Universiteit
Leiden
The Netherlands

Reproducing Reality

A Comparison of Methodologies and Machine Learning Algorithms for the Calibration of Agent-Based Models in Ecology

Femke Keij

Thesis advisor: dr. George van Voorn, Biometris, Wageningen University & Research

Defended on March 13th, 2024

MASTER THESIS

STATISTICS AND DATA SCIENCE
UNIVERSITEIT LEIDEN

Foreword

The work in this thesis is mine, performed without help by outside sources except for the advice of my supervisor Dr. George van Voorn.

The Fire and Wolf Sheep Predation Agent Based Models originate from the freely available NetLogo models library. Any generated data and code are also freely available from my GitHub repository.

Special thanks to my supervisor Dr. George van Voorn, and my second reader Dr. Julian Karch.

Contents

Abstract

Although Agent-Based Models (ABMs) have recently gained popularity in the ecological modelling community, their methodology has not yet been standardised. Calibration is an important part of ABM development, since it ensures that the ABM accurately represents the real-world system it was modelled after. In order to contribute to the standardisation of ABM calibration methodology, this study investigated the effects of summarising ABM output, noisy data, and using time series on calibration. We also compared 4 different machine learning algorithms for calibration. We trained a linear regression (LR), partial least squares (PLS), random forest (RF), and multivariate random forest (MRF) algorithm on the NetLogo model library Fire and Wolf Sheep Predation ABMs using various different training data sets. RF and MRF showed improvements when using more time points or full time series, but LR and PLS did not. Although summarising repeated runs by their mean is very common, here we showed that it can be detrimental to ABM calibration. Noise in the real-world data (here we used noise in the test data as a proxy) was also detrimental to ABM calibration, although that is currently not taken into account in practice. No large differences between uni-variate (LR, RF) and multivariate (PLS, MRF) algorithms were observed, but multivariate algorithms were much slower. Non-linear algorithms (RF, MRF) performed better than linear algorithms (LR, PLS) in most cases. Although differences were small, MRF performs best and LR performs worst. This study showed that the common practices of summarising repeated runs and summarising time series are potentially problematic. Since overall calibration performance was poor, but small differences between algorithms were observed, further investigating other optimisation algorithms may be worthwhile. The methodology used by this study may be used to further investigate issues in calibration methodology, or when introducing new calibration algorithms.

Acronyms

ABM Agent Based Model.

LR Linear Regression.

MCC Matthews Correlation Coefficient.

MRF Multivariate Random Forest.

NRMSE Normalized Root Mean Squared Error.

PLS Partial Least Squares.

PPP Point Prediction Performance.

RF Random Forest.

RMSE Root Mean Squared Error.

Glossary

calibration The process of selecting Agent Based Model (ABM) parameters by comparing ABM outputs to real-world data.

calibration performance Given a training data set with ABM outputs generated by known input parameters, the ability to retrieve the *true* parameters, i.e. the parameters used to generate the output of a test data set.

equifinality The scenario where different parameter sets lead to the same or very similar output.

output The data resulting from running the ABM with given parameters. It usually takes the form of information about how the entities in the ABM behave through time and space. For example, a time series of body mass fluctuations or a map showing migratory behaviour.

parameter The quantitative settings governing the behaviour of the ABM. For example, reproductive rates, energy expenditures, and time spent foraging in one spot before moving on.

parameter space The full grid of all possible combinations of parameter settings.

reference-table algorithm An algorithm that is trained on a fixed set of training data. It creates a fixed model that relates predictors to responses (input to output). The same model can be used on different test cases.

search-based algorithm An algorithm that searches the parameter space for the optimal solution. It creates no fixed model but finds a unique solution to each problem. Each test case requires running a new model.

unidentifiable parameter A parameter whose effect on the ABM output is identical to that of another parameter, such that it is impossible to retrieve the correct parameter value from the output.

Chapter 1: Introduction

Ecological models can be used as *in silico* laboratories to answer questions that cannot be answered experimentally due to ethical objections or the large scope of the research question. They can therefore be used to study how ecological systems are affected by changes in management, environmental conditions, or infrastructure [1,2]. Examples of studies are investigating the effect of pollution on a river, or the effect of climate change on migration routes. Traditional ecological models, based on differential equations, represent a population as a single entity by using aggregated state variables [1, 3]. More recently, Agent Based Models (ABMs) have gained popularity in the ecological modelling community. ABMs describe rules that govern the behaviour of individual organisms, as well as interactions with their environment and other individuals. From these behavioural rules and interactions, population dynamics emerge. ABMs are thought to more accurately represent the fact that real-world dynamics emerge from individual decision-making entities. This representation of underlying processes yields greater predictive power in novel conditions, and makes ABMs better suited to answering research questions regarding, for example, climate change [1, 3, 4]. Aside from the behavioural rules that make up the model, ABMs have two other important components. First, before running the ABM, one needs to set the parameters. Where the behavioural rules determine how the ABM behaves qualitatively, the parameters govern the quantitative aspects of that behaviour (e.g., the behavioural rule tells us that an animal should move, the movement parameter determines how far). Examples of parameters are reproductive rates, energy expenditures, and time spent foraging in one spot before moving on. Second, when running the ABM with set values for those parameters, it generates output. This usually takes the form of information about how the entities in the ABM behave through time and space. This can be, for example, a time series of body mass fluctuations or a map showing migration routes.

