the most informative input variables, whereas the best models of the larger sizes are also likely to contain less informative input variables. Moreover, the varying levels of correlation amongst the input variables results in the less informative input variables appearing even less informative than they actually are based on the R^2 curve. Hence, at larger model sizes, increasing the model size results in smaller improvements in the fit of the model to the training data.

Since R^2 is a nondecreasing function with respect to model complexity, it is a poor indicator of the optimum model size (see Section 2.4.2.6 on page 28). On the other hand, adjusted R^2 , the C_p statistic and BIC, which are also functions of the training error, penalise model complexity, thereby providing better indications of optimum model size.

As is usually the case, BSS found different optimum sizes for the median and total harvest density models, depending on the statistic examined. The optimum model sizes for median harvest density are 9 (BIC), 10 (C_p) and

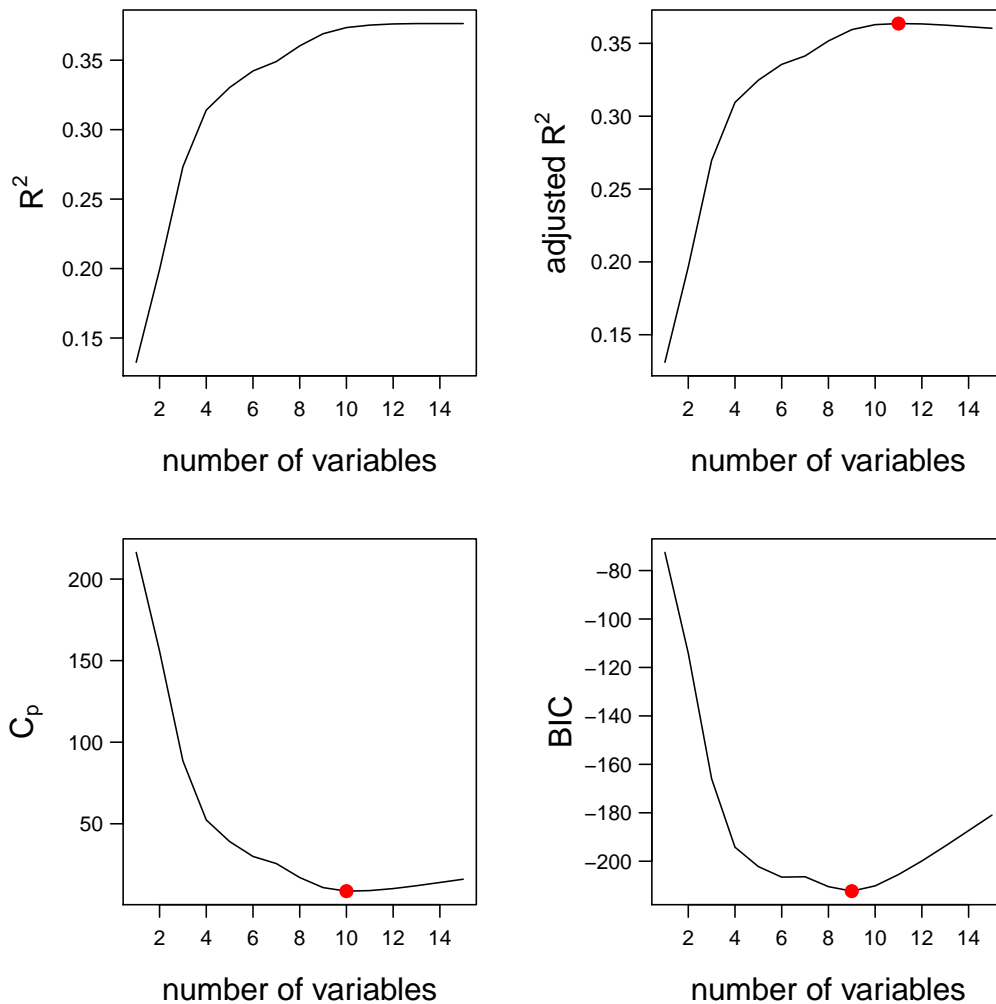


Figure 4.2: Statistics for median harvest density models of different sizes chosen by best subset selection. The complexity measures are R^2 (top left), adjusted R^2 (top right), Mallows' C_p statistic (bottom left) and Bayesian information criterion (BIC; bottom right). A red dot marks the optimum value for each statistic.

11 (adj. R^2 ; Figure 4.2), while those for total harvest density are 8 (BIC) and 11 (adj. R^2 and C_p ; Figure 4.4). Hence, BIC, which tends to penalise model complexity heavily (see Section 2.4.2.6 on page 28), was optimised by the smallest model for both responses. BIC was arbitrarily chosen as the criterion for selecting the optimum model size for median and total harvest density (which are indicated by the red dot in the bottom right-hand plot in Figures 4.2 and 4.4). The terms included in the resulting models are indicated by black blocks in the top row of the bottom right-hand plot in Figures 4.3

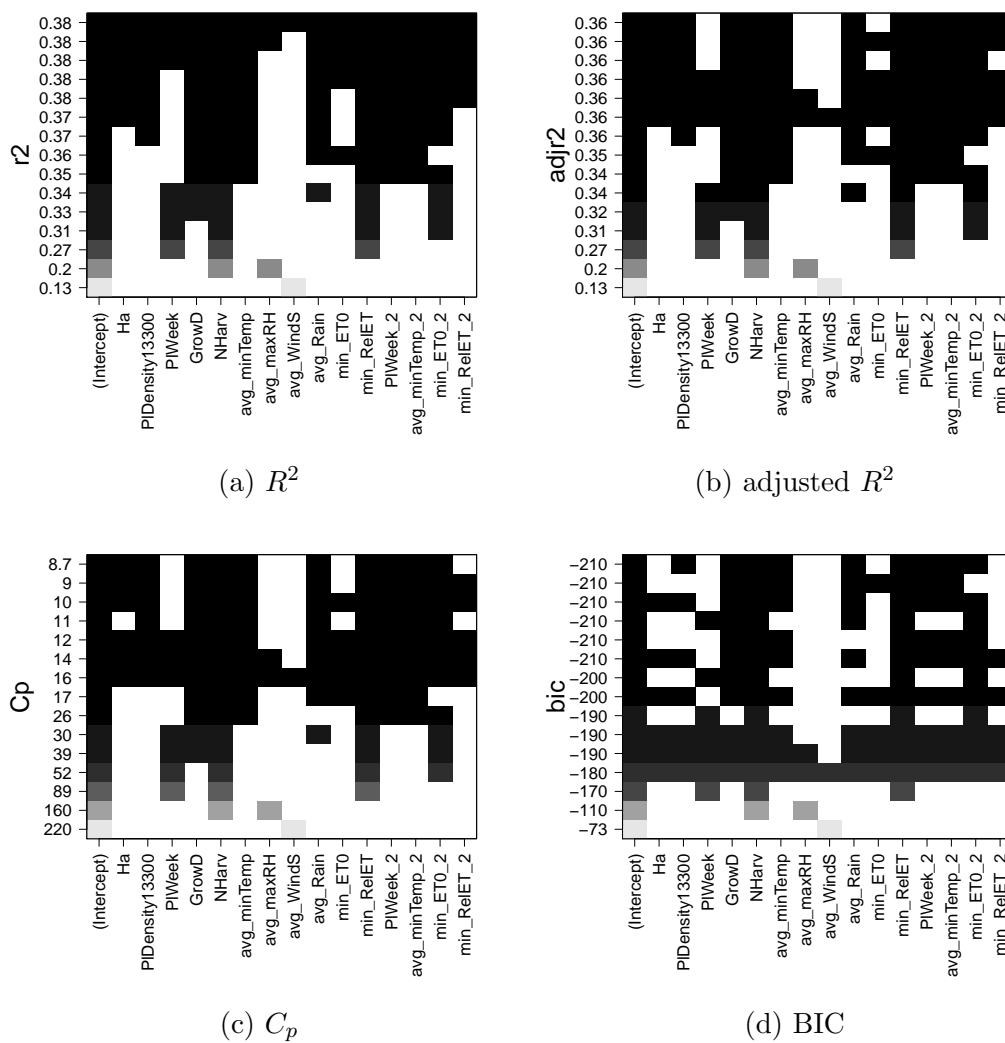


Figure 4.3: Input terms included in the median harvest density models for different complexity measures. The largest model contains all eleven linear terms and all four quadratic terms. Included terms are indicated by coloured blocks, whereas white signifies exclusion. The model with the optimum value is provided in the top row for each statistic.

and 4.5.

Figure 4.6 on page 134, Figure 4.9 on page 138 and Figure 4.12 on page 143 show plots of the residuals from the median harvest density MLR models containing the linear terms (Figure 4.6), the linear and quadratic terms (Figure 4.9), and the linear and quadratic terms chosen by BSS (Figure 4.12), while Figure 4.7 on page 136, Figure 4.10 on page 140 and Figure 4.13 on page 145 show the corresponding total harvest density plots. The residual plots iden-

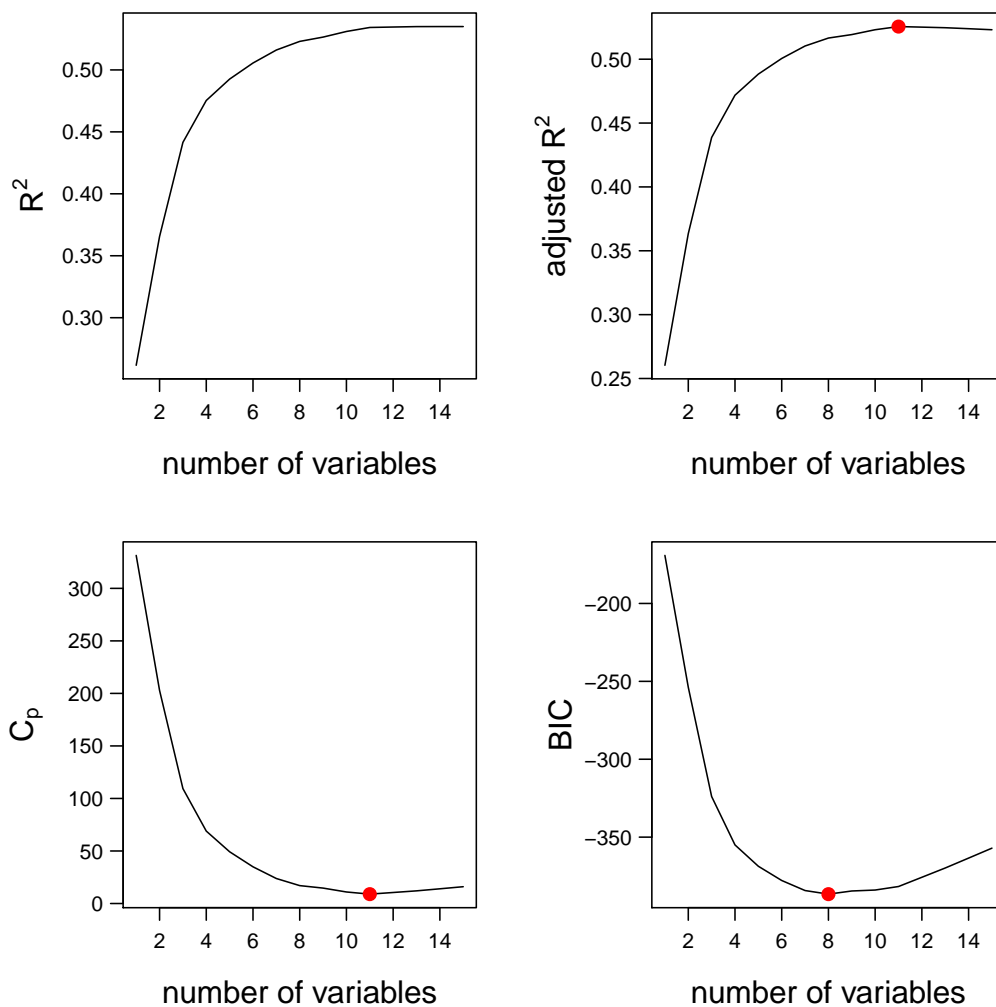


Figure 4.4: Statistics for total harvest density models of different sizes chosen by best subset selection. The complexity measures are R^2 (top left), adjusted R^2 (top right), Mallows' C_p statistic (bottom left) and Bayesian information criterion (BIC; bottom right). A red dot marks the optimum value for each statistic.

tify different observations as outliers for the two responses. Consequently, no observation looks obviously suspicious. The curves fit to the residuals of the six models are all reasonably flat (top left-hand plot in Figures 4.6, 4.7, 4.9, 4.10, 4.12 and 4.13), with the two median harvest density models containing quadratic terms showing the greatest curvature (top left-hand plots in Figures 4.9 and 4.12). However, the training residuals can be misleading in that those in more poorly represented parts of the training hull tend to have low variance, despite the fact that poor sampling is likely to result in the model be-

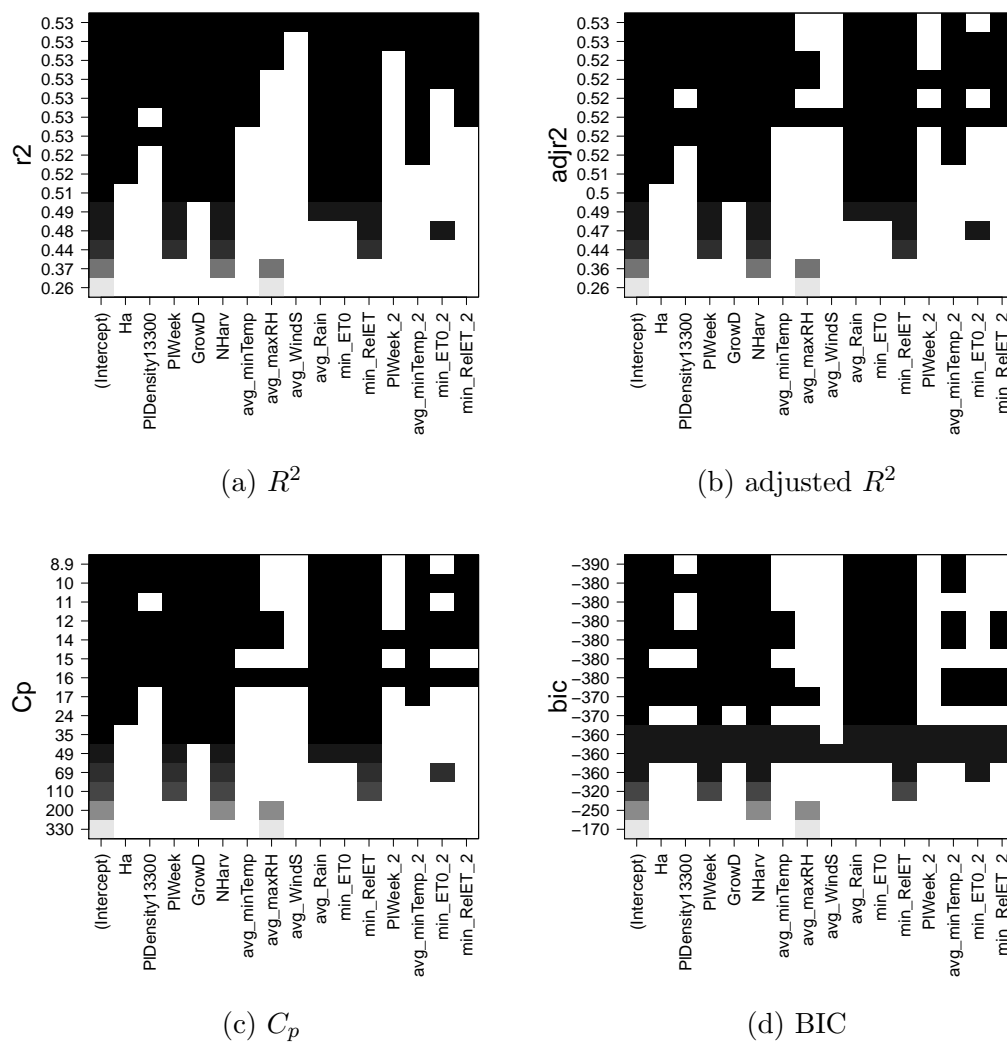


Figure 4.5: Input terms included in the total harvest density models for different complexity measures. The largest model contains all eleven linear terms and all four quadratic terms.

ing less accurate in these regions (Rencher and Schaalje, 2008). This is evident from the almond-shaped residual plots in the top left of Figures 4.6, 4.7, 4.9, 4.10, 4.12 and 4.13. For this reason, it is often preferable to scale the residuals to have an expected value of 0 and variance of 1. The resulting standardised residuals, shown in the bottom left-hand plot in Figures 4.6, 4.7, 4.9, 4.10, 4.12 and 4.13, have a far more even variance across the input space, suggesting no obvious divergence from the assumption of *homoscedasticity*. Mostly the same observations are identified as outliers before and after standardisation. The curves fitted to the plots of the standardised residuals against the fitted values

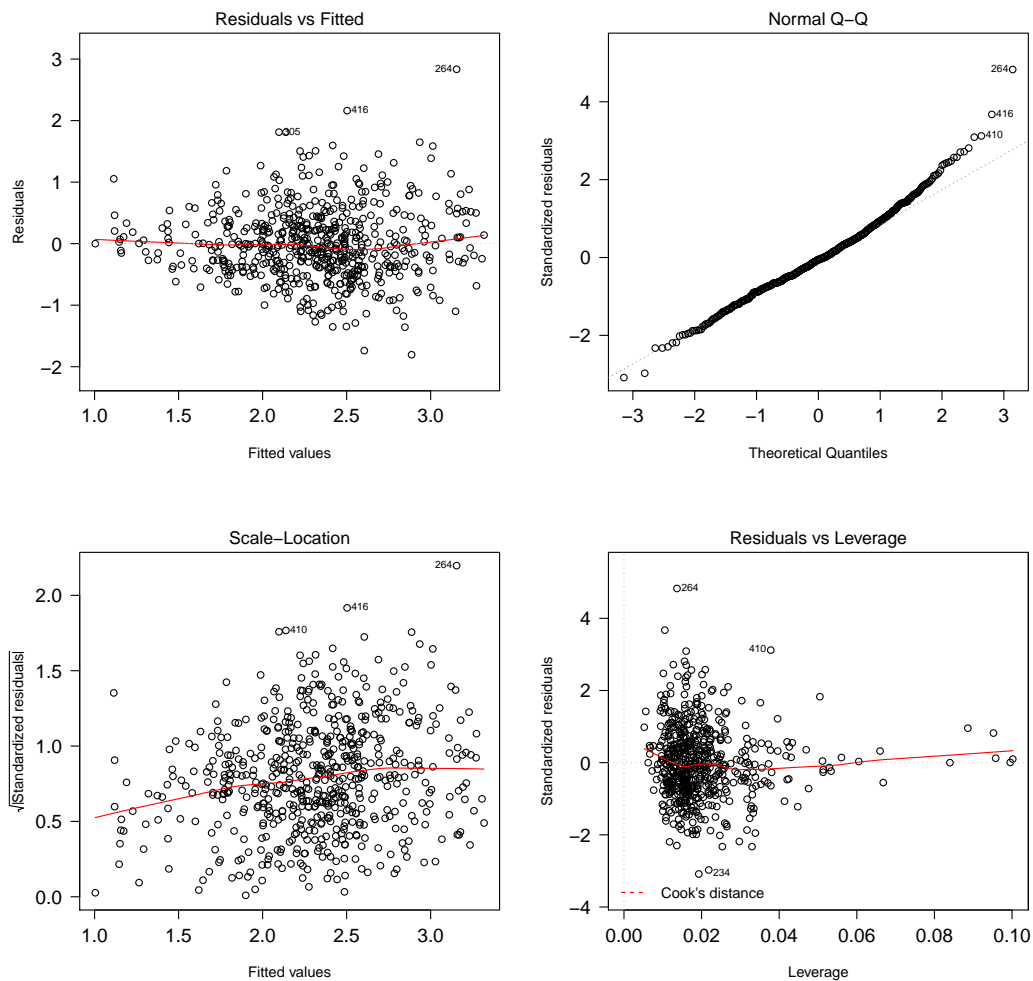


Figure 4.6: Model residuals of the multiple linear regression model of median harvest density containing the eleven linear terms. Shown are the model residuals against the fitted values (top left), a normal QQ plot of the standardised model residuals (top right), the square root of the standardised model residuals against the fitted values (bottom left) and the standardised residuals against the leverage of the training observations (bottom right).

are mostly flat in the six models, indicating that the assumptions made by the MLR algorithm are largely satisfied.

One of the assumptions made by additive error models, including the linear model, is that the error terms share a normal distribution with a shared variance and an expected value of 0 (see (2.4.1) on page 19). The QQ plots (top right-hand plot in Figures 4.6, 4.7, 4.9, 4.10, 4.12 and 4.13) suggest that a normal distribution fits the residuals quite well, although there is some deviation

Table 4.1: Coefficients for the multiple linear regression model of median harvest density containing only the linear terms (RSE: 0.5911 on 588 df; R^2 : 0.3518; adj. R^2 : 0.3397; F -statistic: 29.01 on 11 and 588 df; p -value: $< 2.2 \times 10^{-16}$). The significance levels in the last column are coded as follows: “***”: $p < 0.001$; “**”: $0.001 < p < 0.01$; “*”: $0.01 < p < 0.05$; “.”: $0.05 < p < 0.1$; “ ”: $p > 0.1$.

	Coefficient	Standard error	t -statistic	p -value	
Intercept	2.27	1.2472	1.82	0.0694	.
Ha	-0.07	0.0326	-2.09	0.0370	*
PIDensity13300	-0.17	0.0624	-2.70	0.0071	**
PIWeek	0.07	0.0090	7.64	< 0.0001	***
GrowD	0.01	0.0036	2.65	0.0083	**
NHarv	-0.03	0.0053	-5.13	< 0.0001	***
avg_minTemp	-0.02	0.0295	-0.74	0.4571	
avg_maxRH	-0.01	0.0112	-0.70	0.4866	
avg_WindS	0.31	0.3998	0.78	0.4333	
avg_Rain	-0.13	0.0410	-3.14	0.0018	**
min_ET0	-0.77	0.1174	-6.57	< 0.0001	***
min_RelET	-0.02	0.0040	-3.77	0.0002	***

in the tails from that of a normal distribution.

The leverage plots (bottom right-hand plot in Figures 4.6, 4.7, 4.9, 4.10, 4.12 and 4.13) uncover observations that have disproportionately large influences on the model structure. The main outliers (labelled in each plot) do not have leverages much higher than those of most of the training observations, with the possible exception of observation 410 in the median harvest density MLR models (see bottom right-hand plot in Figures 4.6, 4.9 and 4.12). However, observation 410 is not an outlier in the total harvest density models, and is therefore not obviously incorrect.

The MLR model plots in Figures 4.6, 4.7, 4.9, 4.10, 4.12 and 4.13 suggest that the crop data set satisfies most of the assumptions made by the MLR algorithm, and MLR was therefore used to model the responses.

Table 4.1, Table 4.3 on page 139 and Table 4.5 on page 144 and Table 4.2 on page 137, Table 4.4 on page 141 and Table 4.6 on page 146 provide model statistics and the estimated coefficients for the median and total harvest density MLR models, respectively. The complexity measures of the three median harvest density MLR models are similar to one another, as are those of the total harvest density MLR models. For median harvest density, the true responses

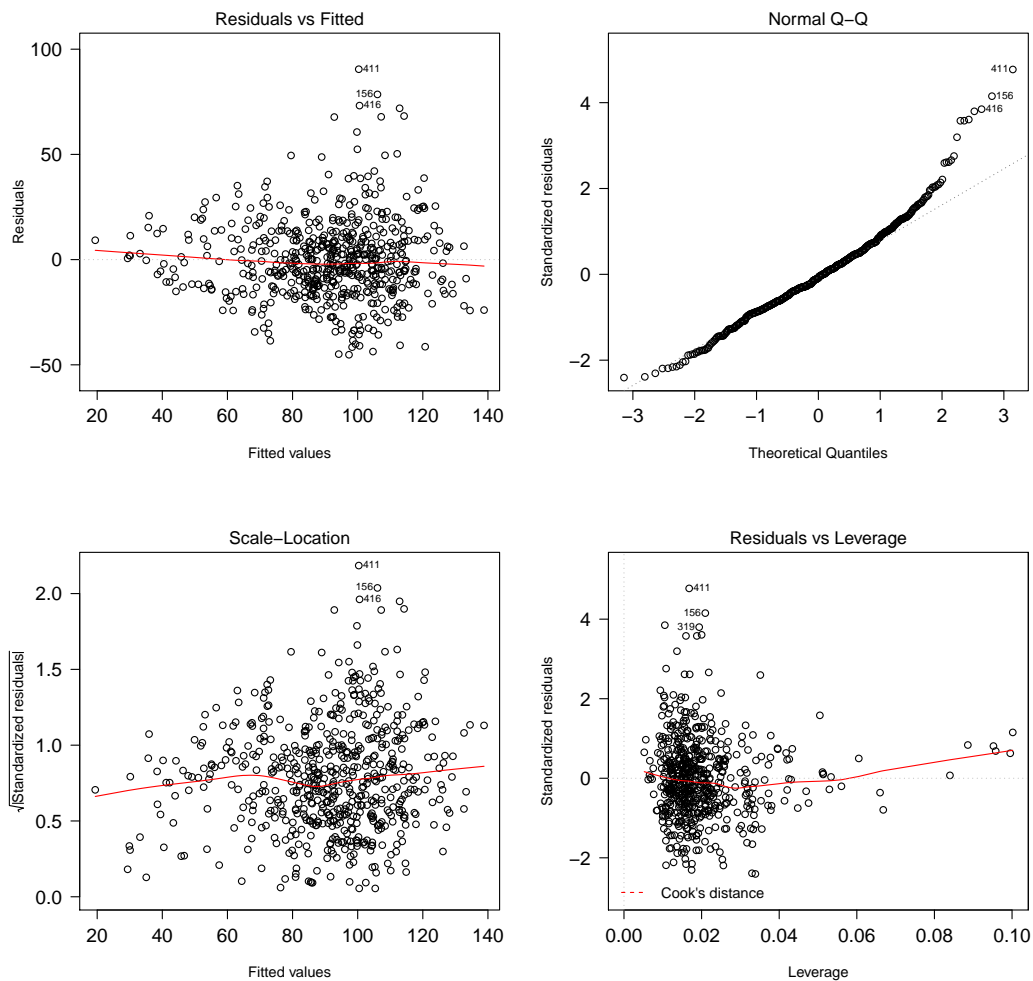
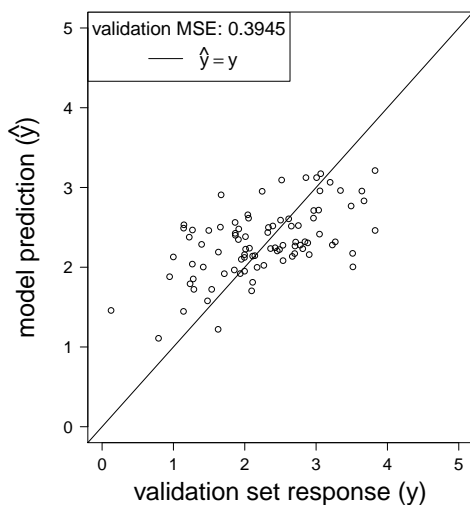


Figure 4.7: Model residuals of the multiple linear regression model of total harvest density containing the eleven linear terms.

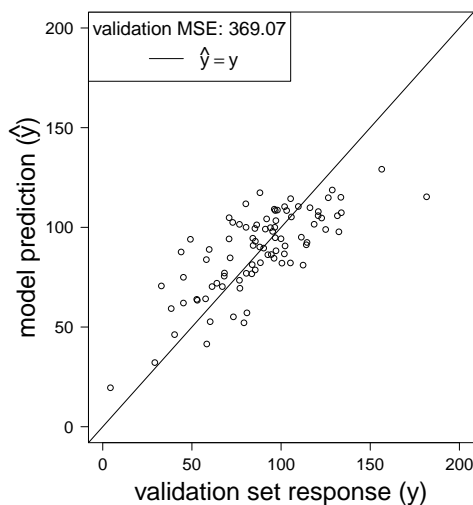
deviate from the MLR model predictions by, on average, $RSE = 0.5911$ t/ha (model containing linear terms), $RSE = 0.5818$ t/ha (model containing linear and quadratic terms) and $RSE = 0.5822$ t/ha (model containing BSS terms). Since the average median harvest density is 2.3143 t/ha (see Section 4.2.2 on page 127), the percentage error is about 25%. Taking sample size and the number of estimated parameters into account, $adj. R^2 = 0.3397$ (linear terms), $adj. R^2 = 0.3603$ (linear and quadratic terms) and $adj. R^2 = 0.3594$ (BSS terms) of the total sum of squares of deviations of the responses about their means are accounted for by the models. In the case of total harvest density, the true responses deviate from the responses predicted by the models by $RSE =$

Table 4.2: Coefficients for the multiple linear regression model of total harvest density containing only the linear terms (RSE: 19.12 on 588 df; R^2 : 0.5249; adj. R^2 : 0.516; F -statistic: 59.05 on 11 and 588 df; p -value: $< 2.2 \times 10^{-16}$). The significance levels in the last column are coded as follows: “***”: $p < 0.001$; “**”: $0.001 < p < 0.01$; “*”: $0.01 < p < 0.05$; “.”: $0.05 < p < 0.1$; “ ”: $p > 0.1$.

	Coefficient	Standard error	t -statistic	p -value	
Intercept	-11.54	40.3364	-0.29	0.7749	
Ha	-4.53	1.0528	-4.30	< 0.0001	***
PIDensity13300	-3.86	2.0184	-1.91	0.0566	.
PIWeek	3.05	0.2923	10.43	< 0.0001	***
GrowD	0.26	0.1154	2.25	0.0250	*
NHarv	2.14	0.1716	12.47	< 0.0001	***
avg_minTemp	-2.79	0.9543	-2.92	0.0036	**
avg_maxRH	0.08	0.3613	0.21	0.8364	
avg_WindS	2.64	12.9316	0.20	0.8386	
avg_Rain	-5.38	1.3255	-4.06	< 0.0001	***
min_ET0	-25.60	3.7970	-6.74	< 0.0001	***
min_RelET	-0.54	0.1297	-4.16	< 0.0001	***



(a) median harvest density



(b) total harvest density

Figure 4.8: Model predictions versus validation responses for the multiple linear regression models of (a) median and (b) total harvest density containing the linear terms. The 45° line in each plot indicates the positions that the points would hold if there were perfect correlation between the validation set responses and the model predictions.

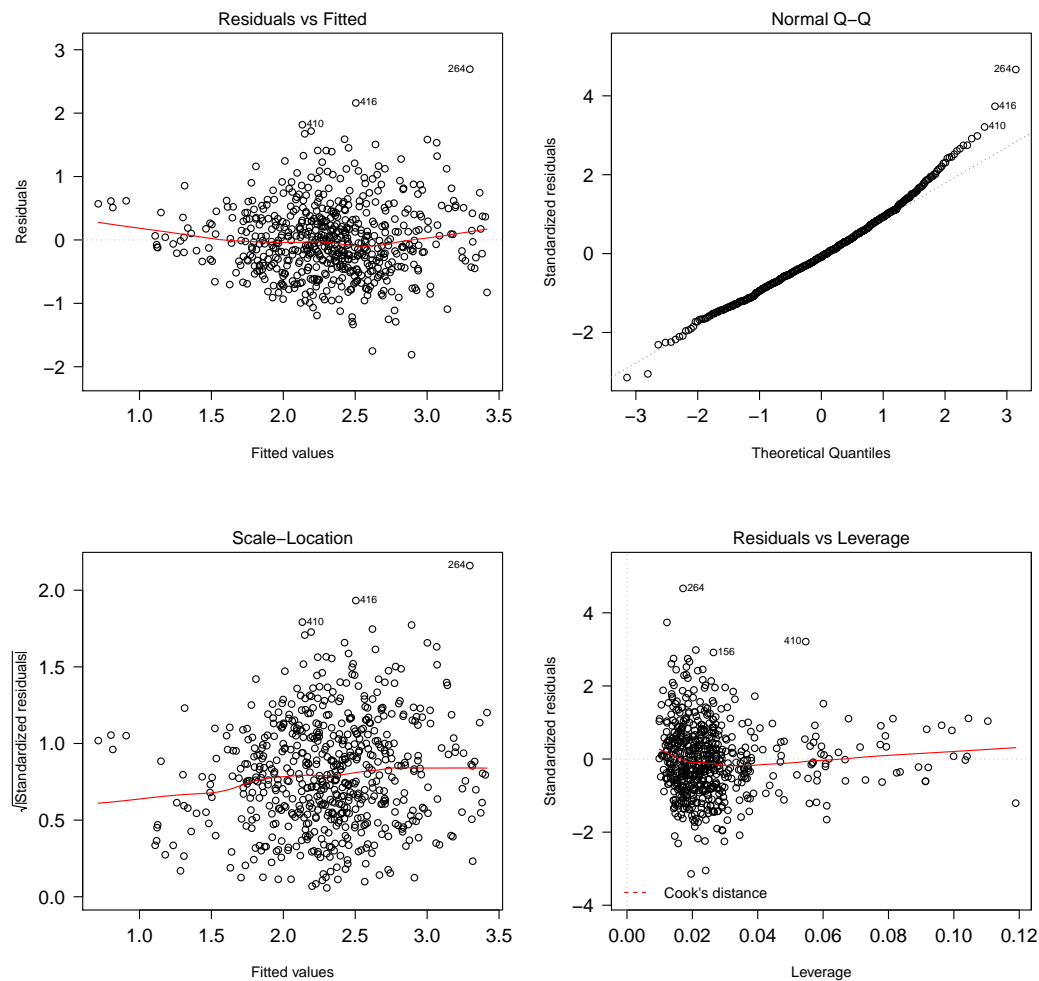


Figure 4.9: Model residuals of the multiple linear regression model of median harvest density containing the eleven linear and four quadratic terms.