The growing use of ABMs has also resulted in increasing focus on methodological issues. Specifically, for ABMs to continue to advance, the associated methodologies need to be standardised [5, 6]. One part of this methodology is ensuring that the ABM is capable of (re)producing real-world dynamics. This ensures that the model accurately represents the underlying processes in the system and therefore has a greater chance of correctly predicting the systems' reaction to novel conditions. To this end the ABM parameters are estimated in a process called calibration [1, 7, 8]. Calibration, also known as inverse parameter estimation, is the process of selecting model parameters by comparing model output to real-world data [8, 9]. Sometimes, parameters can be measured directly, obtained from literature, or inferred by specialists [8]. Often, not all parameters can be obtained in this manner, or only range estimates can be made. Such parameters need to be determined by the developer of the ABM. The less data is available, the more parameters have to be calibrated. Additionally, while the parameters often specify values at the individual level (e.g., physiological parameters), real-world ecological data is often available at a much larger scale (e.g., consumption of a full population). We are, therefore, attempting to estimate the underlying parameters from the effect they have on larger-scale processes [10].

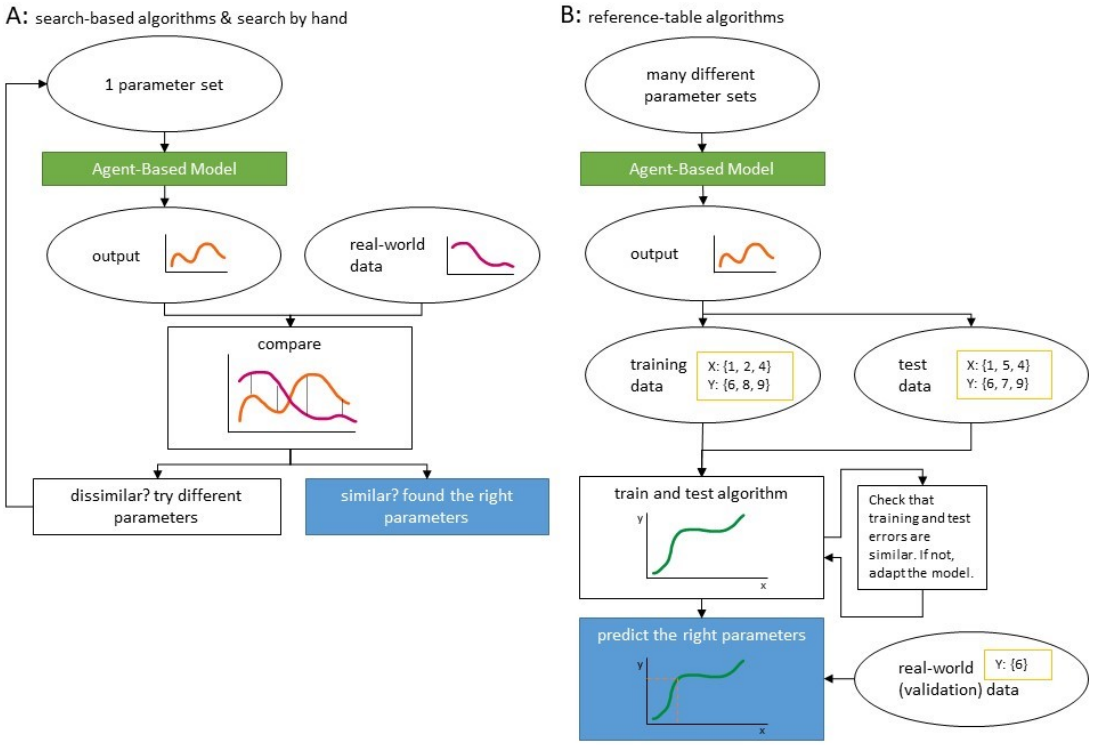


Figure 1.1: The calibration process. A schematic overview of the calibration process for Agent-Based Models with 1 parameter (x) and 1 output (y). A) shows the iterative process applicable to search by hand as well as for search-based algorithms (here simplified, the specifics of the search-based algorithms are not shown but occur within the comparison step). Steps are repeated until ABM output and real-world data are judged to be similar enough. B) shows the non-iterative process applicable to reference-table algorithms, where training and test data are used to train the algorithm and real-world (validation) data is then given to the trained algorithm to obtain the parameters.

The traditional approach to calibrating ecological models involves maximising the likelihood. Since ABMs are not deterministic models, the likelihood is not available and these traditional methods are not suitable for their calibration. Other methods that approximate the likelihood (for example quasi- or pseudo-likelihoods) are often not feasible for ABMs because they are too computationally expensive [3, 9]. Instead, the calibration can be performed as an optimisation problem. This is done by varying the parameters and observing the ABM output. When the output resembles the real-world system that is being modelled, one has found appropriate parameters. Such a search can be done by hand, but this is inefficient for models with large amounts of parameters [1, 11]. Instead, an optimisation algorithm may be used to automate the process by finding the parameters that produce the minimum distance between ABM output and real-world data [10, 12]. This has the additional benefit that an automated process can explore more different parameter combinations than a developer could by hand [13]. This process is outlined in Figure 1.1. Such minimisation requires sampling the ABM parameter space (if using

an algorithm that requires training data), then summarising the ABM output produced by those parameters, choosing a distance function that represents the distance between the ABM output and the real-world data, and then minimising that distance function. All these steps can be done in multiple ways, and it is often unclear how those choices influence calibration performance, i.e. the ability to accurately retrieve the desired parameters, namely those that accurately represent their real-world counterparts. Additionally, a number of issues arise during the process, most of which are also understudied [1]. Both these choices and arising issues will be discussed below.