19.12 t/ha (linear terms), RSE = 18.98 t/ha (linear and quadratic terms) and RSE = 19.11 t/ha (BSS terms), on average. With an average total harvest density of 91.29 t/ha (Section 4.2.2 on page 127), the three MLR models will result in percentage errors of about 21 %. After adjusting for sample size and the number of estimated parameters, $\text{adj. } R^2 = 0.516$ (linear terms), $\text{adj. } R^2 = 0.523$ (linear and quadratic terms) and $\text{adj. } R^2 = 0.5166$ (BSS terms) of the variation in the training responses is explained by the models. Thus, for both median and total harvest density, the model containing all eleven of the linear terms and all four of the quadratic terms has both the highest $\text{adj. } R^2$ value and the lowest RSE value, indicating that the largest model is the best for

modelling both responses. The fact that the adj. R^2 values of the three median harvest density models are far lower than those of the three total harvest density models indicates that there is more variance of the training responses about their model predictions in the median than in the total harvest density models. The linear model therefore models total harvest density better than it does median harvest density. However, despite the greater variance in the median harvest density MLR models, the three median and the three total harvest density models all have F -statistics with p -values $< 2.2 \times 10^{-16} \ll 0.05$. Hence, each of the MLR models contain at least one input variable that is informative for modelling its response. All of the MLR models are therefore useful for predicting harvest density (see (2.4.13) on page 28).

Table 4.3: Coefficients for the multiple linear regression model of median harvest density containing the linear and quadratic terms (RSE: 0.5818 on 584 df; R^2 : 0.3764; adj. R^2 : 0.3603; F -statistic: 23.5 on 15 and 584 df; p -value: $< 2.2 \times 10^{-16}$). The significance levels in the last column are coded as follows: “***”: $p < 0.001$; “**”: $0.001 < p < 0.01$; “*”: $0.01 < p < 0.05$; “.”: $0.05 < p < 0.1$; “ ”: $p > 0.1$.

	Coefficient	Standard error	t -statistic	p -value	
Intercept	-2.2499	1.6732	-1.35	0.1792	
Ha	-0.0615	0.0321	-1.92	0.0558	.
PIDensity13300	-0.1920	0.0623	-3.08	0.0022	**
PIWeek	0.0159	0.0386	0.41	0.6803	
GrowD	0.0130	0.0037	3.56	0.0004	***
NHarv	-0.0254	0.0053	-4.76	< 0.0001	***
avg_minTemp	0.6969	0.1701	4.10	< 0.0001	***
avg_maxRH	-0.0010	0.0115	-0.08	0.9337	
avg_WindS	0.0322	0.4661	0.07	0.9450	
avg_Rain	-0.1514	0.0421	-3.59	0.0004	***
min_ET0	-0.3701	0.4311	-0.86	0.3909	
min_RelET	-0.0221	0.0069	-3.21	0.0014	**
PIWeek_2	0.0013	0.0008	1.66	0.0984	.
avg_minTemp_2	-0.0292	0.0069	-4.26	< 0.0001	***
min_ET0_2	-0.2602	0.2317	-1.12	0.2618	
min_RelET_2	0.0001	0.0001	1.34	0.1825	

Despite the similarities in the complexity measures amongst the three median harvest density MLR models and amongst the three total harvest density MLR models outlined in the previous paragraph, the model coefficients of the

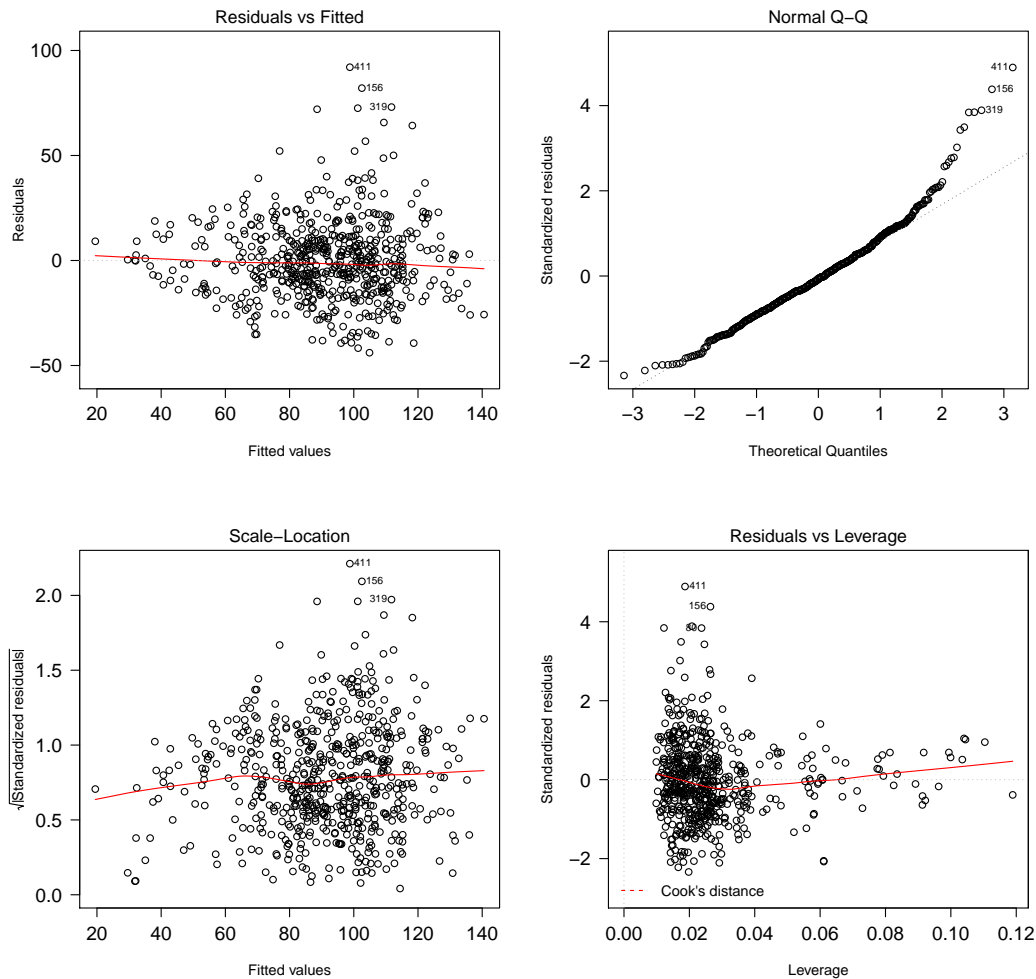


Figure 4.10: Model residuals of the multiple linear regression model of total harvest density containing the eleven linear and four quadratic terms.

individual terms differ depending on the terms included in the models for the two responses. However, despite fluctuations in the coefficient values amongst the models, the number of harvest events (N_{Harv}), the average of the daily total rainfall readings (avg_Rain) and the minima of the dekad RE readings (min_Re1ET) emerge as being highly significant input variables for both responses. This is not surprising, since these three variables are amongst the most highly correlated input variables with the two responses (see Table 3.13 on page 123). The MLR models in Tables 4.1 to 4.6 imply that, if all other input variables are held constant, an additional harvest event results in an average 0.0254 t/ha to 0.0275 t/ha *decrease* in median and an average 2.0216 t/ha

to 2.1464 t/ha *increase* in total harvest density; an increase of 1 mm/d in average daily total rainfall readings results in an average 0.1288 t/ha to 0.1514 t/ha *decrease* in median and an average 4.7971 t/ha to 6.1499 t/ha *decrease* in total harvest density; and an increase of 1 % in the minimum of the dekad RE readings results in an average 0.0151 t/ha to 0.0221 t/ha *decrease* in median and an average 0.5209 t/ha to 0.9686 t/ha *decrease* in total harvest density under the assumption of linearity, depending on the terms included in the MLR model.

Table 4.4: Coefficients for the multiple linear regression model of total harvest density containing the linear and quadratic terms (RSE: 18.98 on 584 df; R^2 : 0.5350; adj. R^2 : 0.523; F -statistic: 44.79 on 15 and 584 df; p -value: $< 2.2 \times 10^{-16}$). The significance levels in the last column are coded as follows: “***”: $p < 0.001$; “**”: $0.001 < p < 0.01$; “*”: $0.01 < p < 0.05$; “.”: $0.05 < p < 0.1$; “ ”: $p > 0.1$.

	Coefficient	Standard error	t -statistic	p -value	
Intercept	-127.091	54.5774	-2.33	0.0202	*
Ha	-4.359	1.0471	-4.16	< 0.0001	***
PIDensity13300	-3.913	2.0314	-1.93	0.0545	.
PIWeek	3.456	1.2604	2.74	0.0063	**
GrowD	0.299	0.1195	2.50	0.0128	*
NHarv	2.146	0.1743	12.32	< 0.0001	***
avg_minTemp	12.054	5.5495	2.17	0.0303	*
avg_maxRH	0.250	0.3751	0.67	0.5047	
avg_WindS	-0.411	15.2026	-0.03	0.9784	
avg_Rain	-6.150	1.3744	-4.47	< 0.0001	***
min_ET0	-18.640	14.0602	-1.33	0.1855	
min_RelET	-0.969	0.2241	-4.32	< 0.0001	***
PIWeek_2	-0.006	0.0258	-0.22	0.8289	
avg_minTemp_2	-0.551	0.2240	-2.46	0.0142	*
min_ET0_2	-5.019	7.5564	-0.66	0.5069	
min_RelET_2	0.007	0.0026	2.55	0.0110	*

Notice that, despite the fact that harvest *densities* are being modelled, the area of the crop (Ha) emerges as a significant input variable in some of the MLR models (see Table 4.1 on page 135, Table 4.2 on page 137, Table 4.4 and Table 4.6 on page 146). Area also emerges as informative in the models in the rest of this chapter and in subsequent chapters, and an explanation for the apparent importance of this variable is posited in Chapter 5.

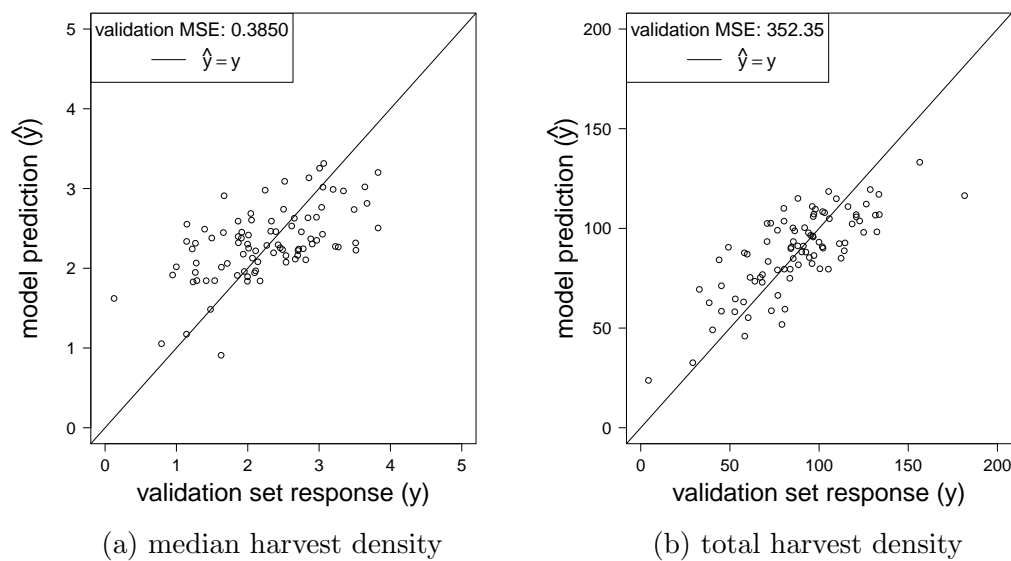


Figure 4.11: Model predictions versus validation responses for the multiple linear regression models of (a) median and (b) total harvest density containing the linear and quadratic terms.

Interestingly enough, the input variables most highly correlated with the two responses—`avg_WindS` for median harvest density ($r = 0.36$) and `avg_maxRH` for total harvest density ($r = -0.51$)—are insignificant in the MLR models containing the linear and the linear and quadratic terms, and absent from the BSS MLR models of the corresponding responses. This is because their coefficient estimators have relatively high variances compared to their coefficient values, probably as a result of the high correlations that exist between these two input variables and many of the other input variables (see Table 3.13 on page 123).

The adjustment of model coefficients depending on the other terms included in the models together with the fact that all three models for each response explain similar proportions of the variance in the training responses despite differences in the terms included suggest a certain degree of dependence amongst the input variables. The fact that many of the input variables are quite highly correlated with one another (see Table 3.13) means that many of the input variables contain similar information. Consequently, a subset of the input variables contains almost as much information for predicting the responses as does the full set of linear and quadratic terms.

Figure 4.8 on page 137, Figure 4.11 and Figure 4.14 on page 146 show

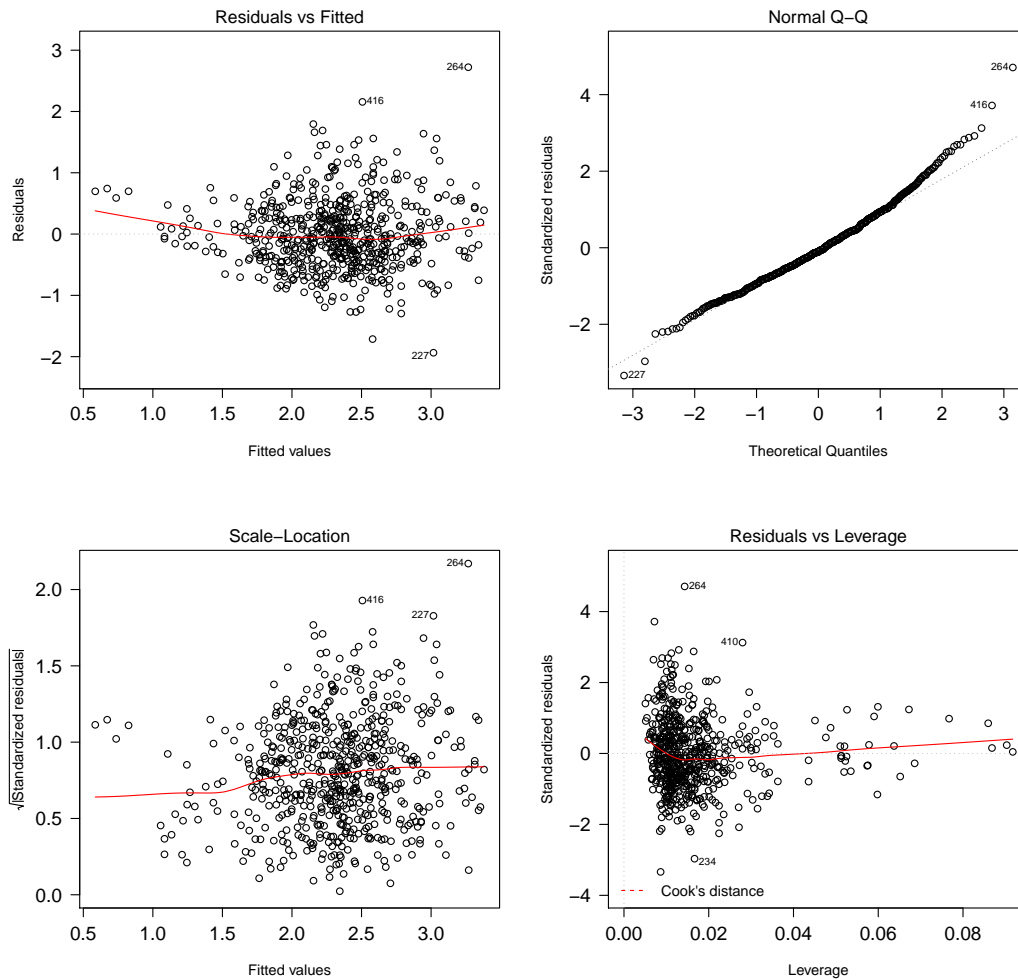


Figure 4.12: Model residuals of the multiple linear regression model of median harvest density containing the linear and quadratic terms chosen by best subset selection.

the validation errors for the three median (left-hand plots) and three total (right-hand plots) harvest density models. For each response, the validation errors are similar to one another: For median harvest density, $MSE_{\text{validate}} = 0.3945$ (linear terms), $MSE_{\text{validate}} = 0.3850$ (linear and quadratic terms) and $MSE_{\text{validate}} = 0.3953$ (BSS terms), and for total harvest density, $MSE_{\text{validate}} = 369.07$ (linear terms), $MSE_{\text{validate}} = 352.35$ (linear and quadratic terms) and $MSE_{\text{validate}} = 362.36$ (BSS terms). That being said, the model containing all eleven of the linear terms and all four of the quadratic terms achieves the lowest validation errors for both responses. The fact that, for each response, the

Table 4.5: Coefficients for the multiple linear regression model of median harvest density containing the linear and quadratic terms selected by best subset selection (RSE: 0.5822 on 590 df; R^2 : 0.3690; adj. R^2 : 0.3594; F -statistic: 38.34 on 9 and 590 df; p -value: $< 2.2 \times 10^{-16}$). The significance levels in the last column are coded as follows: “****”: $p < 0.001$; “***”: $0.001 < p < 0.01$; “**”: $0.01 < p < 0.05$; “.”: $0.05 < p < 0.1$; “ ”: $p > 0.1$.

	Coefficient	Standard error	t -statistic	p -value	
Intercept	-1.831	1.0075	-1.82	0.0697	.
PIDensity13300	-0.170	0.0587	-2.90	0.0038	**
GrowD	0.014	0.0036	3.99	0.0001	***
NHarv	-0.028	0.0047	-5.80	< 0.0001	***
avg_minTemp	0.600	0.1188	5.05	< 0.0001	***
avg_Rain	-0.134	0.0362	-3.70	0.0002	***
min_RelET	-0.016	0.0040	-4.04	0.0001	***
PIWeek_2	0.002	0.0002	10.14	< 0.0001	***
avg_minTemp_2	-0.026	0.0047	-5.54	< 0.0001	***
min_ET0_2	-0.430	0.0638	-6.74	< 0.0001	***

model containing all of the linear and quadratic terms scores both the highest adj. R^2 and lowest MSE_{validate} values indicates that each input variable contributes valuable information to the models (despite their high correlations), and that adding all of the input variables does not lead to overfitting under the assumption of linearity.

Since the MLR model containing both the linear and quadratic terms achieved the lowest validation error of the three MLR models fit to each response, all 15 of these linear and quadratic terms were included in all subsequent models in this chapter.

4.4 The lasso

4.4.1 Introduction and methods

The lasso is a technique for regularising the coefficients of the MLR model, the estimators of which can have high variance due to factors such as high correlation amongst some of the input variables and/or a low signal-to-noise ratio. The lasso algorithm shrinks the absolute values of the MLR coefficients towards zero, thereby reducing the variance (and therefore the prediction error)

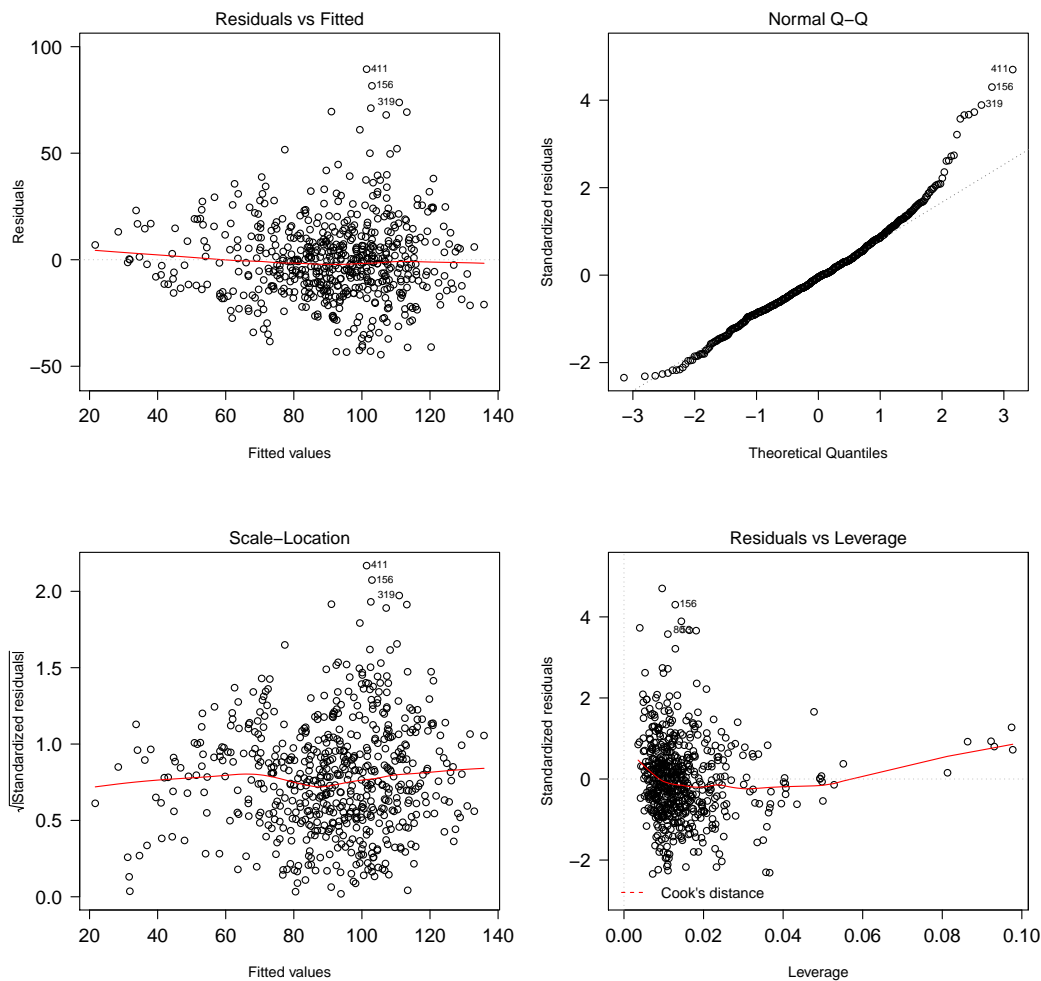


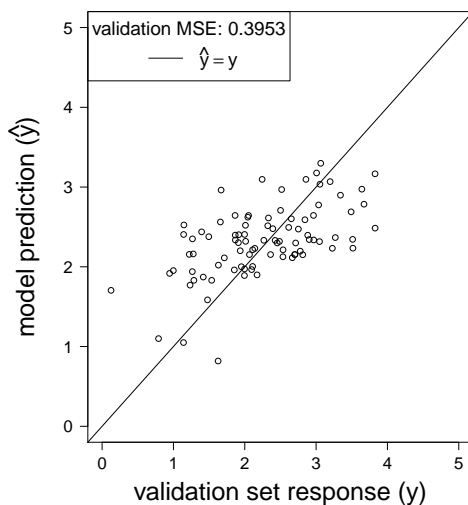
Figure 4.13: Model residuals of the multiple linear regression model of total harvest density containing the linear and quadratic terms chosen by best subset selection.

of the model. Since the optimisation criterion enables some of the coefficients to be shrunk to exactly zero, the lasso also performs variable selection, usually resulting in a sparse model. This makes the lasso even more interpretable than the MLR model. The lasso is especially useful if the MLR model has high variance, which tends to be the case for data sets with a high p/n ratio and/or multicollinearity (Hastie *et al.*, 2009).

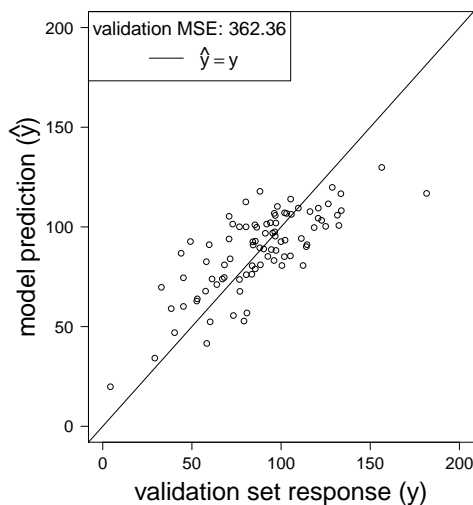
The input variables were standardised to have a mean of 0 and a variance of 1 prior to the analyses. The lasso was fit using the eleven linear terms and four quadratic terms using the `glmnet` package in R. For each response,

Table 4.6: Coefficients for the multiple linear regression model of total harvest density containing the linear and quadratic terms selected by best subset selection (RSE: 19.11 on 591 df; R^2 : 0.5230; adj. R^2 : 0.5166; F -statistic: 81.01 on 8 and 591 df; p -value: $< 2.2 \times 10^{-16}$). The significance levels in the last column are coded as follows: “****”: $p < 0.001$; “***”: $0.001 < p < 0.01$; “**”: $0.01 < p < 0.05$; “.”: $0.05 < p < 0.1$; “ ”: $p > 0.1$.

	Coefficient	Standard error	t -statistic	p -value	
Intercept	-28.01	15.4687	-1.81	0.0707	.
Ha	-4.02	1.0122	-3.97	0.0001	***
PIWeek	3.07	0.2460	12.46	< 0.0001	***
GrowD	0.34	0.1042	3.25	0.0012	**
NHarv	2.02	0.1552	13.03	< 0.0001	***
avg_Rain	-4.80	1.1904	-4.03	0.0001	***
min_ET0	-24.47	3.7626	-6.50	< 0.0001	***
min_RelET	-0.52	0.1149	-4.53	< 0.0001	***
avg_minTemp_2	-0.10	0.0352	-2.94	0.0034	**



(a) median harvest density



(b) total harvest density

Figure 4.14: Model predictions versus validation responses for the multiple linear regression models of (a) median and (b) total harvest density containing the linear and quadratic terms selected by best subset selection.

the optimum value amongst 100 values of λ from the interval $[10^{-6}, 10^2]$ was found by 10-fold CV. The corresponding model coefficients are presented in this section.

The R code is given in Section A.4 on page 215.

4.4.2 Results

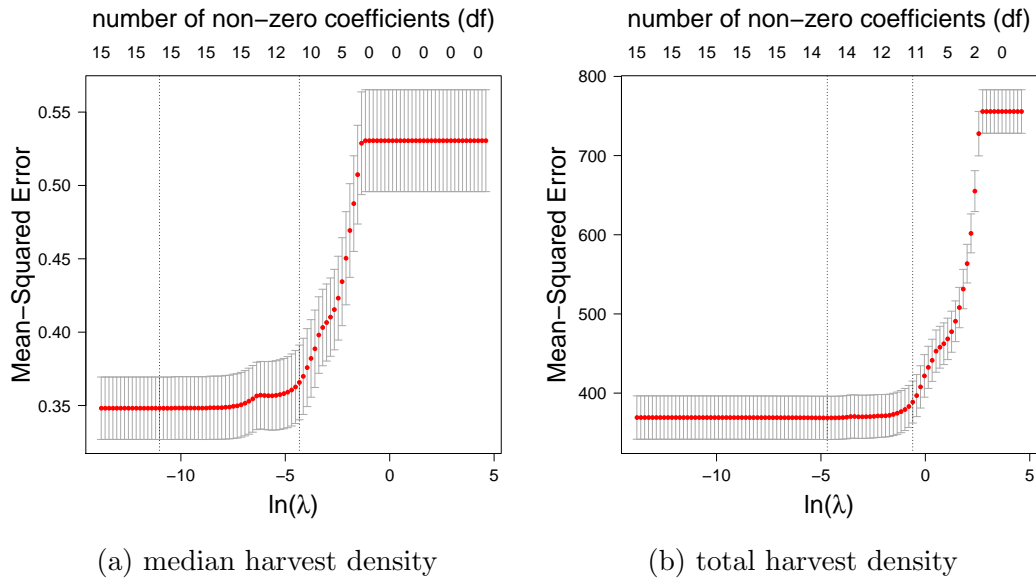


Figure 4.15: The 10-fold cross-validation MSE, along with standard error bars, for different values of $\ln(\lambda)$ to determine the optimum value of the complexity parameter λ for the lasso models of (a) median and (b) total harvest density. The leftmost grey dotted line in each plot marks the λ value $\lambda_{CV_{\min}}$ that resulted in the minimum mean CV error, while the rightmost line marks the λ value $\lambda_{CV_{1se}}$ that resulted in a mean CV error just less than 1 standard error greater than that of $\lambda_{CV_{\min}}$. The value $\lambda = \lambda_{CV_{\min}}$ was chosen in each case.

Figure 4.15 shows the CV MSE values for different values of λ , and Figures 4.16 to 4.17 on the next page display the paths followed by the individual coefficients over the same λ values for the two responses. Notice that, although all of the coefficients eventually shrink towards zero as λ increases, some of the coefficients do not shrink monotonically to zero. This is because the optimisation criterion (2.4.14) on page 32 only ensures that the *sum of the absolute values of the coefficients* decreases with an increase in λ ; no such constraint is placed on the *individual* coefficients. Since the input variables are used collectively to predict the response, correlations amongst the input variables influence the relative importance of the different model terms, with model terms that are highly correlated with one another influencing one another to a greater extent than weakly correlated input variables (Hastie *et al.*, 2009).

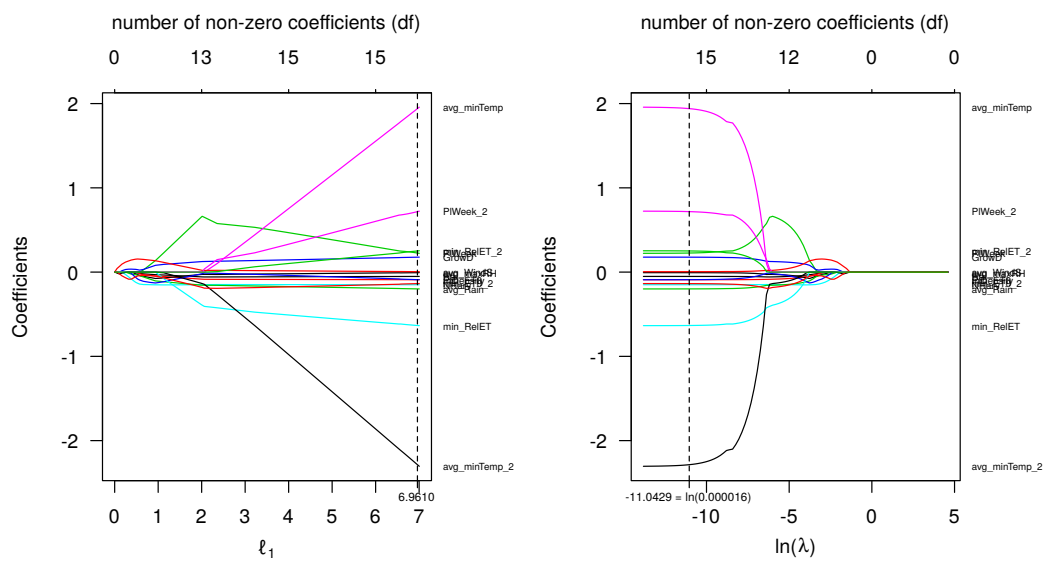


Figure 4.16: Coefficient values for the eleven linear and four quadratic terms over the complexity parameter values $\lambda = 10^{-6}$ to $\lambda = 10^2$ for the lasso model of median harvest density. The left-hand plot shows the coefficient paths as a function of the ℓ_1 norm, where $\ell_1 = \sum_{j=1}^p |\beta_j|$, while the right-hand plot shows the same coefficient paths against $\ln(\lambda)$. The optimum complexity parameter value is indicated in each plot.

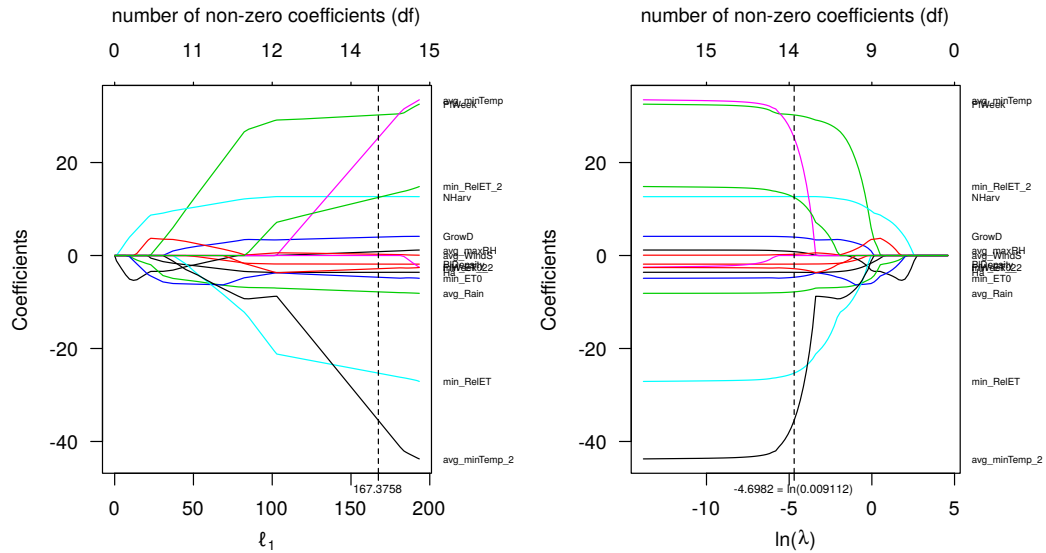


Figure 4.17: Coefficient values for the eleven linear and four quadratic terms over the complexity parameter values $\lambda = 10^{-6}$ to $\lambda = 10^2$ for the lasso model of total harvest density. The optimum complexity parameter value is indicated in each plot.

This is evident in Figures 4.16 to 4.17 on the previous page. For example, the correlation between `P1Week` and `P1Week_2` i.e. $(\text{P1Week})^2$, is $r = 0.9873$. The plots in Figure 4.16 show that the coefficient of `P1Week` (green curve) increases quickly, until `P1Week_2` (pink curve) enters the model at $\ell_1 \approx 2$ and $\ln(\lambda) \approx -5.5$, at which point `P1Week`'s coefficient steadily decreases as that of `P1Week_2` increases. Another example is that of `avg_minTemp` and `min_Re1ET_2` i.e. $(\text{min_Re1ET})^2$, which share a correlation of $r = 0.8760$. Figure 4.17 shows that `min_Re1ET_2` (green curve) increases until `avg_minTemp` (pink curve) enters the model at $\ell_1 \approx 105$ and $\ln(\lambda) \approx -3$, after which `min_Re1ET_2` continues to increase, but at a slower rate.

For median harvest density, $\lambda = \lambda_{\text{CV}_{\min}} = 1.6 \times 10^{-5}$ minimised the optimisation criterion (2.4.14) to (2.4.15) on page 32, while $\lambda = \lambda_{\text{CV}_{\min}} = 9.1 \times 10^{-3}$ was found to be optimum for total harvest density. Hence, both the median and total harvest density MLR models containing the linear and quadratic terms appear to have relatively little variance, requiring only a little shrinkage of the MLR coefficient estimators to reduce their variances. This is, to a certain extent, evident from the model coefficients (see Table 4.3 on page 139 and Table 4.4 on page 141). Most of the model coefficients in Tables 4.3 and 4.4 are significant, indicating that their coefficient values are large in comparison to their standard errors.

The resulting model coefficients are listed in Table 4.7 on the next page. Although the lasso performs variable selection, the small values of the selected complexity parameters for the two responses resulted in the algorithm retaining all of the input variables in the median harvest density model, and all but one of the input variables in the total harvest density model. The linear and the quadratic terms are therefore all important for modelling harvest quantity based on the assumptions made by the lasso algorithm.

A comparison of the lasso model coefficients in Table 4.7 with the corresponding MLR models—Table 4.3 on page 139 for median harvest density and Table 4.4 on page 141 for total harvest density—shows similarities between the median harvest density lasso and MLR models, but differences between the corresponding total harvest density models. This is due to the λ values selected for the different responses. For median harvest density, the very small value $\lambda = 1.6 \times 10^{-5}$ resulted in only modest shrinkage of the MLR coefficients, and most of the lasso coefficients are similar to, although closer to zero than,

Table 4.7: Coefficients for the lasso models of median ($\lambda = 0.000016$) and total ($\lambda = 0.009112$) harvest density containing the linear and quadratic terms. Terms that were excluded by the lasso algorithm are left blank.

	Harvest density	
	median	total
Intercept	2.31	91.29
Ha	-0.05	-3.62
PIDensity13300	-0.09	-1.85
PIWeek	0.23	30.24
GrowD	0.18	3.96
NHarv	-0.15	12.68
avg_minTemp	1.94	25.35
avg_maxRH	-0.01	0.83
avg_WindS	0.01	0.27
avg_Rain	-0.20	-7.82
min_ET0	-0.09	-4.67
min_RelET	-0.63	-25.30
PIWeek_2	0.72	
avg_minTemp_2	-2.29	-35.47
min_ET0_2	-0.14	-2.84
min_RelET_2	0.25	12.50

those in the MLR model. However, the relatively large $\lambda = 9.1 \times 10^{-3}$ selected for the total harvest density lasso model resulted in a greater shrinkage of the MLR model terms, and the lasso and MLR coefficients are therefore quite different from each other.

Figure 4.18 on the following page provides the prediction errors obtained from the validation set for the resulting median ($\text{MSE}_{\text{validate}} = 0.3843$) and total ($\text{MSE}_{\text{validate}} = 352.89$) harvest density lasso models. The validation errors obtained by the lasso models are very similar to those obtained by the MLR models including the eleven linear and four quadratic terms. The validation errors for all of the models covered so far in this chapter are displayed in Table 4.8 on page 152.

Ridge regression, which shrinks the sum of the *squared* coefficient values towards zero, was also used to model median and total harvest density. The R code is provided in Section A.4 on page 215, along with that of the lasso. The ridge regression algorithm is therefore also a regularisation technique. However, unlike the lasso, none of the model coefficients are shrunk to exactly zero, and ridge regression therefore does not perform variable selection.

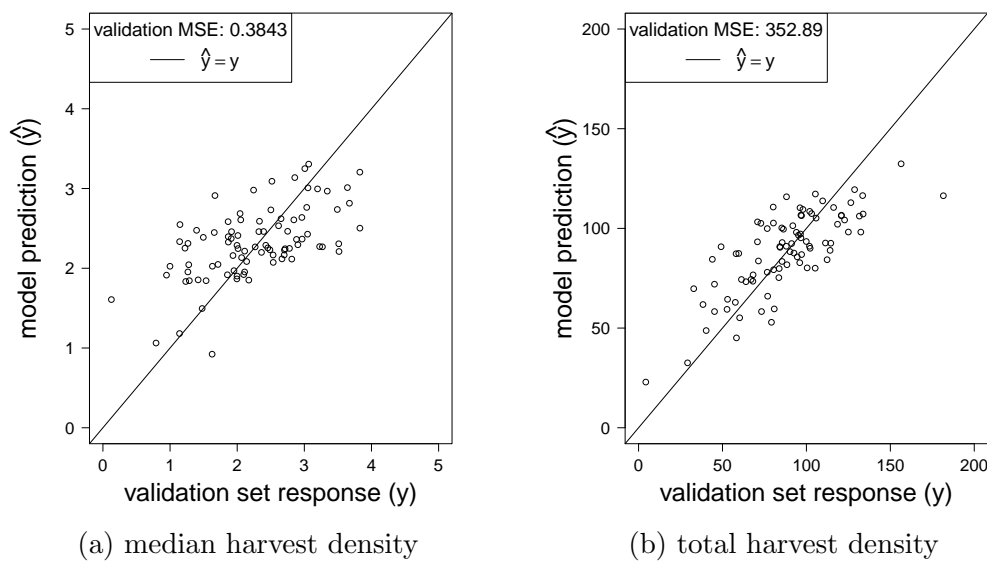


Figure 4.18: Model predictions versus validation responses for the lasso models of (a) median and (b) total harvest density containing the linear and quadratic terms.

The ridge regression algorithm yielded results very similar to those of the lasso, with $\lambda = \lambda_{CV_{\min}} = 1.6 \times 10^{-5}$ obtained for median and $\lambda = \lambda_{CV_{\min}} = 1.3 \times 10^{-2}$ obtained for total harvest density. The resulting model coefficients (not shown due to space constraints) are also similar to those of the lasso. The validation errors obtained by the two ridge regression models are therefore also similar to those of the lasso: $MSE_{\text{validate}} = 0.3844$ for median and $MSE_{\text{validate}} = 353.23$ for total harvest density. Consequently, the lasso provides slightly more accurate models of harvest quantity than does ridge regression.

4.5 Discussion

Table 4.8 on the next page lists the validation errors obtained from the modelling techniques explored in this chapter. The linear and regularised models constitute considerable improvements over the null model. The input variables are therefore useful for modelling both responses. The lasso achieved the best validation error for median harvest density, while the linear regression model containing both linear and quadratic input variables attained the lowest validation error for total harvest density.

Of the three MLR models implemented in this chapter, the model contain-

Table 4.8: Validation errors of linear models for predicting total and median harvest density using the input variables in Table 3.14 on page 124. The models contain the eleven linear terms (lin.), the eleven linear terms as well as the four quadratic input variables (lin. & quad.) or the linear and quadratic terms chosen by BSS. The MSE obtained on the validation set is given for each model, as well as the percentage improvement of each model's validation error over that of the null model. The lowest validation error obtained for each response is highlighted.

Model	Terms	Median harvest		Total harvest	
		MSE	%	MSE	%
null model	-	0.5912	-	846.40	-
linear regression	lin.	0.3945	33.27	369.07	56.39
linear regression	lin. & quad.	0.3850	34.88	352.35	58.37
linear regression	BSS	0.3953	33.13	362.36	57.19
lasso	lin. & quad.	0.3843	35.00	352.89	58.37

ing all eleven of the linear terms and all four of the quadratic terms attained the best prediction errors on the validation data for both median and total harvest density. Moreover, the lasso models, which were fit to the eleven linear terms and the four quadratic terms, were optimised by relatively small complexity parameters, indicating that the algorithm only shrunk the MLR coefficients a little to reach the lasso model for each response. Tomato plants are sensitive to the climatic conditions under which they are grown (see Section 2.1 on page 6), and it is therefore highly likely that each of the variables input into the model influences yield. However, variable importance is not enough; a sufficient amount of data (a sufficient number of observations) are needed for the algorithm to determine the relative influences of the input variables on the response. Modelling occurs in the input-output space, in which the n training observations occur in a space spanned by the input variables and the response. The height of the response surface in a region is estimated from the training responses in that part of the input space. The lower the density of the training observations, the less information will be available to the algorithm for estimating the height of the response surface. This will make the model more vulnerable to noise in the data, thereby increasing the variance of the model coefficient estimators. Thus, the lower the density of the training observations, the less accurately the algorithm is likely to approximate the response surface

in that region. This is exacerbated by multicollinearity amongst the input variables, which makes it even more difficult for the algorithm to distinguish the individual influences of the (highly similar) input variables on the response. Performing variable selection often helps to increase the generalisation error by removing relatively uninformative and/or highly correlated input variables, thereby increasing the density of the training observations in the input-output space and making the remaining input variables less similar to one another. However, if the density of observations is already high, then increasing the density by removing input variables will lead to only a mediocre increase in model accuracy, and overall model accuracy will decrease as a result of the absence of informative input variables from the model. Since the most detailed MLR model achieves the lowest validation errors for both responses in Table 4.8 on the previous page and since the MLR coefficients required relatively little shrinkage to arrive at the lasso model in each case, there appears to be enough training observations in the current study to enable the incorporation of all of the input variables, including highly correlated ones, e.g. $r(\text{PlWeek}, \text{PlWeek}_2) = 0.9873$, into the relatively inflexible models described in this chapter.

The above argument also provides an explanation for why fitting the lasso and ridge regression models did not lead to significant improvements in the validation errors of the two responses in Table 4.8. Model coefficient estimators are less likely to have high variances in more data-rich scenarios. Since regularisation techniques such as ridge regression and the lasso use (slightly) biased estimators in order to counter high variance in the model coefficient estimators resulting from a lack of data, these modelling techniques only achieve modest reductions (or even an increase) in prediction error here.

Interestingly enough, the modelling techniques model total harvest density more accurately than they do median harvest density, in that the best total harvest density model achieves a $\pm 58\%$ improvement over its null model, whereas the best median harvest density model attains a 35% improvement over its null model. This trend will be encountered in the next two chapters as well, and an explanation for this is offered at the end of Chapter 6.

This chapter involves the application of relatively rigid techniques to the data. However, the relationship between the input variables and harvest density is unlikely to be linear, and the linear models and models based on the

MLR coefficients are only expected to provide very rough approximations of harvest density. The next two chapters cover the application of more flexible, tree-based methods to modelling harvest quantity. These methods will allow relationships amongst the input variables and response to be investigated from different points of view using different model assumptions.

Chapter 5

Regression trees and tree-based methods

5.1 Introduction

The previous chapter involves the application of multiple linear regression and the lasso to the modelling of harvest density. This chapter, on the other hand, covers the implementation of tree-based techniques. Decision trees have a number of advantages over the linear model: They take both quantitative and categorical variables as inputs without the need for dummy variables, they are insensitive to monotone predictor transformations and are robust to input variable outliers in the training set, they accommodate nonlinear relationships in their structure and model interactions amongst input variables naturally, they perform variable selection automatically, and small trees are easy to interpret (perhaps more so than linear methods).

The next section covers the modelling of harvest density with regression trees, followed by bagged regression trees. The chapter concludes with the random forest modelling technique. Since decision trees can model nonlinear relationships, the eleven linear terms in Table 3.14 on page 124 are used as inputs into the models covered in this chapter i.e. the four quadratic terms in Table 3.14 are not used.

5.2 Regression trees

5.2.1 Methods

Regression trees of median and total harvest density were constructed using the eleven linear terms in Table 3.14 on page 124. Most of the parameters in the function call were left at their default values (see Section 2.4.4.5 on page 39). However, the default complexity parameter setting of $cp = 0.01$ has been found to be too large in the case of some large data sets (Therneau and Atkinson, 2015), and this parameter was optimised in this study using the `caret` package in R. Complexity parameter values ranging from 0.00001 to 0.01 were tested using 10-fold CV on the training set, and the value that minimised the root mean squared error (RMSE) was selected for each response. The trees were modelled in R with the `rpart` function's `anova` method in the `rpart` package, and the resulting trees were plotted using the `rpart.plot` package (Milborrow, 2016b).

The R code for finding the optimum cost parameter and for building the full and pruned regression trees is given in Section A.5 on page 218.

5.2.2 Results

5.2.2.1 Complexity parameters of the tree models

The setting $cp = 0.0005$ was found to be optimal for median and $cp = 0.001$ for total harvest density (Table 5.1 on the following page and Figure 5.1 on the next page). The responses were therefore modelled using these two values.

5.2.2.2 Full regression trees

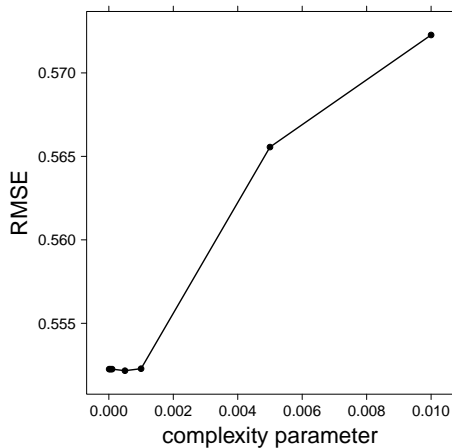
Figures 5.2 to 5.7 on pages 158–168 display the results of modelling median and total harvest density with regression trees.

Figure 5.2 on page 158 shows the regression tree of median harvest density (`MedTonPerHa`). The tree contains 47 splits (internal nodes), with the terminal nodes each containing 7 to 25 training observations. The decision tree algorithm incorporated all eleven of the input variables in the model structure. These input variables have correlations of:

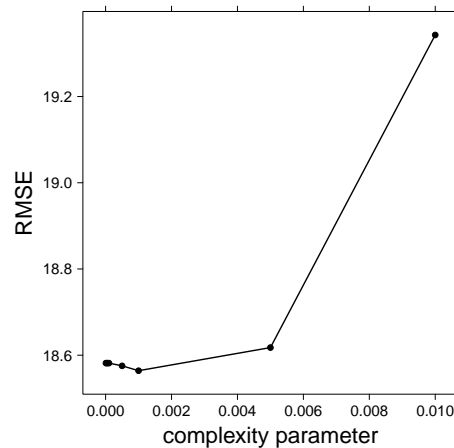
- $r(\text{avg_WindS}, \text{MedTonPerHa}) = 0.36,$

Table 5.1: The 10-fold CV errors in terms of the RMSE of regression trees for predicting median and total harvest density with cost parameter values ranging from 0.00001 to 0.01 specified. For each response, the lowest CV error obtained for the trees is highlighted. These values are plotted in Figure 5.1.

Cost parameter	Median harvest RMSE	Total harvest RMSE
1×10^{-5}	0.5522550	18.58164
5×10^{-5}	0.5522550	18.58164
1×10^{-4}	0.5522550	18.58164
5×10^{-4}	0.5521622	18.57533
1×10^{-3}	0.5522825	18.56404
5×10^{-3}	0.5655571	18.61765
1×10^{-2}	0.5722720	19.34282



(a) median harvest density



(b) total harvest density

Figure 5.1: The 10-fold CV errors in terms of the RMSE achieved on different complexity parameter settings for the (a) median and (b) total harvest density regression trees. These values are given in Table 5.1.

- $r(\text{avg_Rain}, \text{MedTonPerHa}) = -0.34$,
- $r(\text{min_RelET}, \text{MedTonPerHa}) = -0.34$,
- $r(\text{avg_minTemp}, \text{MedTonPerHa}) = -0.33$,
- $r(\text{avg_maxRH}, \text{MedTonPerHa}) = -0.33$,
- $r(\text{GrowD}, \text{MedTonPerHa}) = 0.30$,

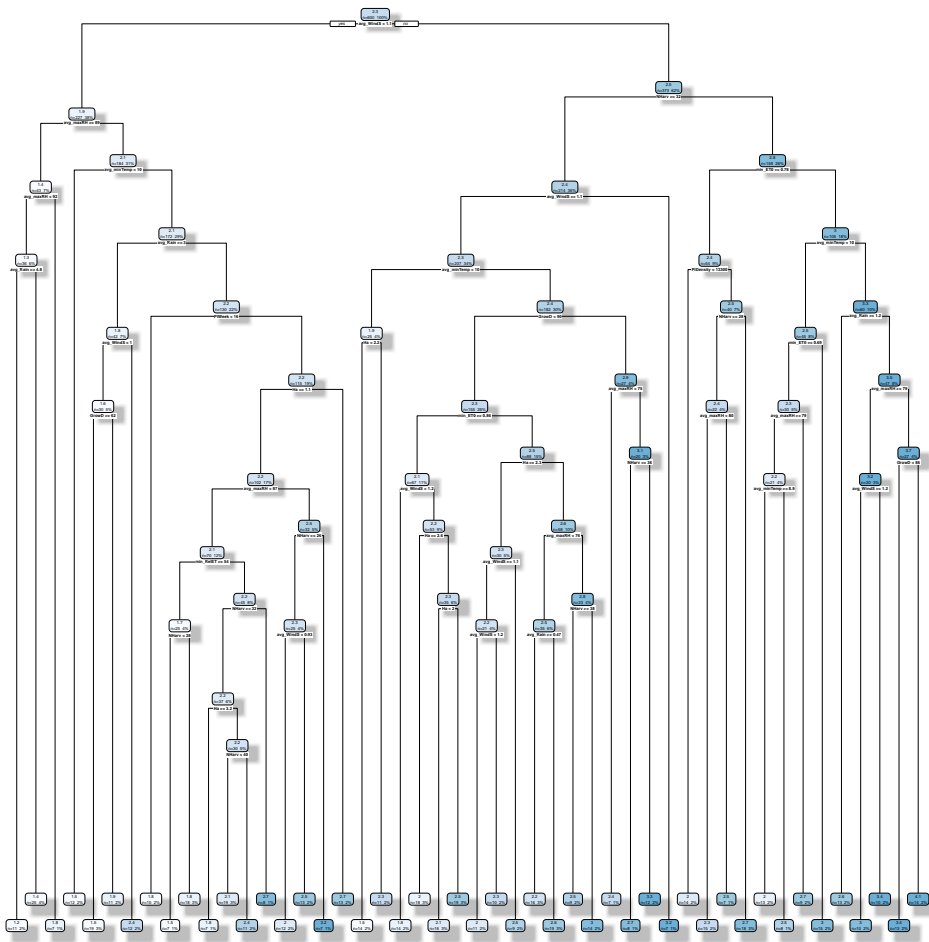


Figure 5.2: Regression tree of median harvest density based on the crop and weather variables in the training data set. Each node shows the predicted response for observations emerging in that region of the input space (top value), as well as the number and percentage of training observations occurring in that region (bottom values). The nodes associated with larger prediction values are coloured darker.

- $r(\text{min_ETO}, \text{MedTonPerHa}) = -0.30$,
- $r(\text{PlWeek}, \text{MedTonPerHa}) = -0.21$,
- $r(\text{Ha}, \text{MedTonPerHa}) = -0.18$,
- $r(\text{NHarv}, \text{MedTonPerHa}) = -0.17$, and
- $r(\text{PlDensity}, \text{MedTonPerHa}) = -0.07$

with the response (see Table 3.13 on page 123). The decision tree algorithm split first on `avg_WindS`—the input with the strongest correlation with `MedTonPerHa`. Notice that area (`Ha`) is informative for predicting harvest density.

The region in Figure 5.2 on the preceding page with the lowest prediction for `MedTonPerHa` i.e. 1.2 t/ha, is characterised by block crops with low average daily average wind speeds ($\text{avg_WindS} < 1.1$), high average daily maximum RH values ($89 \leq \text{avg_maxRH} < 92$) and high daily total rainfall ($\text{avg_Rain} \geq 4.8$) over their growing periods. Referring back to Figure 3.41c on page 101 and Figures 3.42a to 3.42b on page 102 reveals that, since both daily average wind speed and maximum RH have asymmetric distributions over the weeks of the year, these conditions are associated mostly with block crops planted at the end of the year, thereby having growing periods that span the first few weeks or months of the year.

In contrast to the region with the lowest median harvest density prediction, the region generating the largest prediction for the response i.e. 4.1 t/ha, is characterised by a combination of value ranges for seven of the input variables (Figure 5.2). More specifically, block crops with growing periods with high average daily average wind speeds ($\text{avg_WindS} \geq 1.1$), few harvest events ($\text{NHarv} < 32$), low minimum daily total ET_0 readings ($\text{min_ET0} < 0.78$), high average daily minimum temperatures ($\text{avg_minTemp} \geq 10$), low average daily total rainfall readings ($\text{avg_Rain} < 1.2$), low average daily maximum RH readings ($\text{avg_maxRH} < 79$), and long growing periods ($\text{GrowD} \geq 86$) receive the largest predicted median harvest density. Hence, block crops planted towards the middle of the year, thereby having growing periods that span late winter and early spring are most likely to emerge in the region with the largest prediction for median harvest density (see Figure 3.25 on page 86, Figures 3.41b to 3.41c on page 101 and Figures 3.42a to 3.42c on page 102). Note that, although the region generating the largest prediction is characterised by seven splits, the hierarchical structure of a decision tree means that the top few splits are the most important for understanding and predicting which block crops will produce the highest harvest densities.

The decision tree algorithm split on each input variable between 1 and 8 times (Figure 5.2 on the preceding page):

- `Ha` (6 times),

- `PlDensity` (once),
- `PlWeek` (once),
- `GrowD` (3 times),
- `NHarv` (8 times),
- `avg_minTemp` (4 times),
- `avg_maxRH` (8 times),
- `avg_WindS` (8 times),
- `avg_Rain` (4 times),
- `min_ET0` (3 times), and
- `min_ReLET` (once).

This provides an indication of possible interactions amongst the input variables. `NHarv`, `avg_maxRH` and `avg_WindS` were each split on the most number of times (8 times), suggesting that the range of values of each of these variables most conducive to producing any particular harvest density differs depending on the input variables higher up in the same branch of the tree. In contrast, `PlDensity`, `PlWeek` and `min_ReLET` (the latter input variable having the third strongest correlation with the response; see page 156) were only split on once each. This could be the result of very weak relationships existing between the input variables and the response, which appears to be the case for `PlDensity` (see Figure 3.21 on page 84 and Table 3.13 on page 123), or the input variables having simple relationships i.e. no interactions involved, with the response in the presence of the other input variables.

Figure 5.3 on the next page displays the regression tree of total harvest density (`TotTonPerHa`). The total harvest density model (Figure 5.3) is similar in size to the median harvest density model (Figure 5.2 on page 158), consisting of 45 splits, with the leaves containing 7 to 29 training observations. The algorithm included ten of the eleven input variables in the model, which have correlations of:

- $r(\text{avg_maxRH}, \text{TotTonPerHa}) = -0.51$,

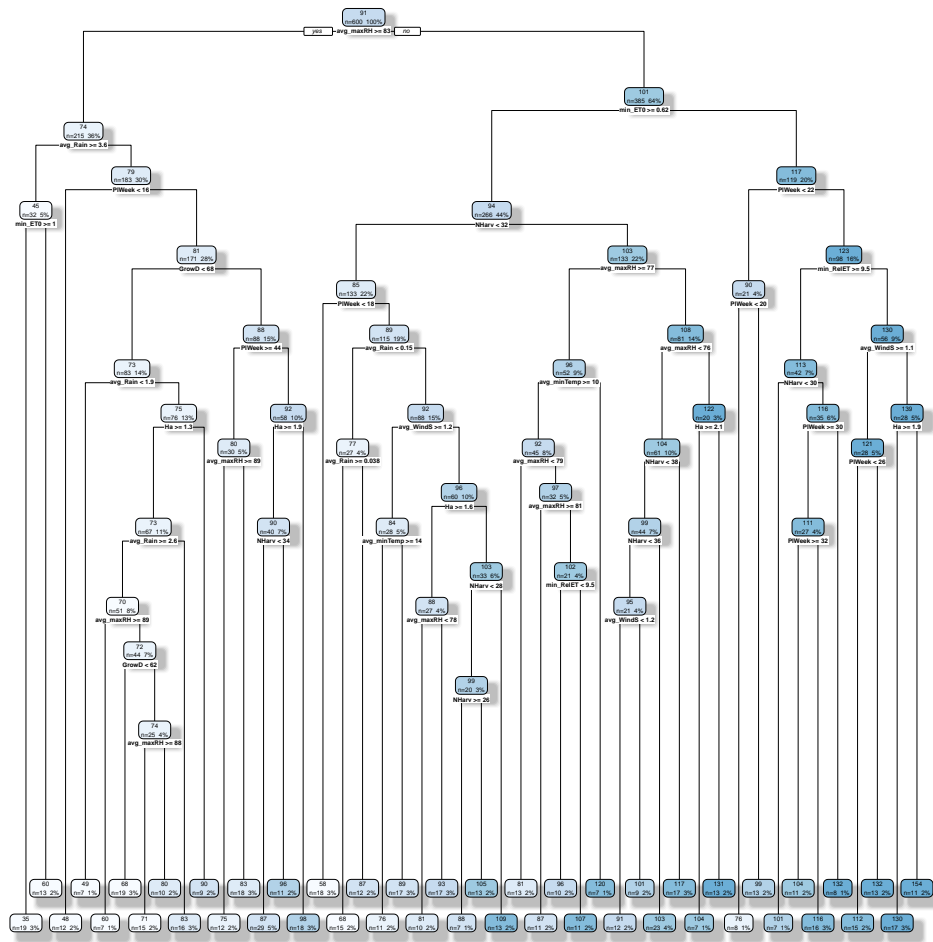


Figure 5.3: Regression tree of total harvest density based on the crop and weather variables in the training data set. The predicted response for observations in the region (top value) as well as the number and percentage of training observations in the region (bottom values) are given for each node. The nodes associated with larger prediction values are coloured darker.

- $r(\text{NHarv}, \text{TotTonPerHa}) = 0.48,$
- $r(\text{avg_Rain}, \text{TotTonPerHa}) = -0.47,$
- $r(\text{min_RelET}, \text{TotTonPerHa}) = -0.43,$
- $r(\text{avg_WindS}, \text{TotTonPerHa}) = 0.41,$
- $r(\text{avg_minTemp}, \text{TotTonPerHa}) = -0.38,$
- $r(\text{min_ET0}, \text{TotTonPerHa}) = -0.37,$
- $r(\text{PlWeek}, \text{TotTonPerHa}) = -0.26,$

- $r(\text{GrowD}, \text{TotTonPerHa}) = 0.19$, and
- $r(\text{Ha}, \text{TotTonPerHa}) = -0.07$

with the response (Table 3.13 on page 123). The algorithm therefore incorporated the eight input variables most strongly correlated with `TotTonPerHa`. Interestingly, planting density, which has a correlation of

- $r(\text{PlDensity}, \text{TotTonPerHa}) = 0.24$

with `TotTonPerHa` (Table 3.13), was omitted, while the less strongly correlated variables `GrowD` and `Ha` were included. This is, however, not surprising, since trees also incorporate nonlinear relationships and interactions amongst the input variables in their structure, and the strength of the correlation between an input variable and the response is therefore not the only factor that determines whether the input variable is included in the model. Notice that the input variables most strongly correlated with the response are not all involved in splits right at the top of the tree. For example, `NHarv`, which is the second most correlated with `TotTonPerHa` ($r = 0.48$) occurs below `min_ET0`, which has a far weaker correlation with `TotTonPerHa` ($r = -0.37$). The tree algorithm splits on the most informative input variable first. However, all of the splits lower down are nested within the regions higher up in the same branch, and interactions and correlations amongst the variables therefore also play a role in determining the usefulness of each input variable for each split. Consequently, a strong correlation does not necessarily mean that the input variable will occur high up in the tree.

The first split of the tree in Figure 5.3 on the preceding page is on average daily maximum RH—the input variable most strongly correlated with `TotTonPerHa` ($r = -0.51$). The tree model tends to predict larger total harvest density values for crops with lower average daily maximum RH over their growing periods, in agreement with the negative correlation between `avg_maxRH` and `TotTonPerHa`. The lowest prediction for total harvest density—35 t/ha—is allocated to those block crops that are exposed to high average daily maximum RH (`avg_maxRH` ≥ 83) and daily total rainfall (`avg_Rain` ≥ 3.6) values, and high minimum total ET_0 values (`min_ET0` ≥ 1) over their growing periods. Figure 3.41c on page 101 and Figures 3.42b to 3.42c on page 102 show that these conditions tend to be experienced by crops planted towards the end of the year or beginning of the following year.

In contrast, the region that assigns the highest total harvest density—154 t/ha—to observations is characterised by low average daily maximum RH ($\text{avg_maxRH} < 83$) and minimum total ET_0 ($\text{min_ET0} < 0.62$) readings, planting weeks later in the year ($\text{PlWeek} \geq 22$), and low minimum dekad RE ($\text{min_Re1ET} < 9.5$) and average daily wind speeds ($\text{avg_WindS} < 1.1$) over its block crops' growing periods. These conditions are associated with block crops with growing periods starting at around mid-year (Figure 3.41c on page 101 and Figures 3.42a and 3.42c to 3.42d on page 102). Strangely enough, as is the case with median harvest density (Figure 5.2 on page 158), area (Ha) also influences the predicted total harvest density (Figure 5.3 on page 161). It could be that the size of the block affects the quality of care given to the plants. However, inconsistencies during data collection resulted in larger harvest quantities sometimes being ascribed to the smallest blocks in the fields (see Section 3.1.3 on page 72), and this could also be contributing towards the importance of Ha in the models.

Thus, both the median and total harvest density models predict low harvest densities for crops planted at the beginning of the year, and high harvest densities for those planted in the middle of the year. The regression tree models therefore broadly reflect the patterns evident in Figures 3.23 to 3.24 on page 85.

The number of times that the algorithm split on each input variable in the total harvest density regression tree are:

- Ha (5 times),
- PlDensity (never),
- PlWeek (8 times),
- GrowD (twice),
- NHarv (7 times),
- avg_minTemp (twice),
- avg_maxRH (9 times),
- avg_WindS (3 times),
- avg_Rain (5 times),

- `min_ETO` (twice), and
- `min_ReLET` (twice).

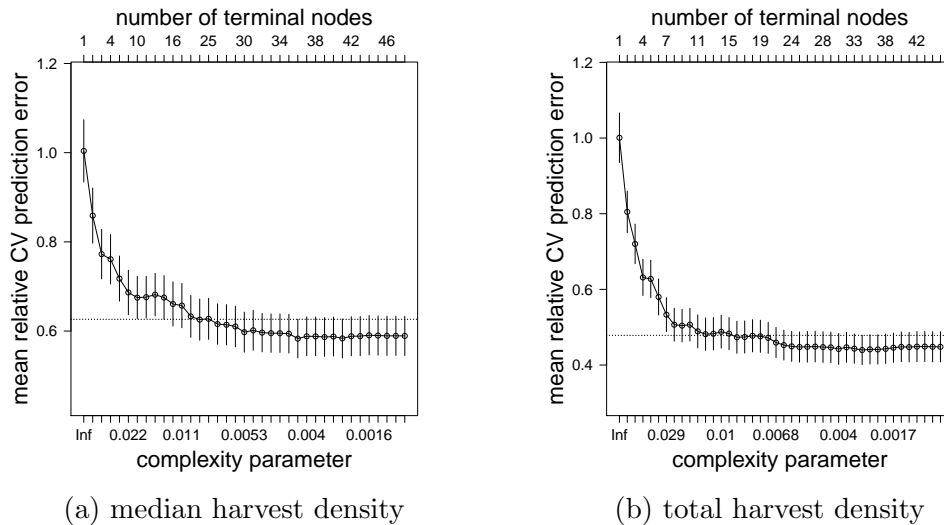
These numbers are quite similar to those found for the median harvest density regression tree model (see page 159). However, in contrast to the model in Figure 5.2, which mostly contains splits on different variables in the same branch, the total harvest density tree model in Figure 5.3 on page 161 contains quite a few instances of the same input variable involved in successive splits in the same branch. These include:

- `avg_Rain < 0.15` followed by `avg_Rain >= 0.038`,
- `NHarv < 28` followed by `NHarv >= 26`,
- `avg_maxRH < 79` followed by `avg_maxRH >= 81`,
- `avg_maxRH >= 77` followed by `avg_maxRH < 76`,
- `NHarv < 38` followed by `NHarv < 36`,
- `P1Week < 22` followed by `P1Week < 20`, and
- `P1Week >= 30` followed by `P1Week >= 32`.