Sampling the ABM parameter space ABMs can take a very long time to run. Therefore, it is often not possible to sample the entire parameter space and obtain a full overview of potential model outputs to train reference-table algorithms (i.e., an algorithm that is trained on a fixed set of training data) and perform model inspections [4, 7, 8, 14]. Recently, the advance of high performance computing and parallel computing enables running larger portions of ABM parameter spaces, but these approaches may be costly and are not accessible to everyone [6]. Instead, the parameter space is sampled to obtain a representative sample of parameter combinations and their outputs. This sample can then be used to train calibration algorithms that will minimise the distance function [8]. Although the sampling step is rarely mentioned in calibration literature, it is important to the calibration process. The *garbage in, garbage out* principle tells us that creating a poor training data set that insufficiently covers the parameter space will not aid in obtaining a well-calibrated model.

There are multiple options for sampling from the parameter space. Random sampling is a simple and often used approach, but may not adequately cover the parameter space. More structured sampling approaches such as factorial or Latin Hypercube sampling may improve on the random sampling strategy by ensuring the sample covers all regions of the parameter space. A more advanced approach would be to train a meta-model (for example a support vector machine) to recognise the different outcome regions of the ABM, and then sample within those regions using either random sampling or a structured sampling approach [15].

Aside from the sampling strategy, one must choose a sample size. In practice, sample sizes are often taken as a convenient number that is not too computationally expensive [9]. It is currently unclear how one should choose a sample size for ABM calibration, and how much influence this choice has on the calibration process.

Since ABMs are often stochastic models, each sampled parameter combination needs to be run multiple times to obtain a robust and reliable estimate (in practice usually the mean of the different runs) of the model output for that parameter combination [1, 7, 8, 16]. Literature does exist on how to choose the appropriate number of runs for each parameter combination, but this is often ignored in practice. Instead, most studies use a conveniently small number or do not repeat parameter combinations at all [1].

Summarising the model output ABMs can produce a lot of data since there are usually multiple output variables that change throughout the run-time. The amount of generated data makes optimisation both slower and more complex. In practice, ABM output is therefore usually summarised into a more concise data set.

Summarising ABM outputs occurs on multiple levels. First, ABMs are commonly spatial models. Agents may move across a grid where different locations can have different environmental characteristics, and where they may encounter other individuals [6]. Output variables of ABMs are often temporal only. By ignoring the spatial component of the ABM, we are also ignoring potential spatial auto-correlation. Whether this is desirable is questionable. Especially for ABMs that will be used to study their spatial components (e.g., migration corridors), spatial calibration may be crucial to the predictive power of the ABM. Second, ABM outputs usually take the

form of time series data, but calibration often does not utilise the full time series but rather a summarised version. Researchers may take the data from a number of time points, in addition to the mean value, standard deviation, and trend [8, 9]. This approach saves computing time and allows for the use of calibration methods that are likely more familiar to researchers, such as simple linear regressions, over more complicated temporal models. However, this ignores both temporal auto-correlation and potentially obscures important behaviours such as tipping points. Calibrating with a full time series will potentially yield much better results [1], although using fewer time points may partially eliminate the temporal auto-correlation and thereby allow for the use of non-temporal optimisation algorithms. Third, to account for stochasticity, the multiple outputs resulting from running the same parameter set multiple times are usually summarised by taking their mean. However, output is not always normally distributed around a single mean, meaning it is not immediately clear that summarising outputs by their mean is meaningful.

When choosing to spatially and temporally summarise the data, there are many different ways to do this. One could use all possible summary statistics of the outputs when calibrating the ABM, but this may be inefficient and does not necessarily lead to better results than using fewer summary statistics, if the added summary statistics are noisy or provide only duplicate information [9]. Instead, summary statistics can be chosen based on subject expertise, or by using an algorithm to weigh their information.

Obtaining real-world data ABMs are developed to represent a certain real-world system or process. To do this, we want to see if the ABM can reproduce real-world data or patterns similar to it. This process makes two assumptions: 1) the ABM is correctly implemented and contains parameters that have a real-world equivalent and can therefore be estimated from real-world data, and 2) there is real-world data available that is accurate and informative enough to be able to infer those parameters from it. Data is often unavailable, but even if it is available it will often be on a different scale than the ABM output. In such cases, one or the other has to be summarised so that the two become comparable [8]. There is no guarantee, however, that the data contains enough information to facilitate the parameter estimation in the first place, much less after summarising it and inevitably loosing some information. The data will also always contain noise [7]. Some types of noise, such as random measurement errors, are not relevant to the ABM. If, however, the noise introduced in the data is created by a systematic process that is not included in the ABM, we may wrongfully conclude that the ABM is wrong because it cannot reproduce the real-world data.