Rather than indicating interactions, these nested splits are more a reflection that these input variables are very informative in the value ranges around the split values, and are therefore worth splitting several times.

5.2.2.3 Pruned regression trees

The regression tree algorithm does not attempt to split nodes containing fewer than a prespecified number of training observations, does not split a terminal node if it would result in either of the daughter nodes containing fewer than a certain number of training observations, and also does not carry out a split that only achieves a small improvement in the fit of the model to the training data. Moreover, the algorithm places a limit on the maximum depth of the tree (see Section 2.4.4.5 on page 39). These precautionary measures prevent the tree from fitting too closely to the observations in the training data set. However, these stopping rules are often not stringent enough, resulting in



(a) median harvest density

(b) total harvest density

Figure 5.4: The mean CV errors, along with standard error bars, achieved on the nested sequence of regression trees for (a) median and (b) total harvest density found by cost-complexity pruning. In each plot, the dotted line indicates one standard error above the minimum mean CV error obtained.

overfitting. Consequently, regression trees are usually pruned in an attempt to find a model that will achieve better generalisation errors.

Figure 5.4 shows the mean CV prediction errors resulting from performing 10-fold CV in order to find the optimum complexity tuning parameter. If one applies the *one-standard-error rule*, then the dotted line indicates a complexity parameter of 0.0076 for median (Figure 5.4a) and 0.0093 for total (Figure 5.4b) harvest density when building a pruned regression tree. Figures 5.5 to 5.6 on pages 166–167 show the resulting pruned trees. The pruned trees are substantially smaller than the full models, with the median harvest density tree containing 23 splits and the total harvest density tree containing 15 splits.

Figure 5.7 a–b on page 168 displays the prediction errors obtained by applying the full tree models (Figures 5.2 to 5.3 on pages 158–161) to the validation set, whereas Figure 5.7 c–d shows those obtained for the pruned trees (Figures 5.5 to 5.6 on pages 166–167). The full trees provide better models for harvest density than do the pruned trees, indicating that the full regression tree models do not overfit the training data. The stopping rules specified in the tree-building algorithm are therefore strict enough to prevent overfitting for this data set.

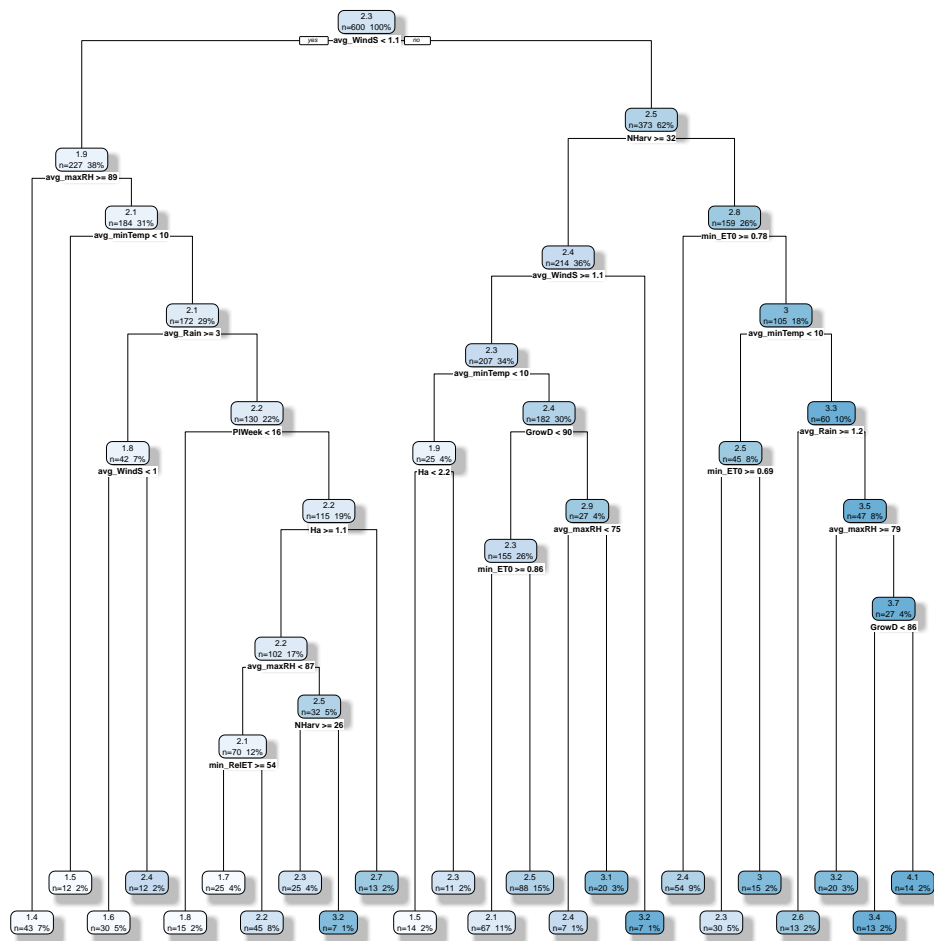


Figure 5.5: Regression tree of median harvest density from Figure 5.2 on page 158 after pruning.

5.3 Bagged regression trees

Bagging is a tree-based technique that involves averaging the predictions from numerous decision trees built from bootstrap samples of the training data set in order to produce prediction estimators with lower variance.

5.3.1 Methods

Median and total harvest density were each modelled with bagged regression trees based on the eleven linear terms in Table 3.14 on page 124 using the `randomForest` package in R. The minimum number of training observations allowed in the terminal nodes was chosen from the values 1, 3, 5 (the default setting for regression), 10, 15, 20 and 30 by CV using the `tune.randomForest`

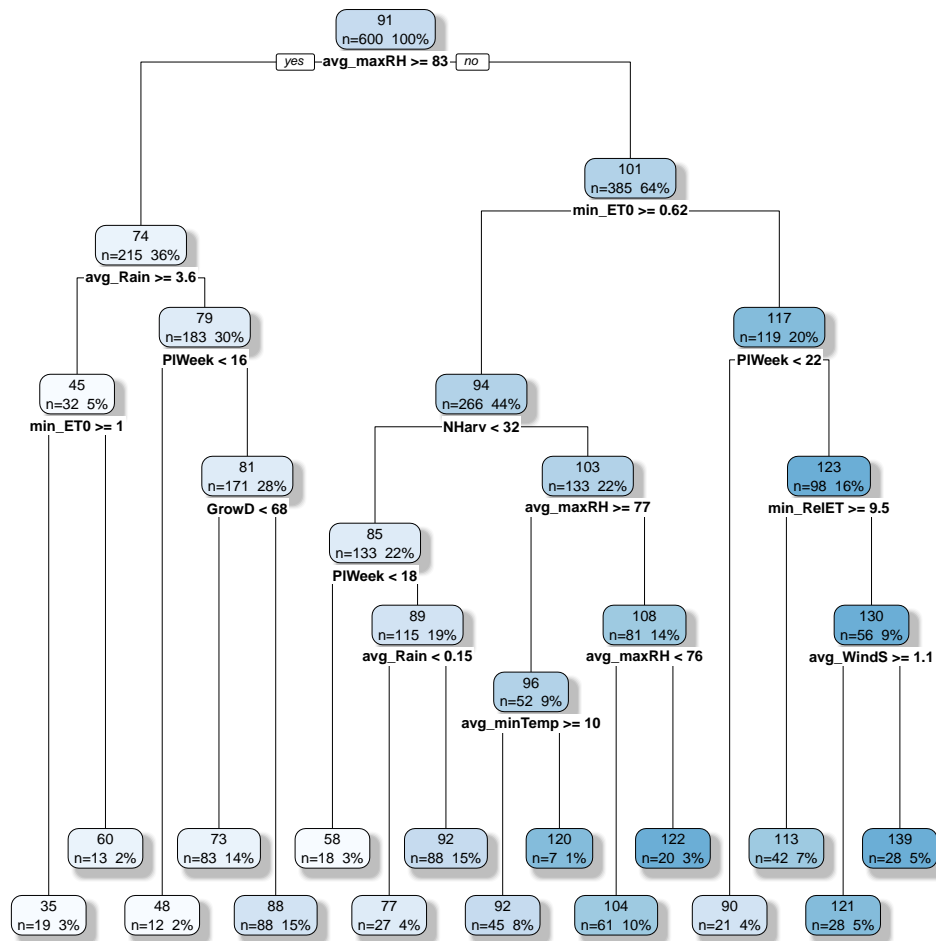


Figure 5.6: Regression trees of total harvest density from Figure 5.3 on page 161 after pruning.

function in the `e1071` package. The number of input variables considered for each split at each node was set to the total number of input variables i.e. `mtry=11`, and the importance of the input variables was calculated by setting `importance=TRUE`. Apart from these changes, the `randomForest` function was called with its default settings (see Section 2.4.5.3 on page 47). The main effects of the input variables on the predicted response were plotted.

The code for optimising the terminal node sizes and for training the bagged regression tree models is provided in Section A.6 on page 223.

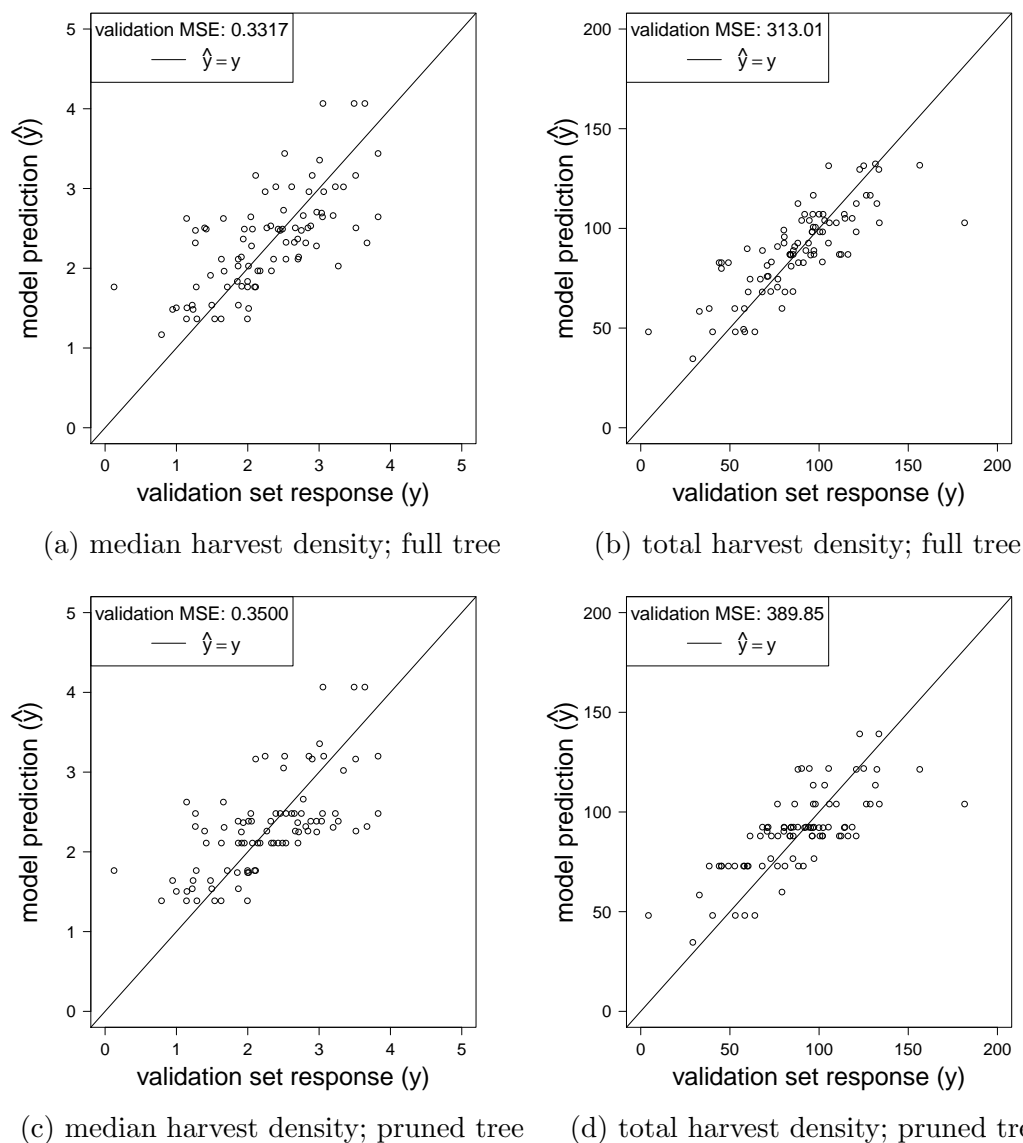


Figure 5.7: Model predictions versus validation responses for the full and pruned regression trees of median and total harvest density.

5.3.2 Results

The optimum minimum terminal node size selected by CV was 5 for median and 3 for total harvest density.

Figure 5.8 on the next page shows the OOB prediction errors as functions of the number of trees grown in the bagged models. For both the median and total harvest density models, the OOB error estimate starts off high when the model only contains a few deep BS regression trees, but quickly stabilises

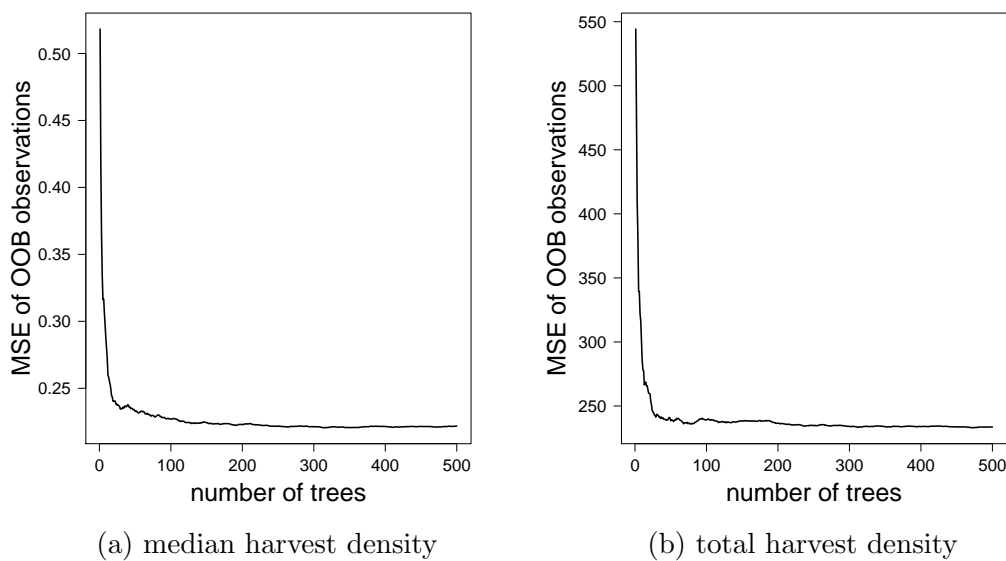


Figure 5.8: Prediction error in terms of MSE estimated using the OOB observations of each bootstrap sample for different numbers of trees included in the bagged regression tree model of (a) median and (b) total harvest density.

as more trees are added. The estimated error is stable even in the case of relatively small models for both responses, suggesting that the default setting of `ntree = 500` in the `randomForest` function does not need to be increased.

Figure 5.9 on the following page provides an indication of the importance of each input variable in the bagged regression tree models. The importance of the input variables according to the different measures are similar to each other in both of the models. As is the case in the median harvest density regression tree model (Figure 5.2 on page 158), the averages of the daily average wind speed values over the block crops' growing periods (`avg_WindS`) is the most important input variable in the bagged regression tree model of median harvest density according to both measures. However, input variable importance appears to differ in the bagged regression tree model of total harvest density, depending on how it is measured. More specifically, as is the case in the total harvest density regression tree model (Figure 5.3 on page 161), splits involving the averages of the daily maximum RH values over the block crops' growing periods (`avg_maxRH`) resulted in the largest decrease in the RSS_{train} . However, shuffling the minima of the daily total ET_0 over the block crops' growing periods (`min_ET0`) led to the largest increase in test error on the OOB observations. The minima of the dekad RE values (`min_Re1ET`) and the plant-

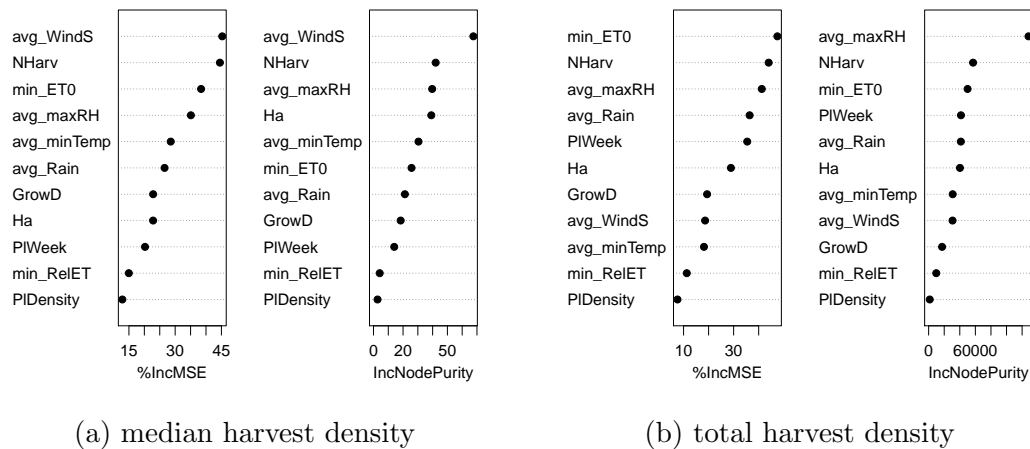


Figure 5.9: Importance of the input variables in the bagged regression tree models of (a) median and (b) total harvest density. Variable importance is assessed in two different ways, one being the normalised average decrease in prediction accuracy of the OOB observations as a result of shuffling the values of that variable (%IncMSE), and the other being the decrease in the training error resulting from splits of the input variable averaged over the trees (IncNodePurity).

ing density (PIDensity) are the least informative input variables in the bagged models.

Figures 5.10 to 5.11 on pages 171–172 display partial dependence plots of the bagged regression tree model of median and total harvest density, respectively. The input variables Ha, PIWeek, GrowD, avg_minTemp, avg_maxRH, avg_WindS, avg_Rain and min_ET0 share broadly the same trends between the two responses, whereas NHarv shows opposite tendencies. The variables GrowD, avg_minTemp and avg_WindS appear to be more informative for predicting median than total harvest density, while PIWeek and avg_Rain are more informative for total than median harvest density. The fitted response curves are more or less flat across the domains of PIDensity and min_RelET for both responses, providing graphical evidence that these are relatively uninformative input variables. Figures 5.10 and 5.11 suggest that harvest density tends to be largest during the time of year that has average maximum RH levels of 70 % to 90 %, average wind speeds of 1.1 m/s to 1.3 m/s, average rainfall levels below 3 mm/d and minimum ET_0 below 0.75 mm/d. A planting week in the middle or towards the end of the year and a longer growing period also increase harvest density. Notice that a small block crop area (Ha) also tends

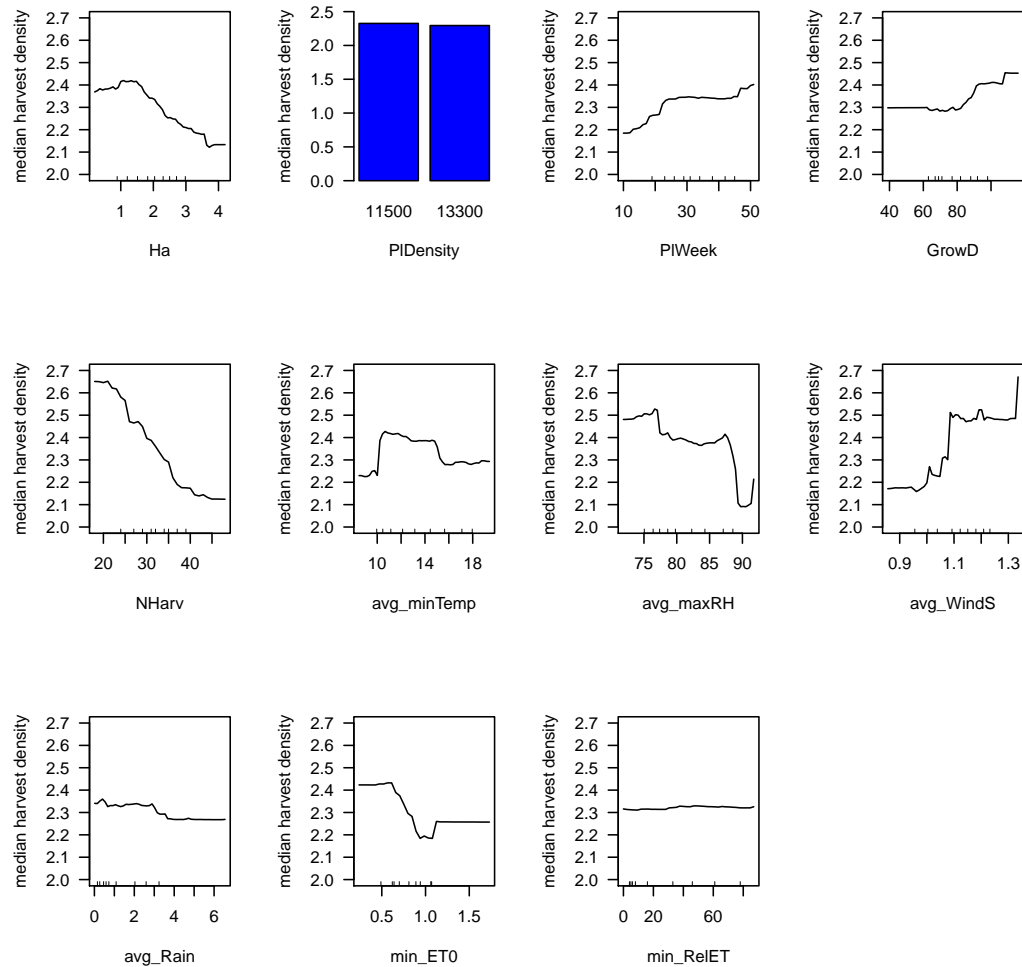


Figure 5.10: Partial dependence plots of the fitted response curve against each of the input variables in the bagged regression tree model of median harvest density after accounting for the average effects of the other input variables on the response. The rug at the base of each plot shows the distribution of the deciles of the response in the domain of that input variable.

to be associated with high harvest densities, suggesting that there might still be erroneous records present in the cleaned crop data set.

Note, however, that Figures 5.10 to 5.11 only show the effects of each input variable on the model after averaging over the values of the response for the other input variables. They therefore only provide an indication of the *marginal* effect of each input variable on the response. Consequently, input variables that play significant roles in interactions with other input variables appear less important in these plots than they actually are in the models.

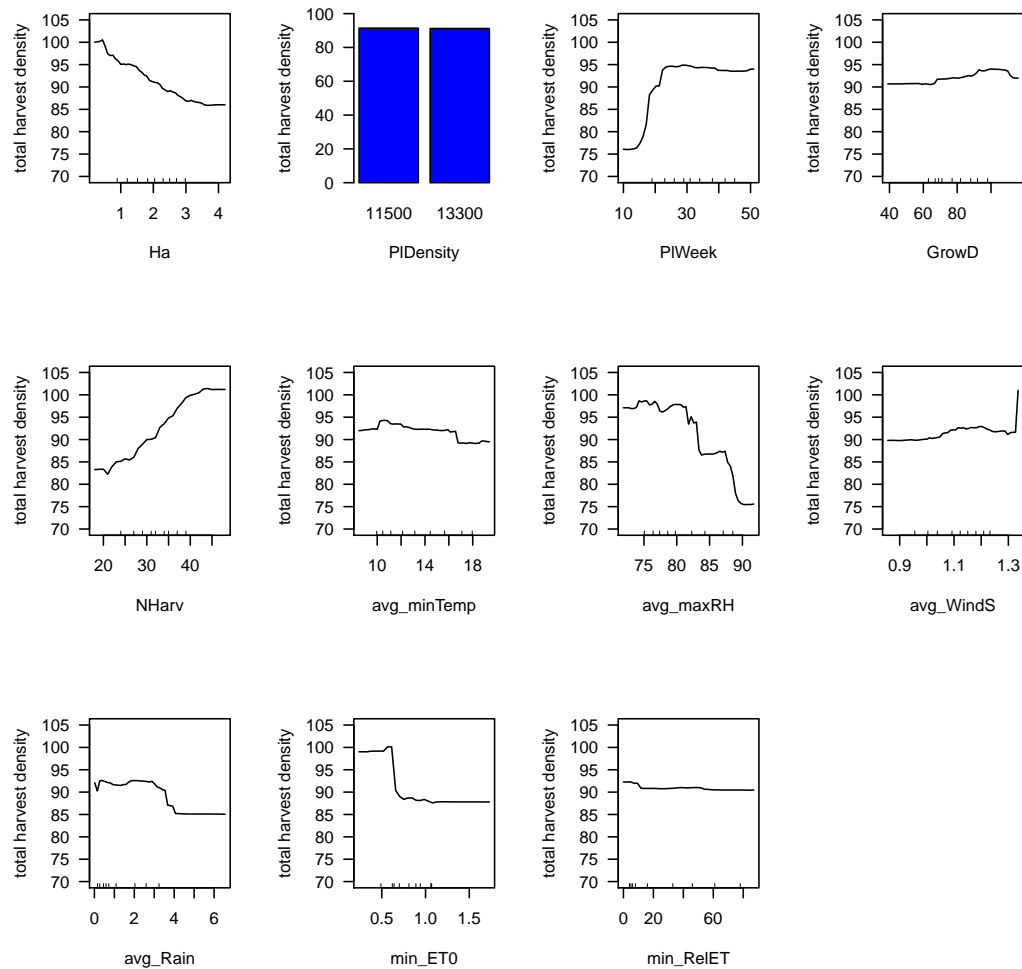


Figure 5.11: Partial dependence plots of the fitted response curve against each of the input variables in the bagged regression tree model of total harvest density after accounting for the average effects of the other input variables on the response. The rug at the base of each plot shows the distribution of the deciles of the response in the domain of that input variable.

For example, `avg_Rain` is ranked as a more important input variable in Figure 5.9a on page 170 than is either `GrowD` or `P1Week` in the median harvest density bagged regression tree model. However, in Figure 5.10 on the preceding page the response curve seems to vary more over the domains of `P1Week` and `GrowD` than it does over that of `avg_Rain`. This discrepancy between the relative informativeness of these three variables in Figure 5.9a and their relative influences on the response curve shown in Figure 5.10 suggests that `avg_Rain` is more heavily involved in interactions with other input variables

than are either **GrowD** or **PlWeek**.

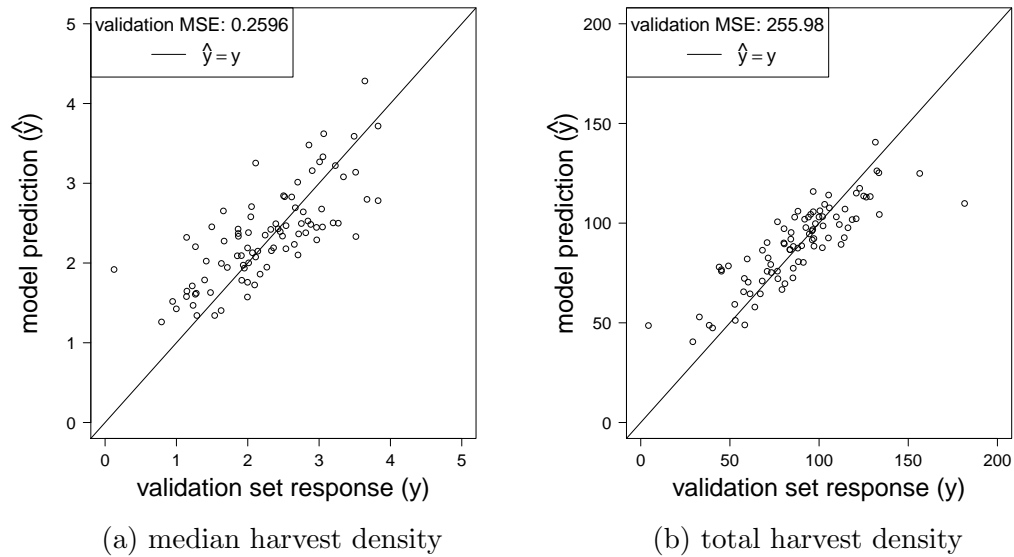


Figure 5.12: Model predictions versus validation responses for the bagged regression tree models of (a) median and (b) total harvest density.

The validation errors obtained from the bagged regression tree models (Figure 5.12) are substantial improvements on those obtained from the regression tree models (see Figure 5.7 on page 168). Consequently, as expected, bagging deeply-grown regression trees from this data set yields better models than do single regression trees.

5.4 Random forests

The RF algorithm attempts to reduce the variance of the model prediction estimators to an even greater extent than does the bagging algorithm. In addition to constructing numerous dissimilar trees from BS samples of the training data set (as is the case in bagging), the RF algorithm ensures that the trees are even more dissimilar to one another by restricting the number of candidate input variables for each split in each tree.

5.4.1 Methods

The RF models for median and total harvest density were also developed using the `randomForest` package in R. The minimum number of training observations in the terminal nodes and the number of input variables to be considered at each split were optimised using the `tune.randomForest` function in the `e1071` package. The importance of the input variables was calculated, and the main effects of the input variables on the predicted response were plotted.

The R code for selecting the optimum terminal node size and the optimum number of candidate input variables and for constructing the RF models is given in Section A.7 on page 226.

5.4.2 Results

The minimum number of training observations in the terminal nodes was 3 for median and 1 for total harvest density. The number of candidate input variables was 3 for median and 5 for total harvest density.

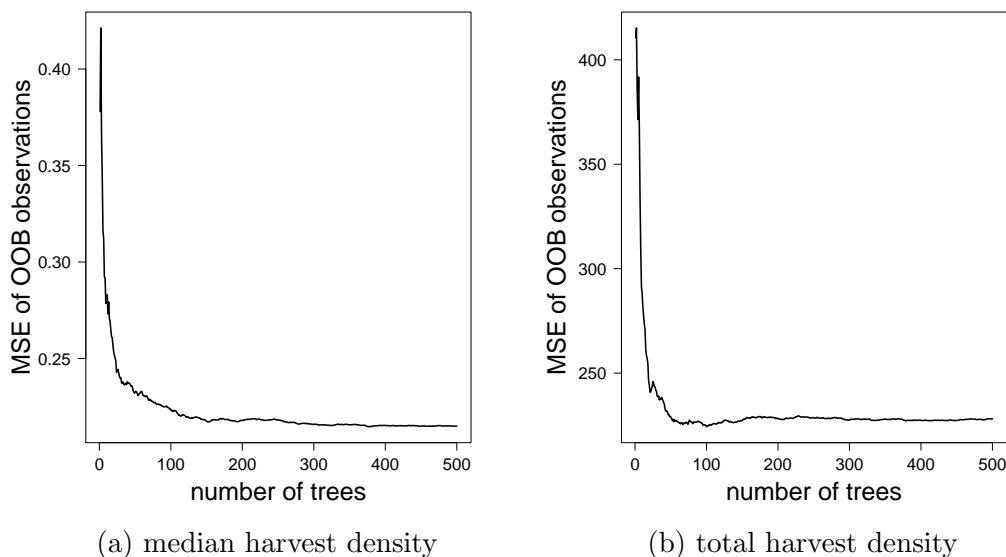


Figure 5.13: Prediction error in terms of MSE estimated using the OOB observations of each bootstrap sample for different numbers of trees included in the random forest model of (a) median and (b) total harvest density.

Figure 5.13 shows that, as is the case for the bagged regression tree models in the previous section, the default setting of `ntree = 500` in the `randomForest`

function seems to be adequate for this data set.

The importance of the input variables in the RF models (Figure 5.14) are similar to those obtained in the bagged regression tree models (Figure 5.9 on page 170), with `avg_WindS` emerging as the most important input variable in the median harvest density model according to both measures, and `avg_maxRH` emerging as the most important input variable for reducing the RSS_{train} in the total harvest density RF model. Once again, `min_RelET` and `PIDensity` are the least important input variables in both of the RF models.

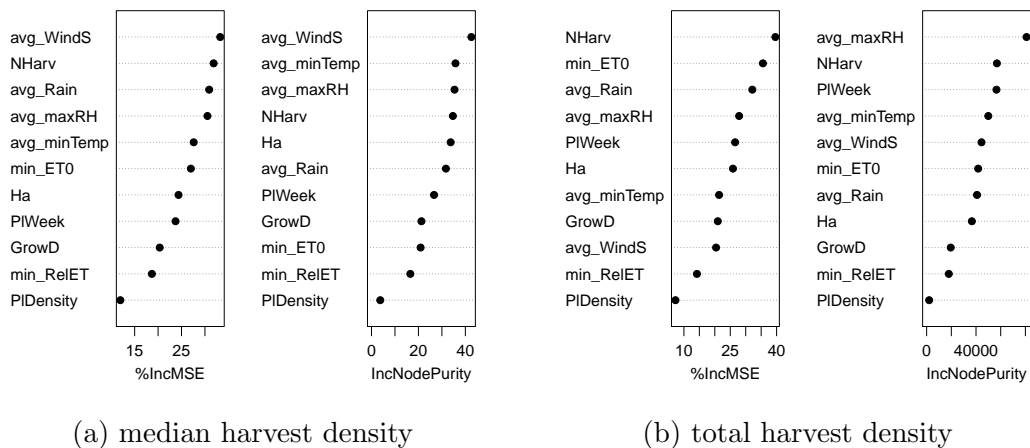


Figure 5.14: Importance of the input variables in the random forest models of (a) median and (b) total harvest density. Variable importance is measured by the effect of shuffling the input variable values on the OOB prediction error (%IncMSE), and by the improvement in the training error resulting from splits on the input variable (IncNodePurity).

Figures 5.15 to 5.16 on pages 176–177 show the partial dependence plots for the RF median and total harvest density models. They are very similar to their bagged regression trees counterparts in Figures 5.10 to 5.11 on pages 171–172.

Figure 5.17 on page 178 displays the validation errors achieved by the RF models. Interestingly enough, whereas the RF model achieved a lower validation error than did the bagged model for median harvest density (compare Figure 5.17a with Figure 5.12a on page 173), the opposite was the case for the total harvest density (compare Figure 5.17b with Figure 5.12b).

To investigate this surprising result further, Figure 5.18a on page 178 displays the validation errors of RF models based on different input variable subset sizes. It seems that, while the validation error for the median harvest

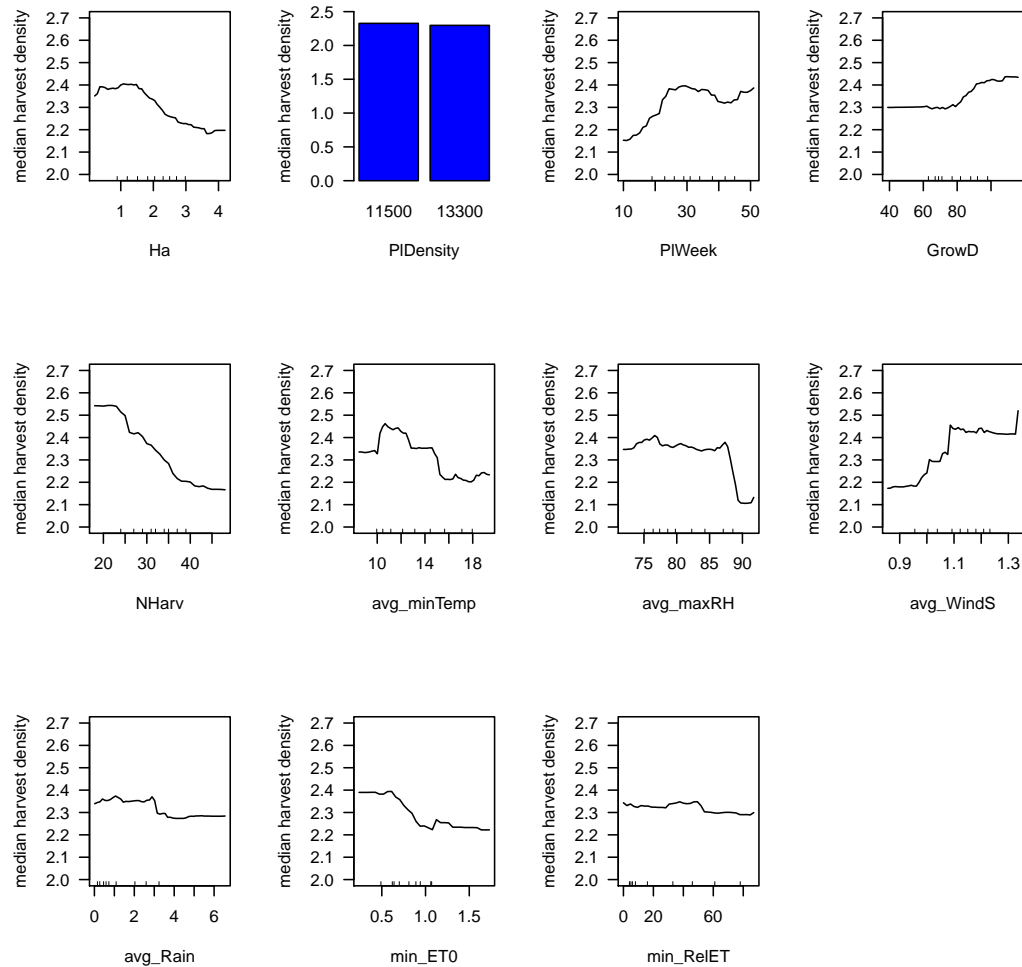


Figure 5.15: Partial dependence plots of the fitted response curve against each of the input variables in the random forest model of median harvest density after accounting for the average effects of the other input variables on the response. The rug at the base of each plot shows the distribution of the deciles of the response in the domain of that input variable.

density RF models tends to increase with the input variable subset size, the opposite is true in the case of the total harvest density RF models. A possible explanation for why the validation error for the total harvest density RF models climbs as the input variable subset size decreases is that small input variable subset sizes may result in only relatively uninformative input variables being available as candidates for many of the splits in the individual BS trees, thereby leading to more trees yielding poor predictions.

To determine whether the inclusion of uninformative input variables is caus-

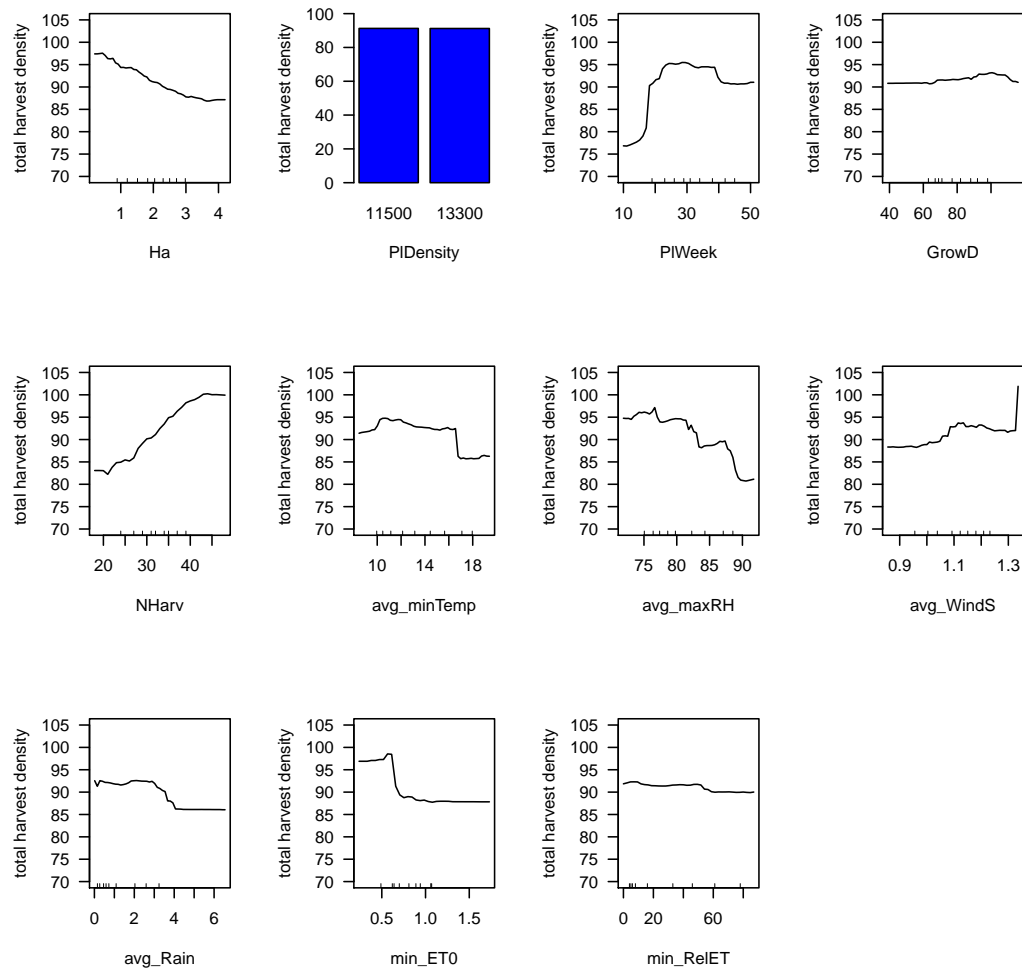


Figure 5.16: Partial dependence plots of the fitted response curve against each of the input variables in the random forest model of total harvest density after accounting for the average effects of the other input variables on the response. The rug at the base of each plot shows the distribution of the deciles of the response in the domain of that input variable.

ing the strange patterns in the validation errors of the total harvest density RF models, the three input variables that tend to emerge as being the least informative according to both measures in both models (median and total harvest density) i.e. `Pldensity`, `min_RelET` and `GrowD` (see Figure 5.14 on page 175), were excluded, and the validation errors were again determined for all possible input variable subset sizes (Figure 5.18b on the next page). Unfortunately, removing the least informative input variables from the RF models led to no change in the general trends of the validation errors for different input variable

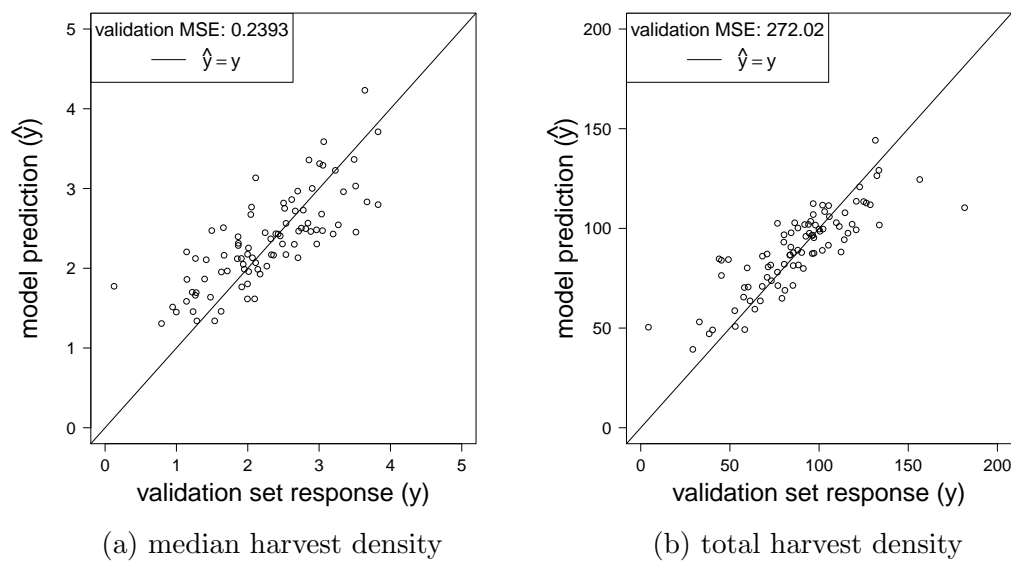


Figure 5.17: Model predictions versus validation responses for the random forest models of (a) median and (b) total harvest density.

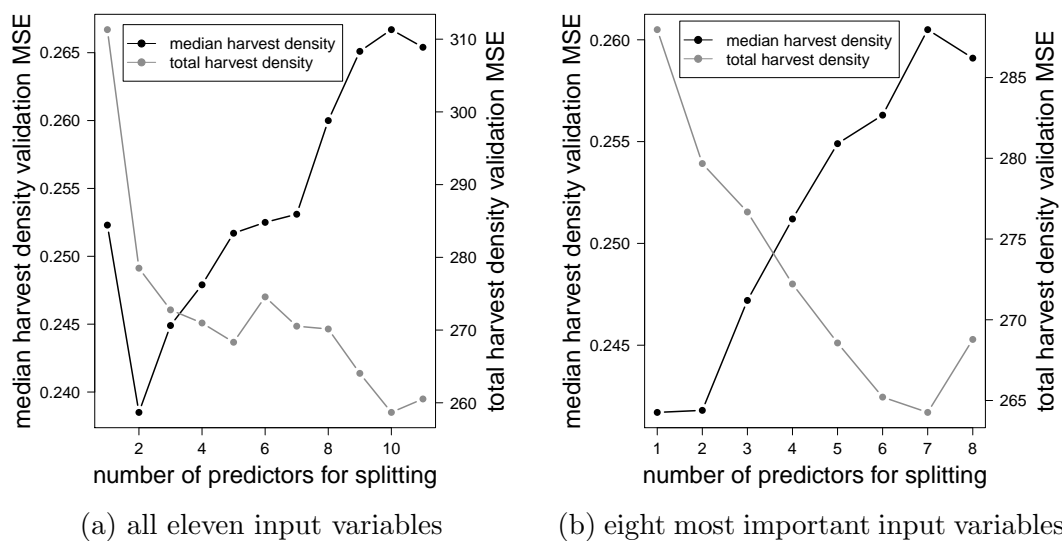


Figure 5.18: The validation errors in terms of the MSE achieved with different input variable subset sizes for the median and total harvest density random forest models containing (a) all of the input variables and (b) the eight most important input variables in the random forest models.

subset sizes. This is probably because less accurate predictions resulting from splits made using relatively uninformative input variables will only increase the variance of the prediction estimators; they will not affect the bias of the prediction estimators since the candidate input variables considered for each

split are chosen at random. The variance is then reduced by averaging the resulting predictions, thereby keeping the generalisation error low.

Table 5.2: Validation errors of linear and tree-based models for predicting total and median harvest density using the input variables in Table 3.14 on page 124. The models contain the eleven linear input variables (lin.) or the eleven linear and the four quadratic input variables (lin. & quad.). The MSE obtained on the validation set is given for each model, as well as the percentage improvement of each model's validation error over that of the null model. The lowest validation error obtained for each response is highlighted.

Model	Terms	Median harvest		Total harvest	
		MSE	%	MSE	%
null model	-	0.5912	-	846.40	-
linear regression	lin. & quad.	0.3850	34.88	352.35	58.37
lasso	lin. & quad.	0.3843	35.00	352.89	58.37
full regression tree	lin.	0.3317	43.89	313.01	63.02
pruned regression tree	lin.	0.3500	40.80	389.85	53.94
bagging	lin.	0.2596	56.09	255.98	69.76
random forests	lin.	0.2393	59.53	272.02	67.86

5.5 Discussion

Table 5.2 lists the validation errors obtained for the lasso and the best multiple linear regression model fit in the previous chapter, as well as the models fit in this chapter for median and total harvest density. The tree-based models of the current chapter achieved superior validation errors to the linear models and the lasso of the previous chapter, suggesting that the relationship between the crop and weather input variables and each of the harvest density responses is not approximated well by response surfaces based on the linear model.

As is usually the case with analyses based on real-world data, the regression trees achieved far worse validation errors than did bagging and RFs. Regression trees fit a piecewise-constant response surface to the data, which is an unrealistic model for most natural processes. Consequently, they tend to suffer from high bias if grown shallow (since they assign identical predictions to diverse observations) and high variance if grown deep (due to overfitting). Ensemble methods such as bagging and RFs also fit regression trees to the

data, but in such a way as to benefit from the many advantages of tree models (see Section 5.1 on page 155) while circumventing some of their disadvantages. These techniques fit deep trees to different parts of the training data set. Deep trees have the freedom to follow the patterns in the data closely, producing models with low bias. However, real data sets inevitably contain a lot of random noise, and the deep trees tend to suffer from high variance, which hinders their predictive power. To reduce the variance of the final model, predictions for observations are generated by averaging the predictions from the individual trees, which, to a large extent, negates the influence of the noise in the training data set on the model. Consequently, ensemble techniques are more or less guaranteed to produce better models than single regression trees on real, i.e. noisy, data sets.

Similarly to what was found in the previous chapter, it appears that there are enough training observations in this study to incorporate all of the input variables into the models—even the highly nonlinear models of this chapter. This is evident from the fact that the full regression tree performed considerably better on the validation set than did the pruned regression tree for both responses. This is, however, also supported by the fact that bagging achieved better validation errors than did RFs in some of the analyses. RFs was developed as a method to bring about an even greater reduction in variance than does bagging. The RF algorithm only considers a small number $m < p$ of the p input variables for any split in the bootstrap trees, thereby making the trees more different from one another. By taking the average of predictions from more diverse trees, the RF algorithm reduces the correlation amongst the prediction estimators, ρ_{Z^*} , thereby usually accomplishing an overall reduction in the variance of the final prediction estimators compared to that of bagging (see (2.4.22) on page 45). However, (2.4.22) shows that reducing the variance of the final prediction estimators is only brought about if the reduction in the correlation amongst the individual prediction estimators *outweighs* the increase in variance of the individual prediction estimators brought about by making the trees more dissimilar to one another. In other words, for the RF algorithm to construct a model with less variance overall, the decrease in ρ_{Z^*} must outweigh the increase in $\sigma_{Z^*}^2$.¹ This is why setting $m = 1$ in a

¹This can be compared with the strategy that shrinkage methods follow: The lasso and ridge regression algorithms trade small increases in the bias of their coefficient estimators in order to bring about substantial decreases in their variances. Bagging and RFs trade a

RF model does not necessarily lead to a model superior to that of bagging: $m = 1$ would result in highly decorrelated trees, but is also likely to cause the variance amongst the individual prediction estimators to explode.

The two data sets analysed in this study only differ in their responses (their input variables are identical), and yet RFs produced the better model for `MedTonPerHa` while bagging was the most successful for `TotTonPerHa`. The correlations of the response `MedTonPerHa` with the eleven input variables range from 0.07 to 0.36 in absolute value (see page 156), whereas the correlations between `TotTonPerHa` and the input variables range in absolute value between 0.07 and 0.51 (see page 160). Hence, the input variables are collectively far more informative for predicting `TotTonPerHa` than `MedTonPerHa` (if one uses correlation as an indication of informativeness). The lower overall variance due to the high n/p ratio means that RFs have less of an edge over bagging in this data set, making the bagging and RF models more comparable with each other for both responses. This is similar to what was found in the previous chapter, in which the lasso did not perform much better than the MLR model containing the same model terms due to the low overall variance in the data set caused by the high n/p ratio (Table 5.2 on page 179). The greater proportion of noise relative to signal in the `MedTonPerHa` data set makes it possible for the RF algorithm to achieve a better model than bagging. In contrast, the greater informativeness of the input variables for modelling `TotTonPerHa` means that there is less of a need for variance reduction, and bagging (which has more freedom to follow the patterns in the data) yields a superior model to the more restrictive RF algorithm.

The data set of the study appears to contain enough random noise to make single regression trees perform more poorly than bagging and random forests, but not enough random noise to make ridge regression and the lasso perform better than MLR, or to make RFs perform better than bagging for all data sets. The relatively strong signal in this data set is probably largely due to the large number of observations relative to the number of input variables.

The next chapter involves modelling harvest density using boosted regression. A small increase in the variances of their individual prediction estimators in an attempt to substantially reduce their correlations, and therefore the variance of the final model. Both strategies therefore involve restricting the model's ability to learn from the data in order to reduce the influence of the noise in the training data set on the final model.

sion trees, a modelling techniques that allows even more complex relationships to exist amongst the variables than do regression trees, bagged regression trees and RFs.

Chapter 6

An adaptive method: Boosted regression trees

6.1 Introduction

As is the case with bagging and RFs, BRT is also a model-averaging method that constructs a model containing many regression trees. However, successive iterations in boosting are not independent of one another. Rather, the algorithm assesses the fit of the current compound model to the training data at the beginning of each iteration, and fits successive trees predominantly to the observations that the model predicts the least accurately. This is accomplished by modelling the residuals. Hence, boosting is an adaptive method, in that it adjusts the model over successive iterations in an attempt to incorporate outliers.

This “co-operation” amongst successive learners enables the algorithm to model more complex relationships, e.g. functions diagonal in the input space (that is, not perpendicular to any of the input axes). Moreover, the slow learning strategy implemented by boosting has been found to be advantageous, especially in high-dimensional settings. However, an algorithm that fits the training data progressively more closely is liable to overfit, and implementing appropriate stopping rules is a crucial step in developing an accurate BRT model.

6.2 Methods

Boosting involves a number of complexity parameters that affect performance to varying extents. The bag fraction stipulates the proportion of training observations that are randomly sampled for each iteration. Ridgeway (2007) recommends a value of 0.5, which was used in these analyses. The shrinkage parameter specifies the extent to which each tree contributes to the consensus predictions. Ridgeway (2007), Elith *et al.* (2008) and Hastie *et al.* (2009) each recommend using a value that is as small as computationally feasible. Using a tree complexity of 1 (the lowest possible tree complexity), the shrinkage parameter values 0.0005, 0.001, 0.005 and 0.01 were each tested using the `gbm.step` function in the `dismo` package. A low tree complexity setting will result in slower convergence, so a shrinkage parameter that converges for stumps should also converge for higher tree complexity settings. Using a bag fraction of 0.5 and the resulting shrinkage parameter, `gbm.step` was then used to select the optimum number of trees from a maximum of 20 000 to be added to models with tree complexities ranging from 1 to 10.

For each response, a model (the *full model*) with the tree complexity and number of trees producing the lowest average holdout prediction error was then fit. The relative importance of the different input variables in the model was estimated, partial dependence plots were drawn, and the model's validation error was calculated.

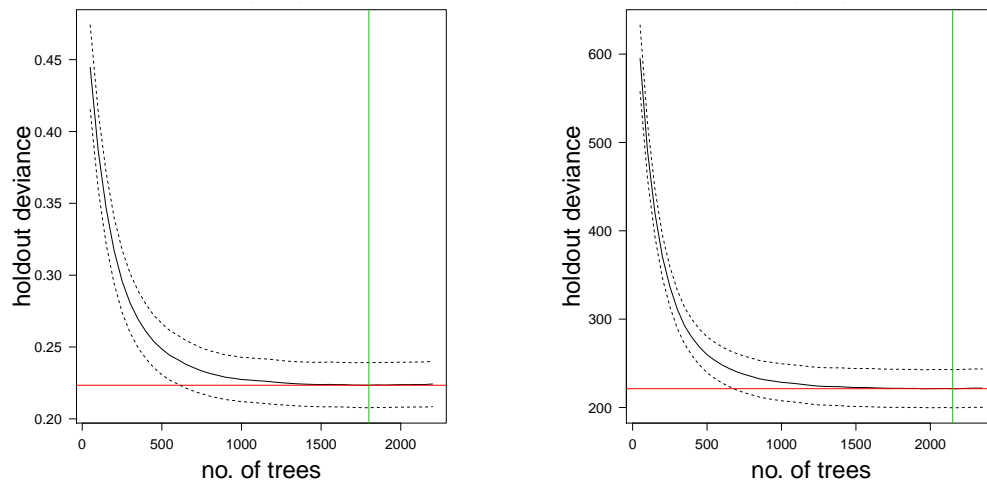
The function `gbm.simplify` in `dismo` was then used to determine the optimum number of input variables to drop from the full model of each response, and the validation errors of the resulting *reduced models* were determined.

The model (full versus reduced) achieving the lowest validation error for each response was chosen as the best boosted model for that response.

The code for fitting the boosted models is presented in Section A.8 on page 227.

6.3 Results

Of the four shrinkage parameters tested, 0.005 was found to be the lowest rate for both responses that resulted in the average holdout prediction error levelling off before 20 000 trees were reached.



(a) median harvest density; tree complexity: 10

(b) total harvest density; tree complexity: 10

Figure 6.1: The mean cross-validation errors (solid black line), along with standard error bands (dashed lines), of the full boosted regression tree model for (a) median and (b) total harvest density containing different numbers of trees. The red line marks the lowest mean CV error obtained, and the green line marks the model size that achieved the lowest mean CV error in each plot.

The CV prediction errors of the resulting boosted models with the optimum tree complexity for each response are shown in Figure 6.1. The model achieving the lowest average holdout prediction error for median harvest density contained 1800 trees, each with an interaction level of 10, whereas a model containing 2150 trees, also with a tree complexity of 10, was found to be optimal for total harvest density.

The importance of the input variables in the resulting median and total harvest density models are shown in Figure 6.2 on the following page. These are similar to those in the bagged (Figure 5.9 on page 170) and RF (Figure 5.14 on page 175) models in the previous chapter, with `avg_WindS` emerging as the most important input variable for median and `avg_maxRH` the most important for total harvest density. The input variables `min_ReIET` and `PlDensity` once again emerged as the least informative for both responses.

Figures 6.3 to 6.4 on pages 187–188 show the fitted responses for the observed values of each of the input variables for median and total harvest density. The scatterplots in Figures 6.3 to 6.4 are similar to the corresponding ones in Chapter 3 (Figures 3.20 to 3.21 on pages 83–84, Figures 3.23 to 3.24 on page 85, Figure 3.28 on page 87, Figure 3.31 on page 89, Figures 3.52 to 3.53

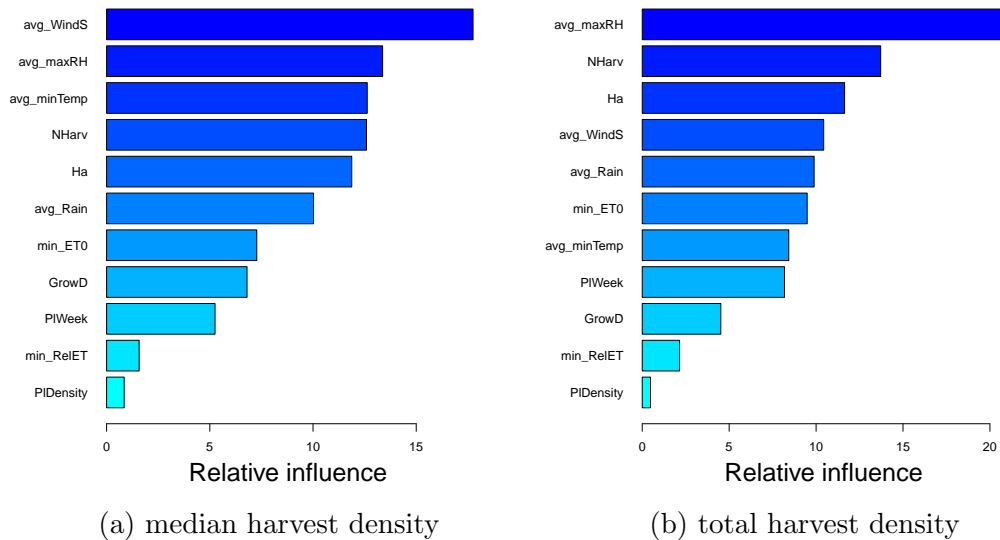


Figure 6.2: Importance of the input variables in the full boosted regression tree model of (a) median and (b) total harvest density.

on pages 114–115 and Figures 3.55 to 3.58 on pages 116–117), although the plots in Figures 6.3 to 6.4 show fewer outlier observations. The boosted models therefore appear to model the response variables reasonably accurately.

Figure 6.5 to 6.6 on pages 189–190 display the fitted response curves of the full boosted models of median and total harvest density, respectively. The values on the y -axes in Figures 6.5 to 6.6 do not convey the predicted harvest density, but rather give an indication of whether harvest density is predicted to increase (positive value) or decrease (negative value) in response to each input variable value. The input variables Ha, PIDensity, PIWeek, avg_minTemp, avg_maxRH, avg_WindS, avg_Rain and min_ET0 share similar tendencies between the two responses, whereas NHarv shows opposite tendencies between the two responses. The variables GrowD, avg_minTemp and avg_WindS appear to be much more informative for predicting median harvest density, whereas avg_maxRH is more informative for total harvest density. The fitted response curve is more or less flat across the domain of min_RelET for both responses, indicating that it is a relatively uninformative input variable. The shapes of these curves are similar to those for the bagged (Figures 5.10 to 5.11 on pages 171–172) and RF (Figures 5.15 to 5.16 on pages 176–177) models.

A comparison of the fitted response values in Figures 6.3 to 6.4 on pages 187–188 with the fitted response curves in Figures 6.5 to 6.6 on pages 189–190 reveals that the response curves across the domains of the most important input

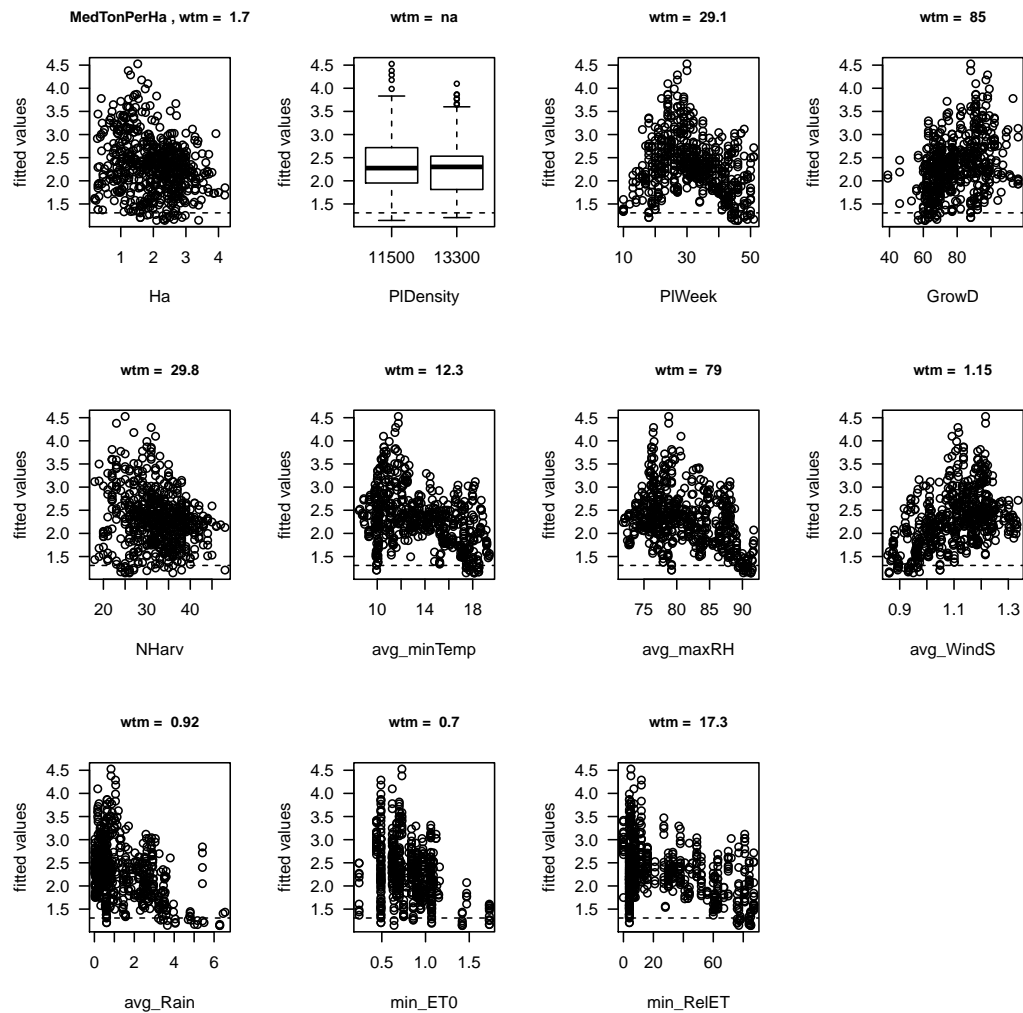


Figure 6.3: Partial dependence plots of the fitted response values against each of the input variables in the full boosted regression tree model of median harvest density after accounting for the average effects of the other input variables on the response. The weighted mean (wtm) of the response for the quantitative input variables is given.

variables (indicated in Figure 6.2 on the preceding page and Figures 6.5 to 6.6) tend to match the patterns evident in the scatterplots more closely than those of the less important input variables.

Figure 6.7 on page 191 shows the validation errors of the full boosted models. Table 6.1 on page 192 shows that these validation errors are considerably better than those obtained by the multiple linear regression and regularised models in Chapter 4 and by the single tree models in Chapter 5, being closer to those obtained for the bagged and RF models in Chapter 5 for both responses.

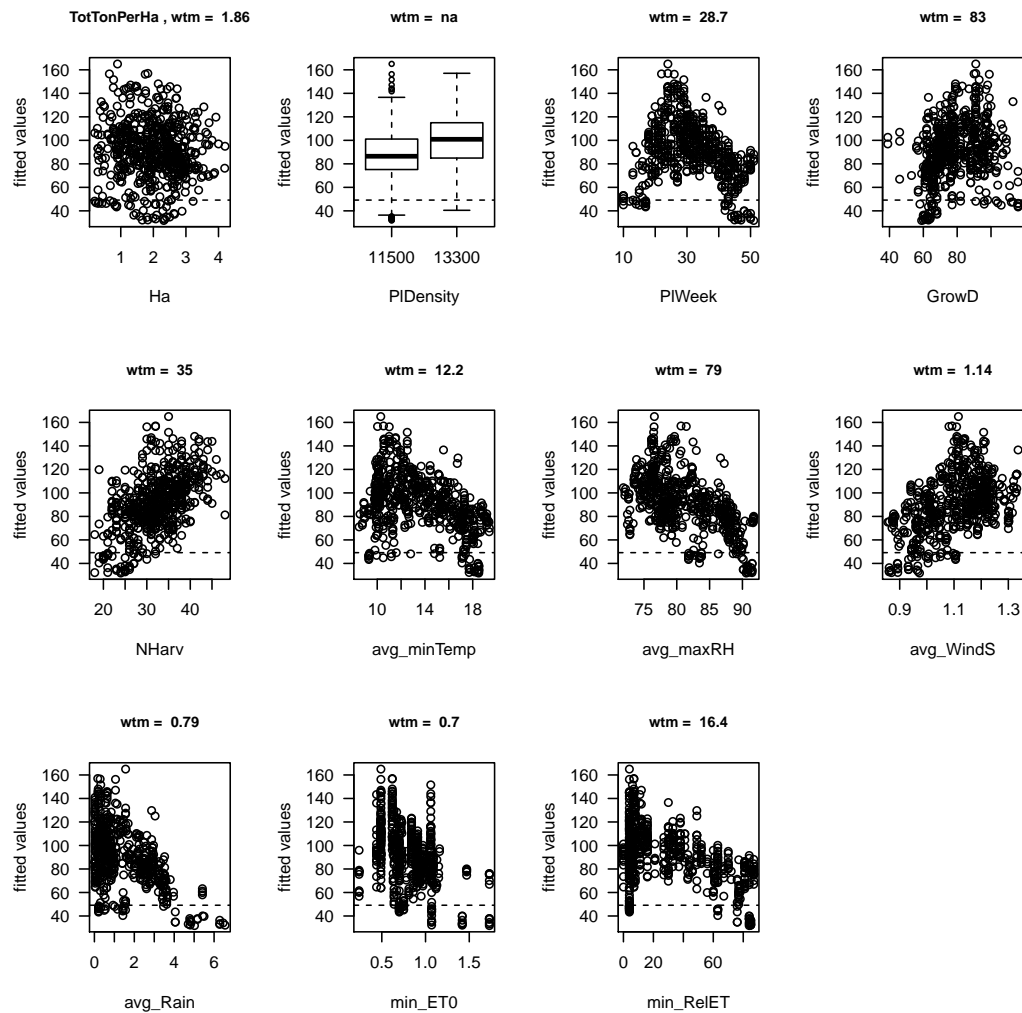


Figure 6.4: Partial dependence plots of the fitted response values against each of the input variables in the full boosted regression tree model of total harvest density after accounting for the average effects of the other input variables on the response. The weighted mean (wtm) of the response for the quantitative input variables is given.