Choosing a distance function ABM outputs are usually dynamic in time and space and may consist of multiple different variables. There are, therefore, multiple ways to compute the distance between ABM outputs and the corresponding real-world data, even after summarising one or both. Additionally, for all but the most basic ABMs, there is likely more than 1 parameter to calibrate. A common approach to fitting more than one parameter is to use a separate calibration algorithm for each parameter and then collect the best-fit parameters from each to form the 'optimal' parameter set. This approach fails to take into account the interaction between parameters and therefore does not guarantee the best fit [10]. Instead, one may use a multivariate algorithm that can optimise multiple parameters at the same time or use Pareto optimisation fronts.

Minimising the distance function Once training data has been obtained and summarised, real-world data has been collected, and a distance function has been chosen, we can train a calibration algorithm to minimise the distance between ABM output and real-world data and obtain parameter estimates. This is an optimisation problem and can therefore be tackled

using optimisation algorithms. We can distinguish two types of optimisation algorithms used for calibration. Reference-table algorithms are those that are trained on a fixed set of training data (the reference-table). They create a fixed model that relates input to output and can be used to evaluate multiple test and/or validation cases. Search-based algorithms, on the other hand, search the parameter space for the optimal solution. This does not create a fixed model but a unique solution to each problem, and they therefore need to be re-trained for every test and/or validation case. Reference-table algorithms are more efficient if the algorithm needs to be re-used multiple times for different applications on the same problem (for example, optimisation and validation). Search-based algorithms are generally faster when using them for a single application, because they do not explore the parameter space around bad solutions as extensively as reference-table algorithms do [9]. Regardless of type, there are many optimisation algorithms available and not all of them are accessible to researchers without extensive statistical training. To complicate matters, there is no clear consensus on which optimisation algorithms are best to use for ABM calibration [8].

A number of issues from the general optimisation problem emerges when it comes to ABM calibration. For reference-table algorithms, we run the risk of over-fitting the algorithm by fitting it not just to the underlying trend but also to the individual data points. This can lead to an ABM that only perfectly reproduces the training data it was calibrated to, rather than accurately represents the underlying processes. This becomes problematic when the model is subsequently used to make predictions about novel situations [1]. The fact that the search space created by all possible parameter sets likely has local minima / maxima in addition to the global minimum / maximum is an issue for search-based algorithms. They introduce the probability that the parameter estimates do not constitute the best possible parameter set. The local minima / maxima can also slow down search-based algorithms, as they can get stuck in the flat areas of the search space [11, 17].

Other optimisation issues are more unique to ABMs. To start, ABMs can have very many parameters and may therefore need a lot of data (for reference-table algorithms) or a long run-time (for search-based algorithms) to obtain accurate parameter estimates. This may be especially problematic in combination with the severe run-time limitations mentioned earlier [7, 14]. To complicate matters, the parameters are often interdependent, such that they have to all be calibrated either simultaneously or in large sets [8, 10]. Additionally, not all parameters may be identifiable. Unidentifiable parameters are those whose effect on the output is identical to that of another parameter, such that it is impossible to retrieve their value from the output alone. Parameters may also be unidentifiable because they do not influence the output at all. This specific type of unidentifiable parameters becomes apparent during sensitivity analysis [7]. Unidentifiable parameters can essentially not be calibrated and must therefore be identified before calibration starts [9]. This can be done using, for example, perturbation experiments, but ideally one would find values for these parameters from literature or experts. Awareness of unidentifiable parameters is important because most calibration algorithms will still yield parameter estimates for them, and researchers may take them at face-value if not aware of this issue. In addition to identifiability, equifinality describes the scenario where different parameter sets lead to the same or very similar model output. This also creates a scenario where it is impossible to retrace the parameters from the ABM output alone [7, 10]. Here too, it is important to study the ABM before calibration so that any equifinality issues are identified before the start of the calibration procedure. Another point to be aware of is the fact that the relationship between ABM parameters and output is often non-linear and may change through time [3, 11]. This has not stopped researchers from using, for example, linear regression to calibrate ABMs. This is not necessarily an issue as long as the calibration method performs well and the goal of the ABM is predictive rather than explanatory, but it could be argued that the underlying structure of ABMs (i.e.,

they use individual-scale processes to predict larger-scale phenomena) makes them explanatory by nature.

Lastly, new methodologies are often introduced together with certain sampling, summarising, and distance function choices. This convolutes the evaluation of the performance of the new calibration algorithm [9].

The ideal calibration algorithm Keeping the issues mentioned above in mind, one can list a number of criteria a calibration algorithm would ideally adhere to. To start, the algorithm should be able to handle non-linear relationships between ABM input parameters and output values, and be able to either optimise multiple parameters at once or a weighted summary of them. Any algorithm that needs fewer data from the ABM to perform well has an advantage due to the extensive computational time required to run an ABM. For the same reason, algorithms that are fast to compute are preferable. Reference-table algorithms have an advantage since the training data created for them can be re-used in other steps of the ABM development process, such as for the sensitivity analysis [9]. These algorithms should, however, not over-fit to noisy data. Note here that training data for ABMs is not generally noisy data since it is generated by the ABM itself and therefore clean. It can however become noisy when multiple runs of the same parameter combination are summarised in a way that inaccurately represents the underlying distribution of the outputs. Search-based algorithms should be able to deal with local minima, by using, for example, a stochastic approach or multiple starts. Since most ABM developers do not have extensive statistical training, an algorithm should be easy to use and interpret correctly. And last but not least, the algorithm should of course be able to consistently and accurately retrieve the best parameters.