Figure 6.8 on page 191 displays the results of dropping less informative input variables from the full boosted models. The optimum number of input variables to exclude was found to be 1 for median and 3 for total harvest density. Dropping 1 input variable from the full median harvest density boosted model resulted in a validation error of $MSE_{\text{validate}} = 0.2719$ (as opposed to the validation error of $MSE_{\text{validate}} = 0.2670$ for the full model), while excluding 3 input variables from the full total harvest density boosted model resulted in a validation error of $MSE_{\text{validate}} = 287.55$ (as opposed to the validation error of

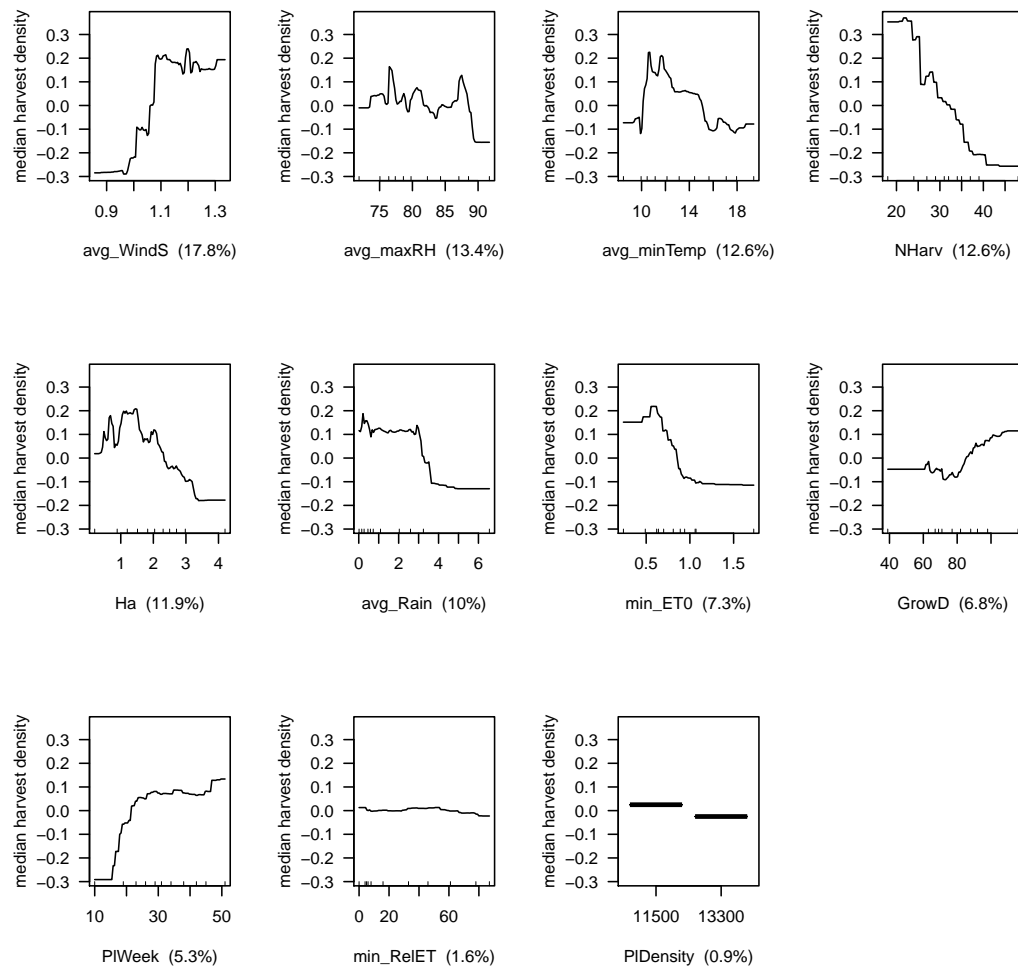


Figure 6.5: Partial dependence plots of the fitted response curve against each of the input variables in the full boosted regression tree model of median harvest density after accounting for the average effects of the other input variables on the response. The relative importance of each input variable in the model is indicated in brackets after the input variable name. The rug at the base of each plot shows the distribution of the deciles of the response in the domain of that input variable.

$MSE_{\text{validate}} = 269.77$ for the full model). These discrepancies in the validation errors between the full and reduced models suggest that all of the input variables contribute towards predicting the response in the boosted models. Since both of the reduced models achieved worse validation errors, the reduced models were not explored any further and the full models were retained.

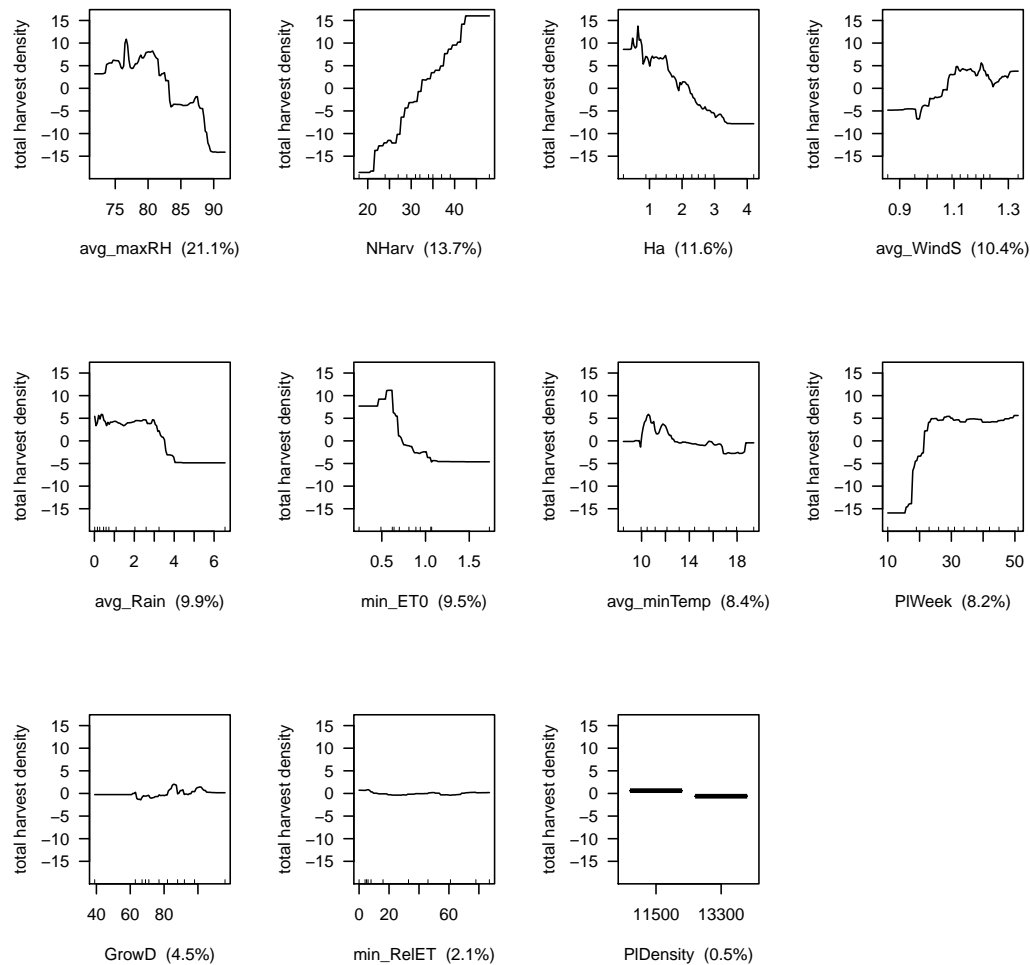


Figure 6.6: Partial dependence plots of the fitted response curve against each of the input variables in the full boosted regression tree model of total harvest density after accounting for the average effects of the other input variables on the response. The relative importance of each input variable in the model is indicated in brackets after the input variable name. The rug at the base of each plot shows the distribution of the deciles of the response in the domain of that input variable.

6.4 Discussion

As is the case for the models in the previous chapters, there are enough observations in the training data set to accommodate all of the input variables in the models, and the validation error therefore tends to increase if input variables, even relatively uninformative ones, are dropped. Consequently, the

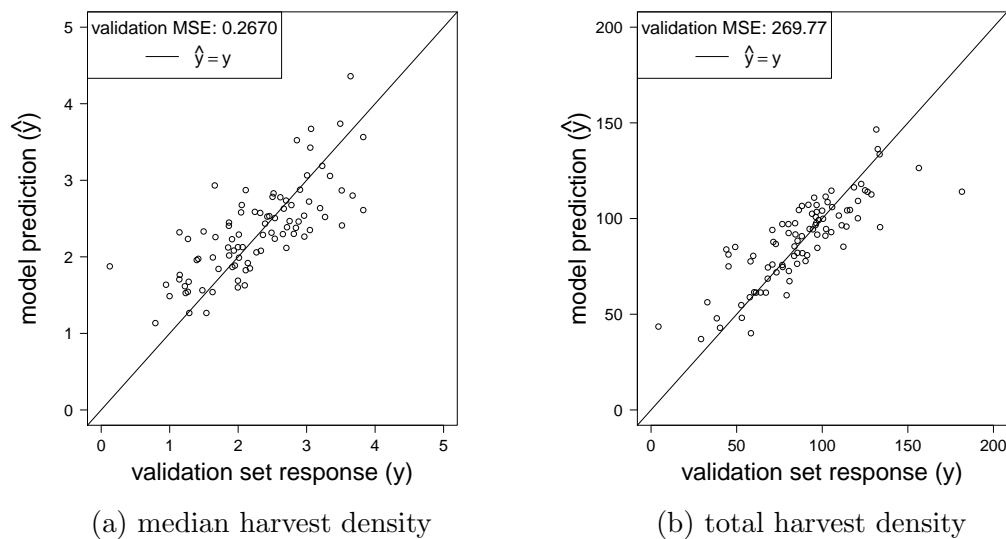


Figure 6.7: Model predictions versus validation responses for the full boosted regression tree model of (a) median and (b) total harvest density.

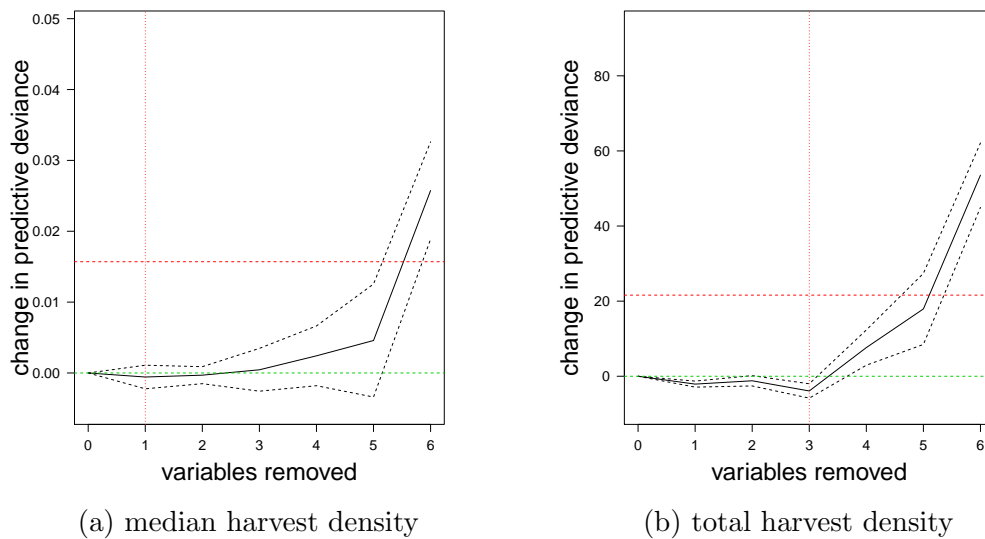


Figure 6.8: The mean cross-validation errors (solid black line), along with error bands (dashed lines), of the number of input variables to be dropped from the full (a) median and (b) total harvest density boosted models. The green line corresponds to no change in the mean CV error, and the vertical red line marks the number of excluded input variables that achieved the lowest mean CV error in each plot.

full boosted models have superior predictive powers to the reduced boosted models. This is, however, to be expected, not only in this study, but probably in boosting analyses in general. Unlike in RFs, where $m < p$ increases

Table 6.1: Validation errors of linear and tree-based models, including boosting, for predicting total and median harvest density using the input variables in Table 3.14 on page 124. The models contain the eleven linear input variables (lin.) or the eleven linear and the four quadratic input variables (lin. & quad.). The MSE obtained on the validation set is given for each model, as well as the percentage improvement of each model's validation error over that of the null model. The lowest validation error obtained for each response is highlighted.

Model	Terms	Median harvest		Total harvest	
		MSE	%	MSE	%
null model	-	0.5912	-	846.40	-
linear regression	lin. & quad.	0.3850	34.88	352.35	58.37
lasso	lin. & quad.	0.3843	35.00	352.89	58.37
full regression tree	lin.	0.3317	43.89	313.01	63.02
bagging	lin.	0.2596	56.09	255.98	69.76
random forests	lin.	0.2393	59.53	272.02	67.86
boosting	lin.	0.2670	54.84	269.77	68.13

the likelihood of all of the input variables (even the less useful ones) being used in the model, the boosting algorithm is under no such constraint, and is therefore free to ignore uninformative input variables (Hastie *et al.*, 2009). Consequently, the fact that an input variable has been included in a BRT model means that it is useful, and dropping useful input variables is likely to result in a reduction in the predictive power of the model. Reducing the number of input variables in a BRT model is therefore probably chiefly a means to increase the interpretability of the model, rather than to increase accuracy.

Table 6.1 shows that, of all the model types applied in this study, bagging, RFs and boosting achieved the lowest validation errors. These are all techniques that have been proposed within the last few decades to model highly nonlinear relationships accurately by using highly flexible component models while keeping the bias and variance of the final model in check.

Of these three model types, boosting achieved the worst validation error for median harvest density (0.2670 for boosting, 0.2596 for bagging and 0.2393 for RFs) and a validation error close to the worst for total harvest density (272.02 for RFs, 269.77 for boosting and 255.98 for bagging). The component models in bagging and RFs are constructed independently of one another from independent bootstrap realisations of the training data sets. Consequently,

each tree is confronted with similar scenarios that must be modelled. The individual models are then averaged to produce more stable predictions. In contrast, boosting follows a much more directed approach, fitting successive models to the residuals of the growing model in an attempt to produce a model that accommodates all of the observations into its structure. The strategy followed by boosting is highly advantageous in cases in which the response surface cannot be modelled well by a regression tree. A tree model consists of a hierarchy of splits, with each split forming a step in the predicted response surface perpendicular to the axis of the input variable that was split on in the input space. So, if the true response surface, for example, consists of steps or inclines, each of which run perpendicular to one of the input axes, then a single tree will be able to model this surface accurately. However, if the steps or slopes of the true response surface are inclined at angles diagonal to all of the input axes in the input space, then a single tree will struggle to model this function and will produce a model that will do no better than random guessing. Likewise, multiple independent regression trees will each struggle to model this relationship, and bagging and RFs will therefore produce final predictions that are not much better than guessing. This is a scenario in which boosting will have the upper hand. Each successive tree in the boosting algorithm will be confronted with large residuals in parts of the input space different from those that the previous tree modelled, resulting in a sequence of very different trees that should together produce a reasonably accurate model of the true response surface.

While this ability of the boosting algorithm to adapt to the observations in the different parts of the input space is highly advantageous in some settings, it can also be a hindrance. It makes the algorithm much more susceptible to unusual or erroneous values, especially in sparsely sampled parts of the input space, and might lead to a model that models parts of the input space poorly. It could be this weakness that is leading to boosting generating less accurate predictions in this study. During the data-cleaning process, quite a few unrealistic observations were identified and removed (see Section 3.1.3 on page 72). However, less obvious errors could easily have been overlooked.

As is the case for the models in Chapters 4 to 5, the full total harvest density boosted model in this chapter explains more of the variance in its response than does the full median harvest density boosted model (compare

the sizes of the residuals (the shape of the “residual cloud”) on the validation set between plots (a) and (b) in Figure 4.11 on page 142, Figure 4.18 on page 151, Figure 5.7 on page 168, Figure 5.12 on page 173, Figure 5.17 on page 178 and Figure 6.7 on page 191, and the percentage improvement in the validation MSE of each model over the null model for each response in Table 6.1 on page 192). Hence, despite the fact that the median is a more robust statistic than is the total, the statistical techniques model total harvest density more accurately than median harvest density. This suggests that the input variables are more informative for total harvest density.

The input variables provide information on the area of the crop, planting density, time of year in which the crop was planted, the length of the growing period, the number of harvest events and the weather conditions over the growing period (Table 3.14 on page 124). The input variables are therefore informative in terms of the productivity (or yield density, and therefore also harvest density) of the crop (as opposed to other factors that might influence the quantity of tomatoes harvested, e.g. the market price of tomatoes, availability of labour for picking, etc.). The more factors that are not included in the models that influence the quantities of tomatoes harvested, the less accurate will be the resulting models.

Considering the nature of the input variables, the results in Table 6.1 on page 192 suggest that the cumulative harvest density over the harvest period is more dependent on the yield density of the crops than are the quantities of tomatoes harvested during individual harvest events. The median harvest densities of the crops would be less dependent on yield density if the quantity of tomatoes harvested on any given day is influenced by factors other than the quantity of tomatoes ripe enough to be picked. This would be the case if, for example, the size of the picking team sent to a block crop and/or the time available for harvesting the fruit from each block crop is not solely dependent on the quantity of tomatoes available for picking, or if “harvest effort” is partially dependent on the current market price of tomatoes. Such situations would result in the tomatoes not necessarily being harvested as soon as they are ripe enough, making median harvest density less predictable from the included input variables.

An alternative explanation for the superior predictive powers of the total as opposed to the median harvest density models in this study is that the

total yield (and therefore total harvest) of tomato crops may be more stable than the distribution of that yield throughout the yield period. This would be the case if the yield quantity is already largely determined by the time the harvest period starts (for example, if the total yield quantity is mostly dependent on genetics and the total biomass accumulated during the growing period), and that the weather during the harvest period only influences yield by determining the rate at which the total biomass is transferred to the fruit. This scenario would also make median harvest a less stable statistic than total harvest.

Chapter 7

Discussion

The SLT methods employed to model harvest density from weather, planting and harvest variables were all relatively successful in predicting the response variables. All of the modelling techniques achieved lower validation errors than did the corresponding null model, indicating that at least one of the included input variables is useful for modelling harvest quantity. As is the case with many data sets, the ensemble techniques (bagging, RFs and boosting) proved to be more accurate than techniques that fit a single model to the data (MLR, the lasso and regression trees), probably because ensemble methods allow for more complex models to be fit to the data while implementing ways to reduce overfitting. The nonlinear models used predict harvest density better than do the models based on the multiple linear regression coefficients, suggesting that harvest density has a highly nonlinear relationship with the included crop and weather variables.

The most successful model for predicting median harvest density was RFs, with a validation error of $MSE_{\text{validate}} = 0.2393$. This model achieved test errors of $MSE_{\text{test}} = 0.2410$ and $MAE_{\text{test}} = 0.3732$, suggesting that the model will predict the median harvest density of future crops with an accuracy of approximately ± 0.37 t/ha, on average. The most successful model for predicting total harvest density was bagged regression trees, with a validation error of $MSE_{\text{validate}} = 255.98$. This model achieved test errors of $MSE_{\text{test}} = 363.12$ and $MAE_{\text{test}} = 12.67$, suggesting that the model will predict the total harvest density of future crops with an accuracy of about ± 12.67 t/ha, on average. The smaller value of MSE_{validate} compared to MSE_{test} for both models is to be expected, since these models were chosen based on the fact that they attained

the smallest prediction errors on the validation set, and the validation set is therefore biased in favour of these models. However, there is a large discrepancy between MSE_{validate} and MSE_{test} in the case of the bagged regression trees model for total harvest density. This is probably due to the small sizes of the validation and test sets (88 and 50 observations, respectively) relative to the training set (600 observations), and better estimates of future prediction accuracy for both responses would probably have been obtained if the observations had been partitioned more evenly amongst the three data sets.

Median harvest density of future crops on the tomato farm can therefore be predicted using the RF model developed during this study, while total harvest density can be predicted using the bagged model. The values for most of the models' input variables will already be known at planting time (block crop area and planting density and week) or can be input with increasing accuracy as the harvest period approaches (length of the growing period and the summary statistics of the weather variables). In contrast, the number of harvest events is only known for certain at the end of the harvest period, when generating a prediction of harvest quantity is no longer necessary. It may seem strange that a variable such as the number of harvest events that can only be known at the end of the harvest period is included in models meant for predicting harvest quantities. However, these models were not only developed for predictive purposes, but also for interpretive purposes. Possible values that could be input into the model for this variable when predicting the harvest density of a crop include the minimum, average and maximum number of harvest events for block crops grown at the same time of year on this farm. Inputting different values for this variable into the model will provide an idea of the range of harvest quantities that can be expected depending on the number of harvest events. For generating values for the summary statistics of the weather time series over the growing period of a crop, weather for the next few weeks can be simulated using a stochastic weather generator based on weather records from previous weeks and what is known about the weather at the particular time of year from historical records. Stochastic weather generators available on the internet include LARS-WG (Barrow and Semenov, 1995), Marksim (Jones and Thornton, 2000) and ClimGen (Stöckle *et al.*, 2001). The closer one is to the end of a crop's growing period, the smaller will be the proportion of weather data that will have to be simulated and the greater the

accuracy of predicted harvest quantities should therefore be.

The fact that the lasso did not achieve superior validation errors to MLR, the pruned regression trees did not accomplish better validation errors than did the full regression trees (indeed, the validation errors of the pruned trees were much worse than were those of the full trees), and RFs did not outperform bagging provide evidence that the model parameter estimators tend to have low variance. This indicates that the training data set contains enough observations to model harvest density using static modelling techniques. The large number of training observations enables the modelling algorithms to incorporate all of the input variables into their structures without suffering (much) from dimensionality and/or multicollinearity problems. The models tended to achieve inferior results if they were restricted with regards to the input variables that they are allowed to incorporate, suggesting that all of the included input variables (as well as the four quadratic terms in the case of the linear models) are important for predicting harvest density.

Of the eleven predictors (the eleven linear terms, which were included in both the linear and nonlinear models), the average of the daily average wind speed readings over the crops' growing periods emerged as important for predicting median harvest density. Air movement is important, since it promotes the removal of the saturated layer of air surrounding leaves. This both provides higher concentrations of atmospheric CO₂ to fuel photosynthesis—a process that synthesises the carbohydrates necessary for a plant's survival and for high yields of good quality fruit—and promotes transpiration. Plants in greenhouses often suffer as a result of retarded air movement, more easily suffering from heat damage under high temperatures due to the lack of cooling that would result from greater transpiration (Taiz and Zeiger, 2002). Hence, as long as the wind is not strong enough to damage plants structurally or to cause desiccation, air movement is beneficial to plants. In contrast, the average of the daily maximum relative humidity readings over the crops' growing periods emerged as an important predictor of total harvest density. A high relative humidity also hinders evaporation and therefore the ability of the plant to dissipate heat. A high relative humidity combined with high temperatures, which is characteristic of the summers in the farm's region, can cause heat stress. Heat stress retards important processes such as photosynthesis, slowing down growth and reducing yield (Taiz and Zeiger, 2002).

All of the modelling techniques predict total harvest density more accurately than median harvest density. Possible explanations are that total yield is a more stable statistic of a crop's yield time series than is median yield, and/or that total harvest is a more stable statistic than is median harvest based on the input variables. Most of the input variables included in the models contain information collected during the crops' growing periods, and greater predictability of total relative to median harvest density would therefore be expected if total yield is largely determined by conditions during the crop's growing period whereas median yield is more heavily influenced by conditions during the harvest period. It could, for example, be the case that total yield is largely determined by the amount of photosynthesis that the crop plants managed to achieve over the crop's growing period, whereas the rate at which crop plants yield fruit is more heavily influenced by environmental conditions during the harvest period. It is certainly the case that a cold snap during the harvest period can result in far fewer tomatoes being harvested over subsequent days, while an unusually warm period results in accelerated ripening of the developing fruit. However, it may be the case that these conditions also affect total harvest density. As with crop yield, the models would predict total harvest density more accurately than median harvest density if the input variables are more informative with regards to total harvest density. This would be the case if the farmer always harvests all of the fruit produced by a crop (thereby keeping total harvest density variance to a minimum), but not necessarily at the rate at which the tomatoes reach maturity on the plants (thereby increasing the variance of median harvest density). Inconsistent harvesting of fruit would be facilitated by growing cultivars that have long shelf-lives, which would give the farmer more flexibility with regards to when he chooses to harvest the fruit.

This study has several limitations, one of which is that the validation and test sets were relatively small, and the prediction errors reported in this study are probably less accurate as a result. However, it is probably more useful to the farmer to have more accurate models (as a result of a larger training set) than to have more accurate estimates of the prediction accuracy of the models (as a result of larger validation and test sets). The accuracy of the models will become more apparent as they are used in the prediction of crop harvest quantities on the farm.

Another limitation is that there appears to be a large irreducible error, which could have diverse causes. Only weather and a few other predictors are included here—influential variables such as the previous crop, soil characteristics (e.g. type, depth, pH, organic carbon and nitrogen content, etc.), field characteristics (e.g. slope and aspect), pruning, irrigation and fertiliser regimes and incidents such as pests, diseases and weather damage (wind, hail, etc.) would provide more information about growing conditions, probably yielding more accurate models. Furthermore, information on the timing of the different phases of the tomato crops (e.g. seedling establishment, vegetative growth, flower formation, fruit formation, fruit ripening) would make it possible to summarise the time series data over each crop phase (rather than over the growing period as a whole). Tomato plants require different environmental conditions as they mature (see Section 2.1 on page 6), and incorporating more detailed weather conditions into the analyses may yield more accurate models. An important weather variable that was absent from these analyses is incident solar radiation. It may also be useful to include information on tomato price in the models, since this information is probably taken into consideration when decisions are made regarding whether to continue harvesting tomatoes from a crop that is past its best. Another possible contributor to variance in the response is that the models were fit to harvest data instead of to yield data. While both harvest and yield quantities should be affected by the weather, harvest quantities will also be affected by additional factors such as the price of tomatoes and labour availability during the harvest period. However, harvest quantities are probably reasonably accurate reflections of crop yield. The weather time series were summarised in order to obtain numerical variables that could be input into static learning algorithms, which probably resulted in a large loss of information. Moreover, only the weather from the growing period was taken into account in these analyses. Although the weather during a crop's harvest period is bound to affect its yield to a certain extent, it will predominantly affect the yield during the latter parts of the harvest period, and it is therefore debatable whether it should be included in the models. A further possible contributor to the large irreducible error is that the ARC weather data set was obtained from a weather station about 51 m higher up and 4.13 km away from the tomato farm, and therefore probably contains entries slightly different from the weather that the tomato plants were exposed

to in their fields. In addition, the crops included in these analyses contained different tomato cultivars (although all of the same type of tomato), which may have slightly different genetic predispositions to yield quantities under different environmental conditions. While the tomatoes were predominantly harvested for fresh consumption, tomatoes at the end of a crop's harvest period were occasionally harvested for the canning industry. Consequently, a few of the crop records in the data sets probably have inflated harvest values relative to what they would have had if they had only been harvested for the fresh market. Finally, the raw data contained unrealistic (and therefore probably erroneous) entries. While the most obvious errors were excluded, it is likely that the models were trained and tested on some inaccurate readings.

Ideas for future work include transforming predictor and/or response variables. Generalised additive models (GAMs) allow nonlinear relationships to be modelled between the response and each of the predictors, and would therefore probably produce better prediction errors than does MLR. However, the scatterplots of the response variables against each of the predictor variables in Chapter 3 suggest that the relationships can be adequately modelled using linear (and quadratic terms). GAMs may therefore be of limited use for this data set. Another disadvantage is that GAMs only allow additive relationships amongst the predictors, and therefore might not model harvest quantities as accurately as do tree-based methods. As was the case with MLR and the lasso (Chapter 4), principal components regression and partial least squares yielded harvest quantity models with large validation errors (results of these analyses not shown in this document due to space constraints). Hence, additive models in general appear to be suboptimal for modelling harvest quantity. This is probably because they cannot model interactions amongst the predictors, which is probably important for this data set. In addition to tree-based methods, an alternative nonlinear modelling technique that could be tried is support vector regression. Although they are less useful for interpretive purposes, deep-learning methods such as multi-layered neural networks are also good options for modelling processes accurately. However, considering the nature of the data, a family of statistical techniques worth trying is functional data analysis (FDA). FDA enables more information from the weather time series to be included in the models (rather than just summary statistics of the time series data). Consequently, this approach is likely to result in more accurate models,

provided that it can accommodate the type of relationship that exists between the response and the input variables. For example, as mentioned earlier in this paragraph, it must be possible to model nonadditive relationships between the functional input variables, scalar input variables and the response. Performing FDA would probably require the imputation of missing time series readings. Moreover, the weather time series of the crops are highly variable in length (crop growing period ranges from 39 to 116 days), and the time series may have to be summarised in some way prior to the analysis to make them more similar in length. For example, consecutive values within each time series could be averaged to ensure that all time series have a length of ten values, irrespective of their original lengths. Once the most promising modelling techniques have been explored, predictions from diverse model types can be combined using stacking and other model-averaging methods in an attempt to produce even more accurate consensus predictions.

Appendices

Appendix A

Code for the statistical modelling analyses in R

This appendix contains the R code used in the modelling analyses of Chapters 4–6. Much of the code was obtained from James *et al.* (2013). However, the code for specific methods was adjusted from that provided by James *et al.* (2013) with the help of various sources: Kuhn (2015*b*) for using the `caret` package to optimise model parameters, Hastie and Qian (2014) for the lasso code, Therneau and Atkinson (2015) and Milborrow (2016*a*) for the regression tree code, and Elith and Leathwick (2016) for the boosted regression tree code.