Previous studies There is, in general, little literature available on the topic of ABM calibration. As mentioned above, many studies introducing new calibration algorithms mix sampling strategies, summarising, and the actual parameter estimation together such that the performance of the new calibration algorithm on its own is indiscernible. Additionally, many studies illustrate the use of the new method on a single or a few ABMs, without comparing the performance to that of other algorithms on those same ABMs. Specific comparison reviews are, in turn, often limited to a small number of optimisation algorithms or focus only a small number of ABMs, making it impossible to generalise their results. Carella [9] claims to address all these issues in their paper. They investigate a number of reference-table algorithms over a range of ABMs and equation-based models, using a common sampling and summarising strategy for all. They conclude that no single algorithm performs best, and 'we should abandon the hope that a large enough literature survey will uncover the single best estimation algorithm to use' (page 14, [9])). To support this claim, they use the *no free lunch* theorem, originally posted by Wolpert and Macready [18]. This theorem states that the performance of all algorithms is exactly the same if averaged over all possible search problems. We argue it is too soon to make such a claim for ABM calibration. Namely, we are not interested in the best algorithm over all possible search problems, only in the best algorithm for ABM calibration, a specific subset of all possible search problems. Additionally, Carella [9] used very small sample sizes (1250 and 5000 simulation runs), only explored reference-table algorithms, and ignored possibilities other than summarising time series into distinct time points. They do not report on investigating equifinality or unidentifiable parameters in the ABMs they used. They also do not mention how they determined how many times each parameter should be run, and it is generally unclear whether they repeated parameter combinations at all.

Lastly, current studies use ABM generated, and therefore noise-free, data for training and testing. This is convenient but fails to take into account that real-world data that would be used for calibration in practice would be noisy data. The extent to which this influences the performance of calibration algorithms is unclear.

Research questions For many of the issues listed above the theoretical best choice is known (e.g., we expect Latin Hypercube sampling to provide better coverage of the parameter space than random sampling), but the extent to which these choices influence calibration performance is unclear. Potentially, calibration performance does not depend on any of these choices at all, as Carella [9] suggests for the choice of optimisation algorithm. If these choices do matter however, it is interesting to investigate which has the largest influence on calibration performance. The overarching aim of this study is, therefore, to investigate which of the factors mentioned above have a considerable impact on calibration and should therefore be considered when constructing a standardised calibration approach for ABMs. To that end, this study will attempt to answer the following research questions:

1. What is the influence of ...
   (a) Using a full time series vs. using time points
   (b) Summarising vs. not summarising repeated runs of the same parameter combination
   (c) Noise in the real-world data (in this study the test data)

   on calibration performance (i.e., the ability to deduce the input parameters correctly)?
2. Which algorithm most successfully calibrates ABMs?
   (a) A uni-variate or multivariate algorithm?
   (b) A linear or non-linear algorithm?

Note that this study, due to time constraints, does not address all issues and (potential) solutions discussed here and only includes reference-table algorithms. All questions will be addressed in such a way that the effects of each choice can be reconstructed. The methodology used to answer them is discussed in Chapter 2. Chapter 3 discusses the results of the study, followed by a discussion and conclusion in Chapter 4. Comprehensive lists of acronyms and terms can be found on pages 6 and 7, respectively. All used ABMs and results specific to them are detailed in Appendices A and B.

Chapter 2: Methodology

All analyses were performed with R version 4.2.2 [19]. All ABMs were run in Netlogo version 6.2.1 [20], controlled via PyNetlogo [21] with Python version 3.8.5 [22] and Java version 8 [23]. All code was written in Rstudio version 2022.12.0 [24]. The complete code and corresponding documentation are available at this GitHub repository.

2.1 Study design

This study used two ABMs to compare different choices in calibration methodologies. After deciding on ABM outputs to use, the number of repeated runs per parameter combination was determined using mean convergence, the coefficient of variation, and the relative outer variance. A random sample of the parameter space was then taken and the ABMs were run for those parameter sets. The data was either kept as it was, or the repeated runs of each parameter set were summarised by taking the mean values at each time point. The ABMs were inspected by visualising their parameter space, identifying equifinality, identifying unidentifiable parameters, and plotting temporal auto-correlations. To mimic calibration in practice, noise was added to the data before using it for training and testing. A simple linear / logistic regression, random forest, partial least squares, and multivariate random forest were then trained, using 5-fold cross-validation, to retrieve input parameter values from the ABM output. The success of this endeavour was evaluated using a number of error metrics. A complete high-level overview of the study design is visible in Figure 2.1. Each step will be further detailed below.