A.1 Preparing the data sets

```
1 # Setting up the training, validation and test data sets:
2
3 > test_validation_training_indices_sorted <- c(rep(x=1, 50), rep(x=2,
4     88), rep(x=3, 600))
5
6 > test_validation_training_indices <- sample(x=test_validation_
7     training_indices_sorted, size=length(test_validation_training_
8     indices_sorted), replace=FALSE)
9
10 > yield_clim_test_df <- yield_clim_dataset[test_validation_training_
11     indices == 1, ]
12
13 > yield_clim_validate_df <- yield_clim_dataset[test_validation_
14     training_indices == 2, ]
15
16 > yield_clim_train_df <- yield_clim_dataset[test_validation_training_
17     indices == 3, ]
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
15
16
17
18 # Determine number of observations in the training, validation and
    test sets:
19
20 num_test_cases <- nrow(yield_clim_test_df)
21 stopifnot(num_test_cases == 50)
22
23 num_validate_cases <- nrow(yield_clim_validate_df)
24 stopifnot(num_validate_cases == 88)
25
26 num_train_cases <- nrow(yield_clim_train_df)
27 stopifnot(num_train_cases == 600)
28
29
30
31 # Converting P1Density to a categorical variable:
32
33 yield_clim_test_df$P1Density <- as.factor(yield_clim_test_df$
    P1Density)
34 yield_clim_validate_df$P1Density <- as.factor(yield_clim_validate_
    df$P1Density)
35 yield_clim_train_df$P1Density <- as.factor(yield_clim_train_df$
    P1Density)
36
37
38
39 # Get column numbers for predictor and response variables:
40
41 get_col_numbers_from_train_df <- function(column_name, train_df) {
42     col_numbers_vec <- which(names(train_df) == column_name)
43     return(col_numbers_vec)
44 }
45
46
47
48
49
50 # linear predictor terms:
51
52 input_column_names_vec <- c('Ha', 'P1Density', 'P1Week', 'GrowD',
    'NHarv', 'avg_minTemp', 'avg_maxRH', 'avg_WindS', 'avg_Rain',
    'min_ET0', 'min_ReLET')
53 stopifnot(is.atomic(input_column_names_vec))
54 stopifnot(length(input_column_names_vec) == 11)
55
56 input_column_numbers_all_vec <- apply(X=matrix(data=input_column_
    names_vec, ncol=1), MARGIN=1, FUN=get_col_numbers_from_train_
    df, yield_clim_train_df)
57 stopifnot(is.atomic(input_column_numbers_all_vec))
58 stopifnot(length(input_column_names_vec) == length(input_column_
    numbers_all_vec))
59
60 input_column_numbers_numeric_vec <- input_column_numbers_all_vec
    [-2] # exclude the categorical variable P1Density
61 stopifnot(is.atomic(input_column_numbers_numeric_vec))
62 stopifnot(length(input_column_numbers_numeric_vec) == length(input
    _column_numbers_all_vec) - 1)
63
64
65 # quadratic predictor terms:
66
```



```

67     numeric_input_2_column_names_vec <- c('PlWeek_2', 'avg_minTemp_2',
68       'min_ET0_2', 'min_RelET_2')
69     stopifnot(is.atomic(numeric_input_2_column_names_vec))
70     stopifnot(length(numeric_input_2_column_names_vec) == 4)
71     numeric_input_2_column_numbers_all_vec <- apply(X=matrix(data=
72       numeric_input_2_column_names_vec, ncol=1), MARGIN=1, FUN=get_
73       col_numbers_from_train_df, yield_clim_train_df)
74     stopifnot(is.atomic(numeric_input_2_column_numbers_all_vec))
75     stopifnot(length(numeric_input_2_column_names_vec) == length(
76       numeric_input_2_column_numbers_all_vec))
77
78     # response variables:
79     response_column_names_list <- list(TotalTonPerHa='TotTonPerHa',
80       MedianTonPerHa='MedTonPerHa')
81     stopifnot(length(response_column_names_list) == 2)
82     response_column_numbers_list <- lapply(X=response_column_names_
83       list, FUN=get_col_numbers_from_train_df, yield_clim_train_df)
84     stopifnot(length(response_column_names_list) == length(response_
85       column_numbers_list))
86
87     # Define formulas for modelling algorithms:
88
89     MedianTonPerHa_linear_yield_formula <- paste('MedTonPerHa ~',
90       paste(input_column_names_vec, collapse=' + '))
91     MedianTonPerHa_linear_yield_formula <- as.formula(MedianTonPerHa_
92       linear_yield_formula)
93     MedianTonPerHa_linear_quadratic_yield_formula <- paste('
94       MedTonPerHa ~', paste(input_column_names_vec, collapse=' + '),
95       ' + ', paste(numeric_input_2_column_names_vec, collapse=' + '
96       ))
97     MedianTonPerHa_linear_quadratic_yield_formula <- as.formula(
98       MedianTonPerHa_linear_quadratic_yield_formula)
99
100    TotalTonPerHa_linear_yield_formula <- paste('TotTonPerHa ~', paste
101      (input_column_names_vec, collapse=' + '))
102    TotalTonPerHa_linear_yield_formula <- as.formula(TotalTonPerHa_
103      linear_yield_formula)
104    TotalTonPerHa_linear_quadratic_yield_formula <- paste('TotTonPerHa
105      ~', paste(input_column_names_vec, collapse=' + '), ' + ',
106      paste(numeric_input_2_column_names_vec, collapse=' + '))
107    TotalTonPerHa_linear_quadratic_yield_formula <- as.formula(
108      TotalTonPerHa_linear_quadratic_yield_formula)
109
110    return(list(
111      train_df=yield_clim_train_df, num_train_cases=num_train_cases,
112      validate_df=yield_clim_validate_df, num_validate_cases=num_
113        validate_cases,
114      test_df=yield_clim_test_df, num_test_cases=num_test_cases,

```

```

110     MedianTonPerHa_linear_yield_formula=MedianTonPerHa_linear_
        yield_formula, num_vars_linear_MedianTonPerHa_formula=
        length(input_column_names_vec),
111     MedianTonPerHa_linear_quadratic_yield_formula=MedianTonPerHa_
        linear_quadratic_yield_formula, num_vars_linear_quadratic_
        MedianTonPerHa_formula=length(input_column_names_vec),
112     TotalTonPerHa_linear_yield_formula=TotalTonPerHa_linear_yield_
        formula, num_vars_linear_TotalTonPerHa_formula=length(
        input_column_names_vec),
113     TotalTonPerHa_linear_quadratic_yield_formula=TotalTonPerHa_
        linear_quadratic_yield_formula, num_vars_linear_quadratic_
        TotalTonPerHa_formula=length(input_column_names_vec),
114
115     input_linear_column_names_vec=input_column_names_vec, input_
        linear_column_numbers_all_vec=input_column_numbers_all_vec
        ,
116     input_quadratic_column_names_vec=numeric_input_2_column_names_
        vec, input_quadratic_column_numbers_all_vec=numeric_input_
        2_column_numbers_all_vec,
117     response_column_names_list=response_column_names_list,
        response_column_numbers_list=response_column_numbers_list
118
119 ))
120
121 }
122
123
124
125
126 get_data_sets_and_models(yield_clim_test_df, yield_clim_validate_df,
        yield_clim_train_df)

```

A.2 The null model

```

1 training_null_model <- function(response, xlim_ylim_vec, training_df,
        validation_df, num_validate_cases) {
2
3
4
5     # training the model:
6
7     mean_train_response_vec <- rep(x=mean(training_df[, response]),
        length=num_validate_cases)
8     stopifnot(length(mean_train_response_vec) == nrow(validation_df))
9     stopifnot(length(unique(mean_train_response_vec)) == 1)
10
11     print('mean training response:', sprintf('%.4f', mean_train_
        response_vec[1]), '\n')
12
13
14
15     # finding the validation error:
16
17     validation_diff_vec <- mean_train_response_vec - validation_df[,
        response]
18     validation_MSE <- mean(validation_diff_vec^2)
19
20     print('\nvalidation MSE:', sprintf('%.4f', validation_MSE))

```

```

21
22
23
24 pdf(paste('validation_null_model_', response, '.pdf', sep=''))
25
26 old_par <- par(mar=c(4.5, 7, 2, 4), cex.lab=2, cex.axis=1.3, las
    =1)
27
28 plot(x=validation_df[, response], y=mean_train_response_vec, xlim=
    xlim_ylim_vec, ylim=xlim_ylim_vec, xlab='validation set
    response (y)', ylab='')
29 abline(a=0, b=1)
30 axis(side=2, at=mean_train_response_vec, labels=sprintf('%.2f',
    mean_train_response_vec), las=1, cex.axis=1.3)
31 title(ylab=expression(paste('mean training set response(', hat(y),
    ')', sep='')), line=4)
32 legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
    paste('validation MSE:', sprintf(switch(EXPR=response,
    TotTonPerHa='%.2f', MedTonPerHa='%.4f'), validation_MSE)), cex
    =1.5)
33
34 par(old_par)
35
36 dev.off()
37
38
39 if (response == 'TotTonPerHa') {
40
41     return(TotalTonPerHa_null_validation_MSE=validation_MSE)
42
43 } else {
44
45     stopifnot(response == 'MedTonPerHa')
46     return(MedianTonPerHa_null_validation_MSE=validation_MSE)
47
48 }
49
50
51 }
52
53
54
55 # median tonnes/hectare null model:
56
57 training_null_model(response="MedTonPerHa", xlim_ylim_vec=c(0, 5),
    training_df=train_df, validation_df=validate_df, num_validate_
    cases=num_validate_cases)
58
59
60 # total tonnes/hectare null model:
61
62 training_null_model(response="TotTonPerHa", xlim_ylim_vec=c(0, 200),
    training_df=train_df, validation_df=validate_df, num_validate_
    cases=num_validate_cases)

```

A.3 Multiple linear regression

```

1 library(leaps) # for performing best subset selection

```

```
2
3
4
5
6 # Best subset selection for median harvest density:
7
8 > response <- 'MedTonPerHa'
9 > formula_to_use <- MedianTonPerHa_linear_quadratic_yield_formula
10
11
12
13 > regfit.full <- regsubsets(x=formula_to_use, data=training_df, nvmax
    =15)
14 > reg.summary <- summary(regfit.full)
15
16
17
18 > which.max(reg.summary$adjr2)
19 [1] 11
20 > which.min(reg.summary$cp)
21 [1] 10
22 > which.min(reg.summary$bic)
23 [1] 9
24
25
26 > pdf(paste('rss_adjr2_cp_bic_curves_quadratic_', response, '.pdf',
    sep=''))
27
28 > oldpar <- par(cex.lab=1.5, las=1, mar=c(4.5, 5, 2, 2), mfrow=c(2, 2)
    , mgp=c(3, 0.8, 0))
29
30 > plot(reg.summary$rsq, xlab="number of variables", ylab=expression(R
    ^2), type="l")
31
32 > plot(reg.summary$adjr2, xlab="number of variables", ylab=expression(
    paste('adjusted ', R^2)), type="l")
33 > points(11, reg.summary$adjr2[11], col="red", cex=2, pch=20)
34
35 > plot(reg.summary$cp, xlab="number of variables", ylab=expression(C[p
    ]), type="l")
36 > points(10, reg.summary$cp[10], col="red", cex=2, pch=20)
37
38 > plot(reg.summary$bic, xlab="number of variables", ylab="BIC", type="
    l")
39 > points(9, reg.summary$bic[9], col="red", cex=2, pch=20)
40
41 > par(oldpar)
42
43 dev.off()
44
45
46
47
48 > pdf(paste('r2_blocks_quadratic_', response, '.pdf', sep=''))
49 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
    1, 0, 0))
50 > plot(regfit.full, scale="r2", col=gray(seq(0, 0.9, len=11)))
51 > par(oldpar)
52 > dev.off()
53
54
55 > pdf(paste('adjr2_blocks_quadratic_', response, '.pdf', sep=''))
```

```

56 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
57   1, 0, 0))
58 > plot(regfit.full, scale="adjr2", col=gray(seq(0, 0.9, len=11)))
59 > par(oldpar)
60 > dev.off()
61
62 > pdf(paste('Cp_blocks_quadratic_', response, '.pdf', sep=''))
63 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
64   1, 0, 0))
65 > plot(regfit.full, scale="Cp", col=gray(seq(0, 0.9, len=11)))
66 > par(oldpar)
67 > dev.off()
68
69 > pdf(paste('bic_blocks_quadratic_', response, '.pdf', sep=''))
70 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
71   1, 0, 0))
72 > plot(regfit.full, scale="bic", col=gray(seq(0, 0.9, len=11)))
73 > par(oldpar)
74 > dev.off()
75
76
77
78 > pdf(paste('r2_adjr2_cp_bic_blocks_quadratic_', response, '.pdf', sep
79   =''))
80 > oldpar <- par(cex.lab=1.5, las=1, mar=c(4.5, 5, 2, 2), mfrow=c(2, 2)
81   , mgp=c(3, 0.8, 0))
82 > plot(regfit.full, scale="r2", col=gray(seq(0, 0.9, len=11)))
83 > plot(regfit.full, scale="adjr2", col=gray(seq(0, 0.9, len=11)))
84 > plot(regfit.full, scale="Cp", col=gray(seq(0, 0.9, len=11)))
85 > plot(regfit.full, scale="bic", col=gray(seq(0, 0.9, len=11)))
86
87 > par(oldpar)
88
89 > dev.off()
90
91
92
93 > coef(regfit.full, 9)
94   (Intercept) PlDensity13300          GrowD          NHarv      avg_
95   minTemp
96   -1.830848047  -0.170398068    0.014351731  -0.027503708
97   0.600355866
98   avg_Rain      min_RelET      PlWeek_2  avg_minTemp_2      min_
99   ETO_2
100   -0.134208202  -0.016285414    0.001659646  -0.026098092
101   -0.429940346
102
103 # Best subset selection for total harvest density:
104 > response <- 'TotTonPerHa'
105 > formula_to_use <- TotalTonPerHa_linear_quadratic_yield_formula
106
107
108

```

```
109 > regfit.full <- regsubsets(x=formula_to_use, data=training_df, nvmax
110 =15)
111 > reg.summary <- summary(regfit.full)
112
113
114 > which.max(reg.summary$adjr2)
115 [1] 11
116 > which.min(reg.summary$cp)
117 [1] 11
118 > which.min(reg.summary$bic)
119 [1] 8
120
121
122 > pdf(paste(folder_path, 'lin_regress_model/figures/rss_adjr2_cp_bic_
123 curves_quadratic_', response, '.pdf', sep=''))
124 > oldpar <- par(cex.lab=1.5, las=1, mar=c(4.5, 5, 2, 2), mfrow=c(2, 2)
125 , mgp=c(3, 0.8, 0))
126 > plot(reg.summary$rsq, xlab="number of variables", ylab=expression(R
127 ^2), type="l")
128 > plot(reg.summary$adjr2, xlab="number of variables", ylab=expression(
129 paste('adjusted ', R^2)), type="l")
130 > points(11, reg.summary$adjr2[11], col="red", cex=2, pch=20)
131 > plot(reg.summary$cp, xlab="number of variables", ylab=expression(C[p
132 ]), type="l")
133 > points(11, reg.summary$cp[11], col="red", cex=2, pch=20)
134 > plot(reg.summary$bic, xlab="number of variables", ylab="BIC", type="
135 l")
136 > points(8, reg.summary$bic[8], col="red", cex=2, pch=20)
137 > par(oldpar)
138
139 > dev.off()
140
141
142
143 > pdf(paste(folder_path, 'lin_regress_model/figures/r2_blocks_
144 quadratic_', response, '.pdf', sep=''))
145 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
146 1, 0, 0))
147 > plot(regfit.full, scale="r2", col=gray(seq(0, 0.9, len=11)))
148 > par(oldpar)
149 > dev.off()
150 > pdf(paste(folder_path, 'lin_regress_model/figures/adjr2_blocks_
151 quadratic_', response, '.pdf', sep=''))
152 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
153 1, 0, 0))
154 > plot(regfit.full, scale="adjr2", col=gray(seq(0, 0.9, len=11)))
155 > par(oldpar)
156 > dev.off()
157 > pdf(paste(folder_path, 'lin_regress_model/figures/Cp_blocks_
158 quadratic_', response, '.pdf', sep=''))
159 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
160 1, 0, 0))
```

```

159 > plot(regfit.full, scale="Cp", col=gray(seq(0, 0.9, len=11)))
160 > par(oldpar)
161 > dev.off()
162
163
164 > pdf(paste(folder_path, 'lin_regress_model/figures/bic_blocks_
      quadratic_', response, '.pdf', sep=''))
165 > oldpar <- par(cex.axis=1.3, cex.lab=2, mgp=c(3.65, 1, 0), oma=c(3,
      1, 0, 0))
166 > plot(regfit.full, scale="bic", col=gray(seq(0, 0.9, len=11)))
167 > par(oldpar)
168 > dev.off()
169
170
171
172 > pdf(paste(folder_path, 'lin_regress_model/figures/r2_adjr2_cp_bic_
      blocks_quadratic_', response, '.pdf', sep=''))
173
174 > oldpar <- par(cex.lab=1.5, las=1, mar=c(4.5, 5, 2, 2), mfrow=c(2, 2)
      , mgp=c(3, 0.8, 0))
175
176 > plot(regfit.full, scale="r2", col=gray(seq(0, 0.9, len=11)))
177 > plot(regfit.full, scale="adjr2", col=gray(seq(0, 0.9, len=11)))
178 > plot(regfit.full, scale="Cp", col=gray(seq(0, 0.9, len=11)))
179 > plot(regfit.full, scale="bic", col=gray(seq(0, 0.9, len=11)))
180
181 > par(oldpar)
182
183 > dev.off()
184
185
186
187 > coef(regfit.full, 8)
188 (Intercept)      Ha      PlWeek      GrowD      NHarv
189 -28.0076548    -4.0216345    3.0656727    0.3381491    2.0216073
190   avg_Rain    min_ET0    min_RelET  avg_minTemp_2
191  -4.7970673   -24.4683647   -0.5209359   -0.1033174
192
193
194
195
196
197
198 linear_regression_model <- function(terms, response, xlim_ylim_vec,
      training_df, validation_df, num_validate_cases, formula_to_use) {
199
200
201
202   # Training the model:
203
204   regression.crops <- lm(formula_to_use, data=training_df)
205
206   options(scipen=999)
207   print(summary(regression.crops))
208   options(scipen=0)
209
210
211
212   # Figures from model:
213
214   pdf(paste('regression_train_diag_', terms, '_', response, '.pdf',
      sep=''), height=10, width=10)
215

```

```

216   old_par <- par(cex.axis=1.3, las=1, mfrow=c(2, 2), oma=c(0, 0,
217             1.1, 0))
218   plot(regression.crops, main='', sub='')
219
220   dev.off()
221
222   par(old_par)
223
224
225
226   # Finding the validation error:
227
228   yhat <- predict(regression.crops, newdata=validation_df)
229
230   validation_diff_vec <- yhat - validation_df[, response]
231   validation_MSE <- mean(validation_diff_vec^2)
232
233   print(paste('\nvalidation MSE:', sprintf('%.4f', validation_MSE)))
234
235
236
237   pdf(paste('validation_lin_regress_', terms, '_', response, '.pdf',
238             sep=''))
239
240   old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.3, las
241             =1)
242
243   plot(x=validation_df[, response], y=yhat, xlim=xlim_ylim_vec, ylim
244         =xlim_ylim_vec, xlab='validation set response (y)', ylab=
245         expression(paste('model prediction (', hat(y), ')', sep='')))
246   abline(a=0, b=1)
247   legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
248         paste('validation MSE:', sprintf(switch(EXPR=response,
249           TotTonPerHa='%.2f', MedTonPerHa='%.4f'), validation_MSE)), cex
250         =1.5)
251
252   par(old_par)
253
254   dev.off()
255
256   if (response == 'TotTonPerHa') {
257     return(TotalTonPerHa_lin_regress_validation_MSE=validation_MSE
258           )
259   } else {
260     stopifnot(response == 'MedTonPerHa')
261     return(MedianTonPerHa_lin_regress_validation_MSE=validation_
262           MSE)
263   }
264 }
265
266
267 # median tonnes/hectare MLR model including only the linear terms:
268

```



```
269 linear_regression_model(terms='linear', response="MedTonPerHa", xlim_
      ylim_vec=c(0, 5), training_df=train_df, validation_df=validate_df,
      num_validate_cases=num_validate_cases, formula_to_use=
      MedianTonPerHa_linear_yield_formula)
270
271
272
273 # total tonnes/hectare MLR model including only the linear terms:
274
275 linear_regression_model(terms='linear', response="TotTonPerHa", xlim_
      ylim_vec=c(0, 200), training_df=train_df, validation_df=validate_
      df, num_validate_cases=num_validate_cases, formula_to_use=
      TotalTonPerHa_linear_yield_formula)
276
277
278
279 # median tonnes/hectare MLR model including the 11 linear and 4
      quadratic terms:
280
281 linear_regression_model(terms='lin_quad', response="MedTonPerHa", xlim
      _ylim_vec=c(0, 5), training_df=train_df, validation_df=validate_df
      , num_validate_cases=num_validate_cases, formula_to_use=
      MedianTonPerHa_linear_quadratic_yield_formula)
282
283
284 # total tonnes/hectare MLR model including the 11 linear and 4
      quadratic terms:
285
286 linear_regression_model(terms='lin_quad', response="TotTonPerHa", xlim
      _ylim_vec=c(0, 200), training_df=train_df, validation_df=validate_
      df, num_validate_cases=num_validate_cases, formula_to_use=
      TotalTonPerHa_linear_quadratic_yield_formula)
287
288
289
290 # median tonnes/hectare MLR model including the linear and quadratic
      terms chosen by best subset selection:
291
292 preds_to_use <- c('PlDensity', 'GrowD', 'NHarv', 'avg_minTemp', 'avg_
      Rain', 'min_RelET', 'PlWeek_2', 'avg_minTemp_2', 'min_ET0_2')
293 formula_to_use <- paste('MedTonPerHa ~', paste(preds_to_use, collapse=
      ' + '))
294 formula_to_use <- as.formula(formula_to_use)
295
296 linear_regression_model(terms='BSS', response="MedTonPerHa", xlim_ylim
      _vec=c(0, 5), training_df=train_df, validation_df=validate_df, num
      _validate_cases=num_validate_cases, formula_to_use=formula_to_use)
297
298
299 # total tonnes/hectare MLR model including the linear and quadratic
      terms chosen by best subset selection:
300
301 preds_to_use <- c('Ha', 'PlWeek', 'GrowD', 'NHarv', 'avg_Rain', 'min_
      ET0', 'min_RelET', 'avg_minTemp_2')
302 formula_to_use <- paste('TotTonPerHa ~', paste(preds_to_use, collapse=
      ' + '))
303 formula_to_use <- as.formula(formula_to_use)
304
305 linear_regression_model(terms='BSS', response="TotTonPerHa", xlim_ylim
      _vec=c(0, 200), training_df=train_df, validation_df=validate_df,
      num_validate_cases=num_validate_cases, formula_to_use=formula_to_
      use)
```

A.4 The lasso

```

1 library(glmnet) # for ridge regression and the lasso
2
3
4
5
6 ridge_lasso_regression_model <- function(response, xlim_ylim_vec,
7   alpha_param, training_df, validation_df, num_validate_cases,
8   formula_to_use, input_names_vec) {
9
10  tag <- switch(EXPR=alpha_param+1, ridge='ridge', lasso='lasso')
11  stopifnot(length(input_names_vec) %in% c(11, 15))
12
13
14
15  # Prepare the data sets:
16
17  xx_train <- model.matrix(object=formula_to_use, data=training_df)
18  [, -1]
19  yy_train_vec <- training_df[, response]
20  xx_validation <- model.matrix(object=formula_to_use, data=
21  validation_df)[, -1]
22  yy_validation_vec <- validation_df[, response]
23
24  # Standardise the training inputs:
25
26  standardised_train_pred <- scale(xx_train, center=TRUE, scale=TRUE
27  )
28  standardised_mean <- attr(x=standardised_train_pred, which='scaled
29  :center')
30  standardised_scale <- attr(x=standardised_train_pred, which='
31  scaled:scale')
32
33  # Standardise the validation inputs:
34
35  standardised_validation_pred <- t(apply(X=xx_validation, MARGIN=1,
36  FUN=function(x) (x - standardised_mean)/standardised_scale))
37
38  # Find the best lambda:
39
40  grid_vec <- 10^seq(2, -6, length=100)
41  cv.out <- cv.glmnet(x=standardised_train_pred, y=yy_train_vec,
42  alpha=alpha_param, lambda=grid_vec, maxit=800000)
43  # alpha=0 means that ridge regression is fit
44  # alpha=1 means that the lasso is fit
45
46  pdf(paste(tag, '_cv_', response, '.pdf', sep=''))
47
48  old_par <- par(cex.axis=1.3, cex.lab=2, las=1, mar=c(5, 5.5, 5, 2)
49  , mgp=c(3.5, 1, 0))
50
51  plot(cv.out, xlab='')
52  mtext(text=expression(paste('ln(', lambda, ')', sep='')), side=1,
53  line=3.5, cex=2)

```

```

50  mtext(text='number of non-zero coefficients (df)', side=3, line=3,
      cex=2)
51
52  par(old_par)
53
54  dev.off()
55
56
57  lam_CV_min <- cv.out$lambda.min
58
59
60  print(paste('\nlambda:', sprintf('%.6f', lam_CV_min)))
61
62
63
64  # Train the model:
65
66  ridge_lasso_mod <- glmnet(x=standardised_train_pred, y=yy_train_
      vec, alpha=alpha_param, lambda=grid_vec, maxit=800000)
67  # alpha=0 means that ridge regression is fit
68  # alpha=1 means that the lasso is fit
69  print(coef(ridge_lasso_mod, s=lam_CV_min))
70
71
72  cairo_pdf(paste(tag, '_regress_full_mod_', response, '.pdf', sep=
      '), width=9, height=5)
73
74  old_par <- par(las=1, mfrow=c(1, 2), mar=c(4.5, 4.5, 4, 4))
75
76  plot(ridge_lasso_mod, xlab='')
77  abline(v=sum(abs(coef(ridge_lasso_mod, s=lam_CV_min)[2:length(coef
      (ridge_lasso_mod, s=lam_CV_min))])), lty=2)
78  axis(side=1, at=sum(abs(coef(ridge_lasso_mod, s=lam_CV_min)[2:
      length(coef(ridge_lasso_mod, s=lam_CV_min))])), labels=paste(
      sprintf('%.4f', sum(abs(coef(ridge_lasso_mod, s=lam_CV_min)[2:
      length(coef(ridge_lasso_mod, s=lam_CV_min))])), sep=''), tcl
      =-0.3, mgp=c(3, 0.1, 0), cex.axis=0.6)
79  coef_vals_df <- coef(ridge_lasso_mod)
80  coef_vals_vec <- coef_vals_df[-1, ncol(coef_vals_df)]
81  axis(side=4, at=coef_vals_vec, line=-.5, label=input_names_vec,
      las=1, tick=FALSE, cex.axis=0.5)
82  mtext(text=expression(paste("\u2113"[1], sep='')), side=1, line
      =2.5)
83  mtext(text='number of non-zero coefficients (df)', side=3, line
      =2.5)
84
85  plot(ridge_lasso_mod, xvar='lambda', xlab='')
86  abline(v=log(lam_CV_min), lty=2)
87  axis(side=4, at=coef_vals_vec, line=-.5, label=input_names_vec,
      las=1, tick=FALSE, cex.axis=0.5)
88  axis(side=1, at=log(lam_CV_min), labels=paste(sprintf('%.4f', log(
      lam_CV_min)), ' = ln(', sprintf('%.6f', lam_CV_min), ')', sep=
      ''), tcl=-0.3, mgp=c(3, 0.1, 0), cex.axis=0.6)
89  mtext(text=expression(paste('ln(', lambda, ')', sep='')), side=1,
      line=2.5)
90  mtext(text='number of non-zero coefficients (df)', side=3, line
      =2.5)
91
92  par(old_par)
93
94  dev.off()
95
96

```

```

97
98 # Find the validation error:
99
100 ridge_lasso_pred_vec <- predict(ridge_lasso_mod, s=lam_CV_min,
    newx=standardised_validation_pred)
101
102 validation_diff_vec <- ridge_lasso_pred_vec - yy_validation_vec
103 validation_MSE <- mean(validation_diff_vec^2)
104
105 print(paste('\nvalidation MSE:', sprintf('%.4f', validation_MSE)))
106
107
108 pdf(paste(validation_', tag, '_regress_full_', response, '.pdf',
    sep=''))
109
110 old_par <- par(mar=c(4.5, 7, 2, 4), cex.lab=2, cex.axis=1.3, las
    =1)
111
112 plot(x=yy_validation_vec, y=ridge_lasso_pred_vec, xlim=xlim_ylim_
    vec, ylim=xlim_ylim_vec, xlab='validation set response (y)',
    ylab=expression(paste('model prediction (', hat(y), ')', sep='
    ')))
113 abline(a=0, b=1)
114 legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
    paste('validation MSE:', sprintf(switch(EXPR=response,
    TotTonPerHa='%.2f', MedTonPerHa='%.4f'), validation_MSE)), cex
    =1.5)
115
116 par(old_par)
117
118 dev.off()
119
120
121
122 if (response == 'TotTonPerHa') {
123
124     return(TotalTonPerHa_ridge_lasso_regress_validation_MSE=
        validation_MSE)
125
126 } else {
127
128     stopifnot(response == 'MedTonPerHa')
129     return(MedianTonPerHa_ridge_lasso_regress_validation_MSE=
        validation_MSE)
130
131 }
132
133 }
134
135
136
137
138 # median tonnes/hectare lasso model:
139
140 ridge_lasso_regression_model(response="MedTonPerHa", xlim_ylim_vec=c
    (0, 5), alpha_param=1, training_df=train_df, validation_df=
    validate_df, num_validate_cases=num_validate_cases, formula_to_use
    =MedianTonPerHa_linear_quadratic_yield_formula, input_names_vec=c(
    input_linear_column_names_vec, input_quadratic_column_names_vec))
141
142
143
144 # total tonnes/hectare lasso model:

```

```

145
146 ridge_lasso_regression_model(response="TotTonPerHa", xlim_ylim_vec=c
      (0, 200), alpha_param=1, training_df=train_df, validation_df=
      validate_df, num_validate_cases=num_validate_cases, formula_to_use
      =TotalTonPerHa_linear_quadratic_yield_formula, input_names_vec=c(
      input_linear_column_names_vec, input_quadratic_column_names_vec))
147
148
149
150 # median tonnes/hectare ridge regression model:
151
152 ridge_lasso_regression_model(response="MedTonPerHa", xlim_ylim_vec=c
      (0, 5), alpha_param=0, training_df=train_df, validation_df=
      validate_df, num_validate_cases=num_validate_cases, formula_to_use
      =MedianTonPerHa_linear_quadratic_yield_formula, input_names_vec=c(
      input_linear_column_names_vec, input_quadratic_column_names_vec))
153
154
155 # total tonnes/hectare ridge regression model:
156
157 ridge_lasso_regression_model(response="TotTonPerHa", xlim_ylim_vec=c
      (0, 200), alpha_param=0, training_df=train_df, validation_df=
      validate_df, num_validate_cases=num_validate_cases, formula_to_use
      =TotalTonPerHa_linear_quadratic_yield_formula, input_names_vec=c(
      input_linear_column_names_vec, input_quadratic_column_names_vec))

```

A.5 Regression trees

```

1 library(caret) # for parameter optimisation by cross-validation
2 library(rpart) # for building regression trees
3 library(rpart.plot) # for drawing regression trees
4 library(MASS)
5
6
7
8
9 # determine optimum cp parameter for each model:
10
11 optimum_cp_parameter <- function(response, formula_to_use) {
12
13
14
15   # condidate settings for the cp parameter:
16
17   my.df <- data.frame(cp=c(0.01, 0.005, 0.001, 0.0005, 0.0001,
18     0.00005, 0.00001))
19
20
21   # perform 10-fold cross-validation:
22
23   ctrl <- trainControl(method="cv", number=10) # to perform 10-fold
      CV
24   cp_results <- train(form=formula_to_use, data=train_df, method="
      rpart", trControl=ctrl, tuneGrid=my.df)
25
26
27   pdf(paste('tree_CV_cp_', response, '.pdf', sep=''))