Each ABM $M_1, M_2, ..., M_q$ has P_q input parameters $\theta_{q,1}, ..., \theta_{q,p}$ and R_q outputs $y_{q,1}, ..., y_{q,r}$. The machine learning algorithms use outputs $y_{q,1}, ..., y_{q,r}$ to produce parameter estimates $\hat{\theta_{q,1}}, ..., \hat{\theta_{q,p}}$. The training sample size for model q is indicated as n_q. The number of times each parameter combination is repeated is labelled m_q. For legibility purposes, the subscript q is only included when a computation involves multiple ABMs.

2.1.1 The Agent-Based Models

Two ready-to-use ABMs from the NetLogo Models Library were selected for this study. The Fire ABM is a relatively simple application where a fire spreads through a stand of trees. The spread of the fire is dependent on the density of the trees in the stand, as well as in how many directions the simulation allows the fire to spread [25]. The Wolf Sheep Predation ABM consists of wolves and sheep moving around randomly and reproducing with a set probability. Wolves eat sheep they encounter and sheep eat grass they encounter in order to replenish the energy they spend on moving. Grass regrows after a set amount of time [26]. Table 2.1 shows an overview of parameters, outcomes, and runs per parameter combination for both ABMs. Further details on both ABMs can be found in Appendices A and B.

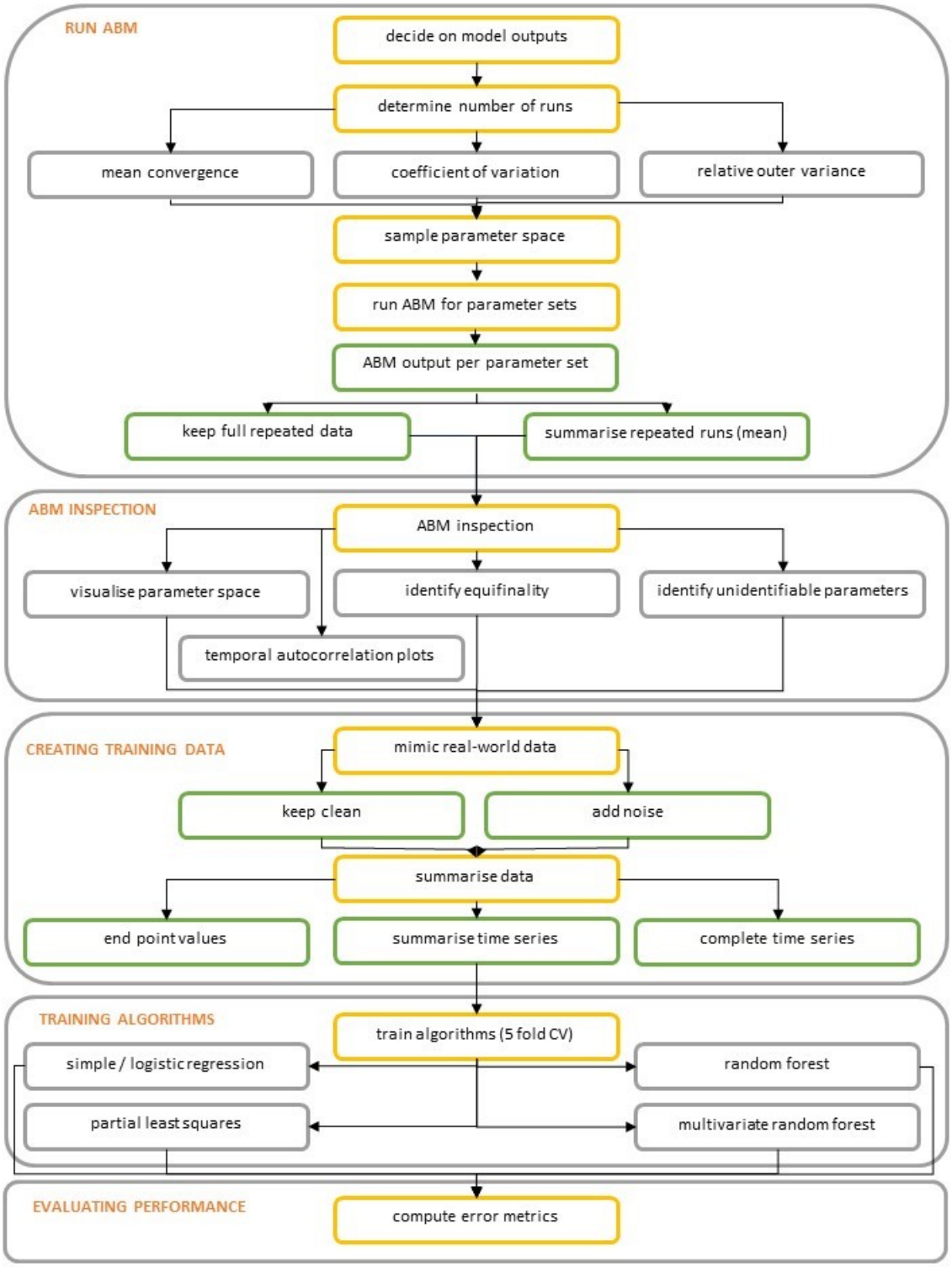


Figure 2.1: Flowchart of Study Methodology. All major steps (as explained in the text) are labelled in yellow boxes, with sub-steps in grey boxes. All steps that produce data are shown as green boxes.

Table 2.1: ABM overview: ABMs used in this study, as well as their parameters and outcomes.

ABM	reference	parameters (number of)	outcomes (number of)	runs per parameter combination	number of parameter combinations
fire	[25]	density, directions (2)	burn percentage (1)	5	200
Wolf Sheep Predation	[26]	initial number of sheep, initial number of wolves, energy gained from food by sheep, energy gained from food by wolves, probability of sheep reproduction, probability of wolf reproduction, grass regrowth time (7)	amount of sheep, grass, wolves (3)	15	$> 13 * 10^{12}$

2.1.2 Running the Agent-Based Model

Determining the number of model runs To ensure precision of and confidence in the output, each parameter combination of a stochastic ABM has to be run multiple times [8, 27]. In practice, the number of runs per parameter set is often chosen to maximise the number of parameter combinations within a limited amount of run time, resulting in parameter combinations being repeated very few times or not at all [16].

There are three ways to determine the appropriate number of runs per parameter combination. In linear stochastic models, the appropriate number of runs m is that at which the coefficient of variation stabilises (within a margin ϵ_{c_v}). The coefficient of variation is computed as the standard deviation over the mean of the outcome variable of interest r as $c_{v,r} = \frac{\sigma_r}{\mu_r}$. This approach only guarantees convergence in linear models, but is commonly used in ABMs even though they are usually not linear [8,27]. Alternatively, one can choose the number of runs as the point where the mean outcome value over all runs converges (i.e., no longer changes up to some threshold value ϵ_μ). This is potentially much more computationally demanding than computing the coefficient of variation, but also less sensitive to non-linearity [28]. Both of these methods encounter (opposite) issues when applied to studying multi-output ABMs, as almost all ABMs are. Outputs with μ_r closer to 0 are likely to have a larger c_v. If we apply the same ϵ_{c_v} to all outputs, we consequently get higher m estimates for outputs with μ_r close to 0 [16]. In contrast, outputs with μ_r closer to 0 are likely to converge more quickly within ϵ_μ than outputs with larger μ_r. Consequently, we get higher m estimates for outputs with μ_r further away from 0. Therefore, both methods under- or overestimate m depending on the value of μ_r. One solution would be to apply a separate ϵ_{c_v} and ϵ_μ for each different output. Alternatively, we can use a third method, the relative outer variance. The relative outer variance aims to combat the aforementioned issue by looking at the stability of the variance, without taking the mean value μ_r into account. The

idea is that at low run numbers the variance in the sample varies greatly from one run number to the next. Therefore, when determining how large the number of runs should be, we should take into account not only the inner variance (i.e., the variance within the sample), but also the outer variance (i.e., the variance between run numbers). Here, we use a window size of 10 to compute the outer variance. We observe the windowed variance ω_r^2 for the ABM output over an increasing number of runs. We set m to the number of runs where the relative outer variance $\omega_r^2/\omega_{r,max}^2$) falls below a certain threshold ϵ_{ω^2} [16].

Here, we took a systematic approach to determining the number of runs per parameter combination. This ensures the training samples gave an accurate representation of the model outcomes. Each parameter was set to a small number of settings and then combined with others to obtain a grid of parameter combinations [27]. The coefficient of variation, mean convergence, and relative outer variance were computed for an increasing number of samples for every outcome of the ABM. We then plot a histogram that shows after how many runs these quantities meet their respective thresholds for each parameter combination. These histograms aid in informing us of the minimum number of runs m necessary to obtain a reliable estimate. Notably, the relative outer variance failed to converge in almost all cases, making the measure impossible to use in practice. Therefore, we only looked at the mean convergence and coefficient of variation. Due to computational constraints, these methods were applied only to summarised outputs, not full time series.

Full details on parameter settings and how the number of runs was determined for each ABM can be found in Appendices A and B. The resulting number of runs per parameter combination for each ABM can be found in Table 2.1.

Sampling parameter space We sampled 198 parameter combinations for the Fire ABM (the maximum number) and 200 parameter combinations for the Wolf Sheep Predation ABM (a number informed by computational limitations). The required number of parameter combinations was obtained through random sampling. Each parameter combination was run m times (see Table 2.1) to obtain the training data. Although it is possible to obtain potentially better samples by first training a surrogate model to recognise different outcome regions and then sampling within and especially around the borders of those outcome regions (as in [15]), this approach is beyond the scope of this study.

Summarise repeated runs There are now 2 ways to proceed. Traditionally, we can take the mean value of each output at each time point, per parameter combination. This summarises the repeated runs of each parameter set into 1 number per time point. However, since not all outputs follow a normal distribution concentrated around a single mean, this approach may not be appropriate for all outputs. Instead, we can choose not to summarise and keep multiple entries for each parameter combination. This increases the computational burden of further computations, but potentially leads to better calibration since the training data contains more realistic output values. In order to answer research question 1a (what is the influence of summarising vs. not summarising repeated runs of the same parameter combination on calibration performance), we applied both strategies and compared them in the evaluation step later on.