```

```
28
29 plot(cp_results, xlab=list(label="complexity parameter", cex=2),
      ylab=list(cex=2), scales=list(cex=1.3), type=c('l', 'p'),
      aspect="fill", par.settings=simpleTheme(col.line='black', lwd
      =2, pch=19, col.points='black', cex=1))
30
31 dev.off()
32
33
34 print(cp_results)
35
36
37 print(cp_results$bestTune)
38
39
40 }
41
42
43
44 # median tonnes/hectare optimum cp parameter:
45
46 optimum_cp_parameter(response="MedTonPerHa", formula_to_use=
      MedianTonPerHa_linear_yield_formula)
47
48
49 # total tonnes/hectare optimum cp parameter:
50
51 optimum_cp_parameter(response="TotTonPerHa", formula_to_use=
      TotalTonPerHa_linear_yield_formula)
52
53
54
55
56 # train the full tree models:
57
58 full_regression_tree <- function(response, xlim_ylim_vec, training_df,
      validation_df, num_validate_cases, formula_to_use) {
59
60
61
62 tree.crops <- rpart(formula=formula_to_use, data=training_df,
      method='anova', cp=switch(EXPR=response, MedTonPerHa=5e-04,
      TotTonPerHa=0.001))
63 print(tree.crops)
64 summary(tree.crops, cp=0.01)
65
66
67
68 # plot full tree:
69
70 pdf(paste('tree_train_full_model_', response, '.pdf', sep=''))
71
72 rpart.plot(tree.crops, extra=101, box.palette="Blues", shadow.col=
      "gray", nn=TRUE)
73
74 dev.off()
75
76
77 pdf(paste('regress_trees/figures/tree_train_full_model_rel_error_
      xerror_', response, '.pdf', sep=''))
78
79 rsq.rpart(tree.crops)
80
```

```
81 dev.off()
82
83
84
85 # Finding the validation error of full tree:
86
87 yhat <- predict(tree.crops, newdata=validation_df)
88
89 validation_diff_vec <- yhat - validation_df[, response]
90 full_tree_validation_MSE <- mean(validation_diff_vec^2)
91
92 print(paste('\nvalidation MSE of full tree:', sprintf('%.4f', full_
93 _tree_validation_MSE)))
94
95 pdf(paste('validation_full_tree_', response, '.pdf', sep=''))
96
97 old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.3, las
98 =1)
99
100 plot(x=validation_df[, response], y=yhat, xlim=xlim_ylim_vec, ylim
101 =xlim_ylim_vec, xlab='validation set response (y)', ylab=
102 expression(paste('model prediction (', hat(y), ')', sep='')))
103 abline(a=0, b=1)
104 legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
105 paste('validation MSE:', sprintf(switch(EXPR=response,
106 TotTonPerHa='%.2f', MedTonPerHa='%.4f'), full_tree_validation_
107 MSE)), cex=1.5)
108
109 par(old_par)
110
111 dev.off()
112
113 if (response == 'TotTonPerHa') {
114     return(list(TotalTonPerHa_full_tree_validation_MSE=full_tree_
115 validation_MSE, full_tree_model=tree.crops))
116 } else {
117     stopifnot(response == 'MedTonPerHa')
118     return(list(MedianTonPerHa_full_tree_validation_MSE=full_tree_
119 validation_MSE, full_tree_model=tree.crops))
120 }
121 }
122
123
124
125 # median tonnes/hectare full regression tree model:
126
127 full_regression_tree(response="MedTonPerHa", xlim_ylim_vec=c(0, 5),
128 training_df=train_df, validation_df=validate_df, num_validate_
129 cases=num_validate_cases, formula_to_use=MedianTonPerHa_linear_
130 yield_formula)
131
132 # total tonnes/hectare full regression tree model:
```

```
132 full_regression_tree(response="TotTonPerHa", xlim_ylim_vec=c(0, 200),
133   training_df=train_df, validation_df=validate_df, num_validate_
134   cases=num_validate_cases, formula_to_use=TotalTonPerHa_linear_
135   yield_formula)
136
137 # train the pruned tree models:
138
139 pruned_regression_tree <- function(response, xlim_ylim_vec, full_tree_
140   model, validation_df, num_validate_cases) {
141
142   min_xerror_pos <- which.min(full_tree_model$cptable[, "xerror"])
143   min_xerror <- full_tree_model$cptable[min_xerror_pos, "xerror"]
144   stopifnot(min_xerror == min(full_tree_model$cptable[, "xerror"]))
145   min_xerror_st_err <- full_tree_model$cptable[min_xerror_pos, "xstd
146     "]
147   max_allowable_xerror_val <- min_xerror + min_xerror_st_err
148   xerror_val_below_max_pos <- min(which(full_tree_model$cptable[, "
149     xerror"] <= max_allowable_xerror_val))
150   xerror_val_below_max_val <- full_tree_model$cptable[xerror_val_
151     below_max_pos, "xerror"]
152   cp_parameter <- full_tree_model$cptable[xerror_val_below_max_pos,
153     "CP"]
154
155   pdf(paste('tree_train_cv_', response, '.pdf', sep=''))
156
157   old_par <- par(ann=FALSE, cex.lab=2, cex.axis=1.3, las=1, mar=c
158     (4.5, 5.5, 4.5, 4))
159
160   plotcp(x=full_tree_model, minline=TRUE, upper="size") # plot of
161   the cross-validation results of an rpart object
162   abline(h=max_allowable_xerror_val, lty="dotted")
163   title(xlab='complexity parameter')
164   title(ylab='mean relative CV prediction error')
165   mtext('number of terminal nodes', side=3, line=2.5, cex=2)
166
167   par(old_par)
168
169   dev.off()
170
171   prune.crops <- prune(tree=full_tree_model, cp=cp_parameter)
172
173   # Plot pruned tree:
174
175   pdf(paste('tree_train_pruned_', response, '.pdf', sep=''))
176
177   rpart.plot(prune.crops, extra=101, box.palette="Blues", shadow.col
178     ="gray", nn=TRUE)
179
180   dev.off()
181
182   # Finding the validation error of pruned tree:
183
184   yhat <- predict(prune.crops, newdata=validation_df)
```



```

184
185 validation_diff_vec <- yhat - validation_df[, response]
186 pruned_tree_validation_MSE <- mean(validation_diff_vec^2)
187
188
189 print(paste('\n\nThe complexity parameter used to prune the tree:',
             sprintf('%.4f', cp_parameter), '\n\n\n'))
190
191
192 print(prune.crops)
193
194
195 print(paste('\nvalidation MSE of pruned tree:', sprintf('%.4f',
             pruned_tree_validation_MSE)))
196
197
198 pdf(paste('validation_pruned_tree_', response, '.pdf', sep=''))
199
200 old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.3, las
             =1)
201
202 plot(x=validation_df[, response], y=yhat, xlim=xlim_ylim_vec, ylim
             =xlim_ylim_vec, xlab='validation set response (y)', ylab=
             expression(paste('model prediction (', hat(y), ')', sep='')))
203 abline(a=0, b=1)
204 legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
             paste('validation MSE:', sprintf(switch(EXPR=response,
             TotTonPerHa='%.2f', MedTonPerHa='%.4f'), pruned_tree_
             validation_MSE)), cex=1.5)
205
206 par(old_par)
207
208 dev.off()
209
210
211
212 if (response == 'TotTonPerHa') {
213
214     return(TotalTonPerHa_pruned_tree_validation_MSE=pruned_tree_
             validation_MSE)
215
216 } else {
217
218     stopifnot(response == 'MedTonPerHa')
219     return(MedianTonPerHa_pruned_tree_validation_MSE=pruned_tree_
             validation_MSE)
220
221 }
222
223
224 }
225
226
227
228 # median tonnes/hectare pruned regression tree model:
229
230 pruned_regression_tree(response="MedTonPerHa", xlim_ylim_vec=c(0, 5),
             full_tree_model=tree.crops, validation_df=validate_df, num_
             validate_cases=num_validate_cases)
231
232
233 # total tonnes/hectare pruned regression tree model:
234

```

```
235 pruned_regression_tree(response="TotTonPerHa", xlim_ylim_vec=c(0, 200)
    , full_tree_model=tree.crops, validation_df=validate_df, num_
      validate_cases=num_validate_cases)
```

A.6 Bagged regression trees

```
1 library(e1071) # for parameter optimisation by cross-validation
2
3
4
5
6 # determine optimum nodesize parameter for each model:
7
8 response <- 'MedTonPerHa'
9 input_column_names_vec <- c('Ha', 'PlDensity', 'PlWeek', 'GrowD', '
    NHarv', 'avg_minTemp', 'avg_maxRH', 'avg_WindS', 'avg_Rain', 'min_
    ETO', 'min_ReLET')
10 train_data_pred <- train_df[, input_column_names_vec]
11 train_data_response <- train_df[, response]
12
13 tune.randomForest(y=train_data_response, x=train_data_pred, nodesize=c
    (1, 3, 5, 10, 15, 20, 30), mtry=11, ntree=500)
14 Parameter tuning of 'randomForest':
15
16 - sampling method: 10-fold cross validation
17
18 - best parameters:
19   nodesize mtry ntree
20     5     11   500
21
22 - best performance: 0.229911
23
24
25
26
27 response <- 'TotTonPerHa'
28 train_data_response <- yield_clim_train_df[, response]
29
30 tune.randomForest(y=train_data_response, x=train_data_pred, nodesize=c
    (1, 3, 5, 10, 15, 20, 30), mtry=11, ntree=500)
31 Parameter tuning of 'randomForest':
32
33 - sampling method: 10-fold cross validation
34
35 - best parameters:
36   nodesize mtry ntree
37     3     11   500
38
39 - best performance: 237.9592
40
41
42
43
44
45 library(randomForest) # for bagging and random forests
46
47
48
```

```

49
50 # train the bagged models:
51
52 random_forests_model <- function(response, xlim_ylim_vec, training_df,
   num_var, nodesize, validation_df, num_validate_cases, formula_to_
   use) {
53
54
55
56   # training the model:
57
58   rand.for.crops <- randomForest(formula_to_use, data=training_df,
   mtry=num_var, nodesize=nodesize, importance=TRUE)
59   print(rand.for.crops)
60
61
62   pdf(paste('rf_num_trees_', response, '.pdf', sep=''))
63
64   old_par <- par(cex.lab=2, cex.axis=1.3, las=1, mar=c(4.5, 5.5, 2,
   4))
65
66   plot(x=1:rand.for.crops$ntree, y=rand.for.crops$mse, type='l', lwd
   =2, xlab="number of trees", ylab="", las=1)
67   title(ylab="MSE of OOB observations", line=3.5)
68
69   par(old_par)
70
71   dev.off()
72
73
74
75   # importance of variables:
76
77   print(importance(rand.for.crops))
78
79
80   pdf(paste('importance_train_rf_', response, '.pdf', sep=''))
81
82   varImpPlot(rand.for.crops, main="", cex=1.3, pch=19, lcolor="
   gray55")
83
84   dev.off()
85
86
87
88   # partial dependence plots of inputs:
89
90   draw_partial_dependence_plots <- function(input_index, input_names
   _vec, var_types, bag.crops, training_df, ylim_numeric_vec,
   ylim_factor_vec) {
91
92     input_name <- input_names_vec[input_index]
93     var_type <- var_types[input_index]
94     do.call('partialPlot', list(x=bag.crops, pred.data=training_df
   , x.var=input_name, xlab=input_name, ylab=switch(EXPR=
   response, TotTonPerHa='total harvest density', MedTonPerHa
   ='median harvest density'), main='', ylim=switch(EXPR=var_
   type, numeric=ylim_numeric_vec, factor=ylim_factor_vec)))
95
96   }
97
98

```

```

99     pdf(paste('bag_partial_dependence_', terms, '_', response, '.pdf',
100            sep=''))
101     old_par <- par(mfrow=c(3, 4), las=1)
102
103     var_types <- attr(x=bag.crops$terms, which="dataClasses")[-1]
104     ylim_numeric_vec <- switch(EXPR=response, TotTonPerHa=c(70, 105),
105            MedTonPerHa=c(2.1, 2.7))
106     ylim_factor_vec <- switch(EXPR=response, TotTonPerHa=c(0, 100),
107            MedTonPerHa=c(0, 2.5))
108     apply(X=matrix(1:length(input_names_vec), ncol=1), MARGIN=1, FUN=
109            draw_partial_dependence_plots, input_names_vec, var_types, bag
110            .crops, training_df, ylim_numeric_vec, ylim_factor_vec)
111
112     par(old_par)
113
114     dev.off()
115
116     # Finding the validation error:
117     yhat.rf <- predict(rand.for.crops, newdata=validation_df)
118     validation_diff_vec <- yhat.rf - validation_df[, response]
119     validation_MSE <- mean(validation_diff_vec^2)
120
121     print(paste('\nvalidation MSE:', sprintf('%.4f', validation_MSE)))
122
123
124     pdf(paste('validation_rf_', response, '.pdf', sep=''))
125
126     old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.3, las
127            =1)
128
129     plot(x=validation_df[, response], y=yhat.rf, xlim=xlim_ylim_vec,
130            ylim=xlim_ylim_vec, xlab='validation set response (y)', ylab=
131            expression(paste('model prediction (', hat(y), ')', sep='')))
132     abline(a=0, b=1)
133     legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
134            paste('validation MSE:', sprintf(sprintf(switch(EXPR=response,
135            TotTonPerHa='%.2f', MedTonPerHa='%.4f'), validation_MSE)),
136            cex=1.5)
137
138     par(old_par)
139
140     dev.off()
141
142     if (response == 'TotTonPerHa') {
143         return(TotalTonPerHa_rand_forests_validation_MSE=validation_
144            MSE)
145     } else {
146         stopifnot(response == 'MedTonPerHa')
147         return(MedianTonPerHa_rand_forests_validation_MSE=validation_
148            MSE)
149     }

```

```

149
150
151 }
152
153
154
155 # median tonnes/hectare bagging model:
156
157 random_forests_model(response="MedTonPerHa", xlim_ylim_vec=c(0, 5),
  training_df=train_df, num_var=11, nodesize=5, validation_df=
  validate_df, num_validate_cases=num_validate_cases, formula_to_use
  =MedianTonPerHa_linear_yield_formula)
158
159
160 # total tonnes/hectare bagging model:
161
162 random_forests_model(response="TotTonPerHa", xlim_ylim_vec=c(0, 200),
  training_df=train_df, num_var=11, nodesize=3, validation_df=
  validate_df, num_validate_cases=num_validate_cases, formula_to_use
  =TotalTonPerHa_linear_yield_formula)

```

A.7 Random forests

```

1 library(e1071) # for parameter optimisation by cross-validation
2
3
4
5
6 # determine optimum nodesize and mtry parameter for each model:
7
8 response <- 'MedTonPerHa'
9 input_column_names_vec <- c('Ha', 'PlDensity', 'PlWeek', 'GrowD', '
  NHarv', 'avg_minTemp', 'avg_maxRH', 'avg_WindS', 'avg_Rain', 'min_
  ETO', 'min_RelET')
10 train_data_pred <- train_df[, input_column_names_vec]
11 train_data_response <- train_df[, response]
12
13 tune.randomForest(y=train_data_response, x=train_data_pred, nodesize=c
  (1, 3, 5, 10, 15, 20, 30), mtry=1:11, ntree=500)
14 Parameter tuning of 'randomForest':
15
16 - sampling method: 10-fold cross validation
17
18 - best parameters:
19   nodesize mtry ntree
20     3      3   500
21
22 - best performance: 0.2361649
23
24
25
26
27 response <- 'TotTonPerHa'
28 train_data_response <- yield_clim_train_df[, response]
29
30 tune.randomForest(y=train_data_response, x=train_data_pred, nodesize=c
  (1, 3, 5, 10, 15, 20, 30), mtry=1:11, ntree=500)
31 Parameter tuning of 'randomForest':

```

```

32
33 - sampling method: 10-fold cross validation
34
35 - best parameters:
36   nodesize mtry ntree
37     1      5    500
38
39 - best performance: 231.4638
40
41
42
43
44
45 # median tonnes/hectare random forests model:
46
47 random_forests_model(response="MedTonPerHa", xlim_ylim_vec=c(0, 5),
48   training_df=train_df, num_var=3, nodesize=3, validation_df=
49   validate_df, num_validate_cases=num_validate_cases, formula_to_use
50   =MedianTonPerHa_linear_yield_formula)
51
52 # total tonnes/hectare random forests model:
53
54 random_forests_model(response="TotTonPerHa", xlim_ylim_vec=c(0, 200),
55   training_df=train_df, num_var=5, nodesize=1, validation_df=
56   validate_df, num_validate_cases=num_validate_cases, formula_to_use
57   =TotalTonPerHa_linear_yield_formula)

```

A.8 Boosted regression trees

```

1 library(dismo) # for boosting
2
3
4
5
6 # finding the column numbers of the predictors and response:
7
8 get_col_numbers_from_train_df <- function(column_name, train_df) {
9
10   col_numbers_vec <- which(names(train_df) == column_name)
11
12   return(col_numbers_vec)
13 }
14
15
16
17 input_column_names_vec <- c('Ha', 'PlDensity', 'PlWeek', 'GrowD', '
18   NHarv', 'avg_minTemp', 'avg_maxRH', 'avg_WindS', 'avg_Rain', 'min_
19   ETO', 'min_ReLET')
20 stopifnot(is.atomic(input_column_names_vec))
21 stopifnot(length(input_column_names_vec) == 11)
22 input_column_numbers_all_vec <- apply(X=matrix(data=input_column_names
23   _vec, ncol=1), MARGIN=1, FUN=get_col_numbers_from_train_df, train_

```

```

24 response_column_names_list <- list(TotalTonPerHa='TotTonPerHa',
   MedianTonPerHa='MedTonPerHa')
25 stopifnot(length(response_column_names_list) == 2)
26 response_column_numbers_list <- lapply(X=response_column_names_list,
   FUN=get_col_numbers_from_train_df, yield_clim_train_df)
27 stopifnot(length(response_column_names_list) == length(response_column
   _numbers_list))
28
29
30
31
32 boosted_regression_trees <- function(response, xlim_ylim_vec, train_df
   , validation_df, input_columns_vec, response_column, learning_rate
   =0.005, bag_fraction=0.5, tree_complexity_vec=1:10, max_num_trees
   =20000) {
33
34
35
36   sink(file='boost_regress_trees_output.tex')
37   cat(noquote(paste('\nlearning rate:', learning_rate, '\n')))
38   cat(noquote(paste('\nbag fraction:', bag_fraction, '\n')))
39   cat(noquote(paste('\ntree complexity settings tried:', '\n')))
40   cat(noquote(paste(paste(tree_complexity_vec, collapse=', '), '\n'
   )))
41
42
43
44   optimise_tree_complexity <- function(tree_complexity_index) {
45
46     cat(noquote(paste('\n\ntree complexity:', tree_complexity_
   index, '\n')))
47
48     pdf(paste('tree_complexity_boost_', tree_complexity_index, '_'
   , response, '.pdf', sep=''))
49     old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.2,
   las=1)
50     boost.crops <- gbm.step(data=train_df, gbm.y=response_column,
   gbm.x=input_columns_vec, family="gaussian", tree.
   complexity=tree_complexity_index, learning.rate=learning_
   rate, bag.fraction=bag_fraction, max.trees=20000)
51     par(old_par)
52     dev.off()
53
54     return(boost.crops)
55
56   }
57
58
59   # Training models with different interaction depths:
60
61   tree_complexity_list <- as.list(tree_complexity_vec)
62   model_list <- lapply(X=tree_complexity_list, FUN=optimise_tree_
   complexity)
63   stopifnot(is.recursive(model_list))
64   stopifnot(length(model_list) == length(tree_complexity_vec))
65
66
67
68   # Choosing the model with the smallest CV error:
69
70   train_deviance_list <- lapply(X=model_list, FUN=function(x) x$cv.
   statistics$deviance.mean)
71   stopifnot(is.list(train_deviance_list))

```

```

72  stopifnot(length(train_deviance_list) == length(tree_complexity_
73  vec))
74  num_trees_list <- lapply(X=model_list, FUN=function(x) x$n.trees)
75  stopifnot(is.list(num_trees_list))
76  stopifnot(length(num_trees_list) == length(tree_complexity_vec))
77  min_train_deviance_1st_pos <- which.min(train_deviance_list)
78  min_train_deviance_tree_complexity <- tree_complexity_vec[min_
79  train_deviance_1st_pos]
80  min_train_deviance <- train_deviance_list[[min_train_deviance_1st_
81  pos]]
82  min_train_deviance_num_trees <- num_trees_list[[min_train_deviance
83  _1st_pos]]
84  best_full_model <- model_list[[min_train_deviance_1st_pos]]
85
86  cat(noquote(paste('\n\ntree complexity of the optimum model:', min
87  _train_deviance_tree_complexity)))
88  cat(noquote(paste('\n\nminimum training deviance obtained by the
89  optimum model:', sprintf('%.4f', min_train_deviance))))
90  cat(noquote(paste('\n\nnumber of trees chosen for the optimum
91  model:', min_train_deviance_num_trees, '\n\n')))
92
93  # Importance of the different input variables:
94
95  pdf(paste('importance_best_model_boost_', response, '.pdf', sep='
96  '))
97  old_par <- par(mar=c(4.5, 6.5, 2, 4), cex.lab=2, las=1)
98  print(summary(best_full_model))
99  par(old_par)
100 dev.off()
101
102 # Partial dependence plots of the full boosted model:
103
104 pdf(paste('boost_full_partial_dependence_', response, '.pdf', sep=
105 ' '))
106 old_par <- par(las=1)
107 gbm.plot(best_full_model, n.plots=11, write.title=FALSE, y.label=
108 switch(EXPR=response, TotTonPerHa='total harvest density',
109 MedTonPerHa='median harvest density'))
110 par(old_par)
111 dev.off()
112
113 pdf(paste('boost_full_partial_dependence_dots_', response, '.pdf',
114 sep=' '))
115 old_par <- par(cex.main=0.9, las=1)
116 gbm.plot.fits(best_full_model)
117 par(old_par)
118 dev.off()
119
120 # Finding the validation error of the full model:
121
122 yhat.boost <- predict(best_full_model, newdata=validation_df, n.
123 trees=best_full_model$gbm.call$best.trees, type="response")
124 validation_diff_vec <- yhat.boost - validation_df[, response]
125 validation_MSE_full <- mean(validation_diff_vec^2)

```



```

121   cat(noquote(paste('\nvalidation MSE of full boosted model:',
122                   sprintf('%4f', validation_MSE_full))))
123
124   pdf(paste('validation_boosting_full_', response, '.pdf', sep=''))
125   old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.3, las
126             =1)
127   plot(x=validation_df[, response], y=yhat.boost, xlim=xlim_ylim_vec
128         , ylim=xlim_ylim_vec, xlab='validation set response (y)', ylab
129         =expression(paste('model prediction (', hat(y), ')', sep='')))
130   abline(a=0, b=1)
131   legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
132         paste('validation MSE:', sprintf(switch(EXPR=response,
133             TotTonPerHa='%2f', MedTonPerHa='%4f'), validation_MSE_full))
134         , cex=1.5)
135   par(old_par)
136   dev.off()
137
138   # Determining whether any of the variables in the full model can
139   # be dropped:
140
141   pdf(paste('reduced_best_model_', response, '.pdf', sep=''))
142   old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.2, las
143             =1)
144   boost.crops.simp <- gbm.simplify(best_full_model)
145   par(old_par)
146   dev.off()
147
148   num_vars_to_drop <- which.min(boost.crops.simp$deviance.summary$
149     mean)
150   cat(noquote(paste('\n\noptimum number of input variables to drop:'
151                   , num_vars_to_drop)))
152
153   # Simplifying the full model by omitting variables:
154
155   pdf(paste('reduced_model_', response, '.pdf', sep=''))
156   old_par <- par(mar=c(4.5, 6.5, 2, 4), cex.lab=2, las=1)
157   reduced_model <- gbm.step(data=train_df, gbm.x=boost.crops.simp$
158     pred.list[[num_vars_to_drop]], gbm.y=response_column, family="
159     gaussian", tree.complexity=optimum_tree_complexity, learning.
160     rate=learning_rate, bag.fraction=bag_fraction)
161   par(old_par)
162   dev.off()
163
164   # Importance of the different input variables of the reduced model
165   # :
166
167   pdf(paste('importance_reduced_model_boost_', response, '.pdf', sep
168           =''))
169   old_par <- par(mar=c(4.5, 6.5, 2, 4), cex.lab=2, las=1)
170   print(summary(reduced_model))
171   par(old_par)
172   dev.off()

```

```

168 # Partial dependence plots of the reduced boosted model:
169
170 pdf(paste('boost_reduced_partial_dependence_', response, '.pdf',
171          sep=''))
171 old_par <- par(las=1)
172 gbm.plot(reduced_model, n.plots=11-num_vars_to_drop, write.title=
173          FALSE, y.label=switch(EXPR=response, TotTonPerHa='total
174          harvest density', MedTonPerHa='median harvest density'))
173 par(old_par)
174 dev.off()
175
176 pdf(paste('boost_reduced_partial_dependence_dots_', response, '.
177          pdf', sep=''))
177 old_par <- par(cex.main=0.9, las=1)
178 gbm.plot.fits(reduced_model)
179 par(old_par)
180 dev.off()
181
182
183
184 # Finding the validation error of the reduced model:
185
186 yhat.boost <- predict(reduced_model, newdata=validation_df, n.
187          trees=reduced_model$gbm.call$best.trees, type="response")
187 validation_diff_vec <- yhat.boost - validation_df[, response]
188 validation_MSE_reduced <- mean(validation_diff_vec^2)
189 cat(noquote(paste('\nvalidation MSE of reduced boosted model:',
190          sprintf('%.4f', validation_MSE_reduced))))
190 sink()
191
192
193 pdf(paste('validation_boosting_reduced_', response, '.pdf', sep='
194          '))
194 old_par <- par(mar=c(4.5, 5.5, 2, 4), cex.lab=2, cex.axis=1.3, las
195          =1)
195 plot(x=validation_df[, response], y=yhat.boost, xlim=xlim_ylim_vec
196          , ylim=xlim_ylim_vec, xlab='validation set response (y)', ylab
197          =expression(paste('model prediction (', hat(y), ')', sep='')))
196 abline(a=0, b=1)
197 legend("topleft", legend=c(expression(hat(y) == y)), lty=1, title=
198          paste('validation MSE:', sprintf(switch(EXPR=response,
199          TotTonPerHa='%.2f', MedTonPerHa='%.4f'), validation_MSE_
200          reduced)), cex=1.5)
198 par(old_par)
199 dev.off()
200
201
202
203 if (response == 'TotTonPerHa') {
204
205     return(TotalTonPerHa_boosted_validation_MSE_full=validation_
206            MSE_full, TotalTonPerHa_boosted_validation_MSE_reduced=
207            validation_MSE_reduced)
208
209 } else {
210
211     stopifnot(response == 'MedTonPerHa')
212     return(MedianTonPerHa_boosted_validation_MSE_full=validation_
213            MSE_full, MedianTonPerHa_boosted_validation_MSE_reduced=
214            validation_MSE_reduced)
215
216 }

```

```
214 }
215 }
216
217
218
219 # median tonnes/hectare boosting model:
220
221 boosted_regression_trees(response="MedTonPerHa", xlim_ylim_vec=c(0, 5)
    , train_df=train_df, validation_df=validate_df, input_columns_vec=
    input_column_numbers_all_vec, response_column=response_column_
    names_list$MedTonPerHa)
222
223
224 # total tonnes/hectare boosting model:
225
226 boosted_regression_trees(response="TotTonPerHa", xlim_ylim_vec=c(0,
    200), train_df=train_df, validation_df=validate_df, input_columns_
    vec=input_column_numbers_all_vec, response_column=response_column_
    names_list$TotTonPerHa)
```

List of References

- Ahrens, C.D. (2008). *Meteorology Today: An Introduction to Weather, Climate, and the Environment*. 9th edn. Brooks/Cole, Cengage Learning.
- Allen, R.G., Pereira, L.S., Raes, D. and Smith, M. (1998). Crop evapotranspiration: guidelines for computing crop water requirements. Tech. Rep. 56, Food and Agriculture Organisation.
- Barrow, E. and Semenov, M. (1995). Climate-change scenarios with high spatial and temporal resolution for agricultural applications. *Forestry*, vol. 84, pp. 349–360.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, vol. 24, pp. 123–140.
- Breiman, L. (2001). Random forests. *Machine Learning*, vol. 45, pp. 5–32.
- Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth, New York.
- Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics*, vol. 7, pp. 1–26.
- Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. No. 57 in Monographs on Statistics and Applied Probability. Chapman and Hall.
- Ehret, D.L., Hill, B.D., Helmer, T. and Edwards, D.R. (2011). Neural network modeling of greenhouse tomato yield, growth and water use from automated crop monitoring data. *Computers and Electronics in Agriculture*, vol. 79, pp. 82–89.
- Elith, J. and Leathwick, J. (2016). *Boosted Regression Trees for ecological modeling*. Available at: <https://cran.r-project.org/web/packages/dismo/vignettes/brt.pdf>
- Elith, J., Leathwick, J. and Hastie, T. (2008). A working guide to boosted regression trees. *Journal of Animal Ecology*, vol. 77, no. 4, pp. 802–813.

- Food and Agricultural Organization of the United Nations statistical database (2014 (accessed 8 December 2016)). <http://www.fao.org/faostat/en/>.
- Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression*. 2nd edn. Sage, Thousand Oaks CA.
Available at: <http://socserv.socsci.mcmaster.ca/jfox/Books/Companion>
- Friedman, J., Hastie, T. and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22.
Available at: <http://www.jstatsoft.org/v33/i01/>
- Gary, C., Baille, A., Navarrete, M. and Espanet, R. (1997). TOMPOUSSE, un modèle simplifié de prévision du rendement et du calibre de la tomate. In: Baille, A. (ed.), *Actes du Séminaire de l'AIP intersectorielle "Serres"*, pp. 100–109. INRA, Avignon.
- Hastie, T. and Qian, J. (2014). Glmnet vignette.
Available at: https://cran.r-project.org/web/packages/glmnet/vignettes/glmnet_beta.html
- Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The elements of statistical learning*. Springer Series in Statistics, 2nd edn. Springer.
- Heuvelink, E. (2005a). Developmental process. In: Heuvelink, E. (ed.), *Tomatoes, Crop Production Science in Horticulture Series*, pp. 53–83. CABI Publishing, Wallingford, U.K.
- Heuvelink, E. (ed.) (2005b). *Tomatoes*, vol. 13 of *Crop production science in horticulture*. CABI Publishing, Cambridge, USA.
- Hijmans, R.J., Phillips, S., Leathwick, J. and Elith, J. (2013). *dismo: Species distribution modeling*. R package version 0.8-5.
Available at: <http://CRAN.R-project.org/package=dismo>
- Irmak, A., Jones, J.W., Batchelor, W.D., Irmak, S., Boote, K.J. and Paz, J.O. (2006). Artificial neural network model as a data analysis tool in precision farming. *Transactions of the ASABE*, vol. 49, no. 6, pp. 2027–2037.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer Texts in Statistics. Springer.

- Jones, J., Hoogenboom, G., Porter, C., Boote, K., Batchelor, W., Hunt, L., Wilkens, P., Singh, U., Gijssman, A. and Ritchie, J. (2003). The DSSAT cropping system model. *European Journal of Agronomy*, vol. 18, pp. 235–265.
- Jones, P. and Thornton, P. (2000). MarkSim software to generate daily weather data for Latin America and Africa. *American Society of Agronomy*, pp. 445–453.
- Kotsiri, S., Zering, K.D. and Mayer, M. (2014). Stochastic frontier yield function analysis to predict returns to a new crop: An example of *Camelina sativa* yields conditional on local factor levels. In: *2014 AAEE Annual Meeting*. Minneapolis, Minnesota.
- Kuhn, M. (2015a). *caret: Classification and Regression Training*. R package version 6.0-52. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca and the R Core Team. Available at: <http://CRAN.R-project.org/package=caret>
- Kuhn, M. (2015b). *A Short Introduction to the caret Package*. Available at: <https://cran.r-project.org/web/packages/caret/vignettes/caret.pdf>
- Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, vol. 2, no. 3, pp. 18–22. Available at: <http://CRAN.R-project.org/doc/Rnews/>
- Lumley, T. (2009). *leaps: regression subset selection*. R package version 2.9. Code adapted from Fortran code by Alan Miller. Available at: <http://CRAN.R-project.org/package=leaps>
- Mccown, R., Hammer, G., Hargreaves, J., Holzworth, D. and Huth, N. (1995). Apsim—an agricultural production system simulation-model for operational-research. *Mathematics and Computers in Simulation*, vol. 39, pp. 225–231.
- Met Office (2011). *National Meteorological Library and Archive: Fact sheet No. 3—Water in the atmosphere*. Met Office.
- Met Office, 2010 (2010). *National Meteorological Library and Archive: Fact sheet No. 6—The Beaufort Scale*. Met Office.
- Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. and Leisch, F. (2014). *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*. R package

- version 1.6-4.
Available at: <http://CRAN.R-project.org/package=e1071>
- Milborrow, S. (2016a). *Plotting rpart trees with the rpart.plot package*.
Available at: <http://www.milbo.org/rpart-plot/prp.pdf>
- Milborrow, S. (2016b). *rpart.plot: Plot “rpart” Models: An Enhanced Version of “plot.rpart”*. R package version 2.0.1.
Available at: <http://CRAN.R-project.org/package=rpart.plot>
- Probert, M.E., Keating, B.A., Thompson, J.P. and Parton, W.J. (1995). Modelling water, nitrogen, and crop yield for a long-term fallow management experiment. *Australian Journal of Experimental Agriculture*, vol. 35, pp. 941–950.
- R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
Available at: <http://www.R-project.org/>
- Raes, D., Steduto, P., Hsiao, T. and Fereres, E. (2009). AquaCrop—the FAO crop model to simulate yield response to water: II. main algorithms and software description. *Agronomy Journal*, vol. 101, pp. 438–447.
- Rencher, A.C. and Schaalje, G.B. (2008). *Linear models in statistics*. 2nd edn. John Wiley & Sons, Inc., Hoboken, New Jersey, USA.
- Ridgeway, G. (2007 August). *Generalized boosted models: A guide to the gbm package*.
- Ross, S.M. (2013). *Simulation*. 5th edn. Academic Press.
- Royal Meteorological Society (2016). Beaufort scale. <https://www.rmets.org/weather-and-climate/observing/beaufort-scale>. Accessed: 2016-07-04.
- Schneider, A. (2016). GPS visualizer: Do-it-yourself mapping. <http://www.gpsvisualizer.com/>. Accessed: 2016-07-04.
- Scholberg, J., Boote, K., Jones, J. and McNeal, B. (1997). Adaptation of the CROP-GRO model to simulate the growth of field-grown tomato. In: Kropff, M., Teng, P., Aggarwal, P., Bouma, J., Bouman, B., Jones, J. and van Laar, H. (eds.), *Systems approaches for sustainable agricultural development: Applications of systems approaches at the field level*, pp. 133–151. Kluwer, Dordrecht, Netherlands.

- Stanger, T.F. and Lauer, J.G. (2008). Corn grain yield response to crop rotation and nitrogen over 35 years. *Agronomy Journal*, vol. 100, no. 3, pp. 643–650.
- Steduto, P., Hsiao, T., Raes, D. and Fereres, E. (2009). AquaCrop—the FAO crop model to simulate yield response to water: I. concepts and underlying principles. *Agronomy Journal*, vol. 101, pp. 426–437.
- Stöckle, C., Donatelli, M. and Nelson, R. (2003). CropSyst, a cropping systems simulation model. *European Journal of Agronomy*, vol. 18, pp. 289–307.
- Stöckle, C., Nelson, R., Donatelli, M. and Castellví, F. (2001 July). ClimGen: a flexible weather generation program. In: *2nd International Symposium Modelling Cropping Systems*, pp. 229–230. Florence, Italy.
- Taiz, L. and Zeiger, E. (2002). *Plant physiology*. 3rd edn. Sinauer Associates, Inc.
- The International Plant Names Index (2012). Available at: <http://www.ipni.org>. Accessed: 2016-08-06.
- Therneau, T., Atkinson, B. and Ripley, B. (2014). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-5.
Available at: <http://CRAN.R-project.org/package=rpart>
- Therneau, T.M. and Atkinson, E.J. (2015). *An Introduction to recursive partitioning using the RPART routines*. Mayo Foundation.
Available at: <https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf>
- Thunderforest (2016). OpenCycleMap.org — the OpenStreetMap Cycle Map. <http://www.opencyclemap.org/>. Accessed: 2016-07-04.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288.
- Tyson, P.D. and Preston-Whyte, R.A. (2000). *The weather and climate of Southern Africa*. 2nd edn. Oxford University Press, Oxford, United Kingdom.
- Van Diepen, C., Wolf, J. and Van Keulen, H. (1989). WOFOST: a simulation model of crop production. *Soil Use and Management*, vol. 5, pp. 16–24.
- Van Keulen, H. and Wolf, J. (1986). *Modelling of agricultural production: weather, soils and crops*. Pudoc, Wageningen, Netherlands.