2.1.3 Agent-Based Model inspection

Visualising parameter space ABM inspection starts with a visualisation of the parameter space and corresponding outputs. The simplest way to do this is to plot the input parameters against the output, or to plot the progression of the output parameter through time. This will

give a general impression of the different behaviours the ABM may display and their connection to the input parameters. Some issues, like equifinality, may become immediately visible. The visualisation also serves to gain a better understanding of the ABM, such that any issues encountered later may be solved more easily.

Identifying equifinality For simple models with few parameters and outputs, equifinality may be easily identified by looking at parameter space visualisations. When plotting the progression of output parameters through time, similar trends produced by very different parameter combinations indicate equifinality. When there are many parameters and / or outputs, however, this is not always trivial. Therefore, we employ a clustering approach to identify similar outcomes and then inspect the parameters that generated outputs within the same cluster. If there are very different parameters within the same cluster, this indicates that very different parameters can produce similar outputs, and we have thereby identified equifinality. To do this, we first standardised the output data to have zero mean and standard deviation one. Then a k-means clustering algorithm was applied using the `factoextra` package [29], where we clustered based on the output data at a few different time points. To determine the appropriate number of clusters, we employ the elbow and silhouette methods [30]. The k-means clustering algorithm is very fast when dealing with large datasets, which is an important advantage in this application. k-means clustering is sensitive to outliers but we did not consider that to be an issue in this case, since we know that the simulation data contains only correct and accurate data, and any outliers are therefore informative. There are other clustering methods that can deal with full time series data, such as hierarchical clustering with differential time warping, but these were too computationally intensive to be applied in this study.

Identifying unidentifiable parameters There are two ways in which a parameter can be unidentifiable: 1) it does not influence the output at all, and 2) its' effect on the output is identical (or exactly opposite) to that of another parameter. To identify these issues, we plotted correlation plots of the input parameters with resulting outputs. When a parameter does not influence the output at all, we observe a flat relationship between input and output (i.e., the output does not change as a result of a change in the input parameter). When two parameters have similar influences, their slopes will be similar (i.e., the output changes as a result of a change in the input parameter, but this change is identical to that as a result of another parameter). One way to identify unidentifiable parameters is with perturbation experiments [31]. Here, we did not include these because of time constraints.

Temporal auto-correlation plots Lastly, we create auto-correlation and partial auto-correlation plots. This allows us to see whether we expect non-temporal algorithms to work well for a specific ABM (when there is little auto-correlation) or not (when there is strong auto-correlation).

2.1.4 Creating training data

Adding noise In practice, one would train and test a calibration algorithm using clean, ABM-generated data and then apply the trained algorithm to the noisy real-world data (see Figure 1.1). However, in order to answer research question 1 c (what is the influence of noise in the real-world data on calibration performance), we added noise to the test data to see how that would influence calibration performance. Noise can be introduced randomly in two ways. First, for both the Fire and the Wolf Sheep Predation ABMs noise was introduced around the output data, with a maximum size of 20% of the mean value of that output data. The noise level was sampled from a normal distribution such that extreme noise is rare but most observations will contain at least

some noise. If noise introduction yields negative values where they are not possible, these values were corrected to 0. Second, noise was introduced to the Wolf Sheep Predation ABM in the time dimension by randomly shifting the time series up to 10 steps forwards or backwards. Noise was not introduced to the time dimension of the Fire ABM because the length of the simulations of that ABM is dependent on the speed of the fire spreading, which is directly related to the density parameter that has to be estimated.

Summarising model output In order to answer research question 1 a (what is the influence of using a full time series vs. using time points on calibration performance), ABM output was summarised in different ways. For the Fire ABM:

- End point values (here the last value in the simulation), mean, minimum, maximum, standard deviation, and trend of the output;
- End- and midpoint values, mean, minimum, maximum, standard deviation, and trend of the output;
- Not summarised (the full time series was kept).

For the Wolf Sheep Predation ABM:

- End point (here time step 500) values only;
- Output values every 10 time steps, as well as mean, minimum, maximum, standard deviation, and trend of the output;
- Output values every 20 time steps, as well as mean, minimum, maximum, standard deviation, and trend of the output;
- Output values every 50 time steps, as well as mean, minimum, maximum, standard deviation, and trend of the output;
- Not summarised (the full time series was kept).

The data for the Fire ABM were less extensive because there is a maximum of 200 parameter combinations, and the variations within those combinations due to stochasticity are minimal. This means that it was impossible to fit more variables using linear regression, due to the multicollinearity in the data. Trends were computed using the `trend` package, which computes the slope of the data on multiple intervals and then returns the median of those slopes as the 'trend' [32].

2.1.5 Training the algorithms

A number of calibration algorithms was implemented and their performance compared. These algorithms were chosen because they are prevalent in the existing calibration literature (simple linear / logistic regression, Random Forest (RF)) or because we believed they have certain advantages that may improve on the existing calibration methodology (Partial Least Squares (PLS), Multivariate Random Forest (MRF)). All algorithms in this study were reference-table algorithms and thus use training data to predict the test data. They were trained to estimate the relationship $\theta = f(y)$ between input parameters and outputs. The trained function $f(y)$ was then used to produce an estimate of the input parameters θ^*. This process was repeated for each input parameter, or in the case of the multivariate algorithms (PLS and MRF) applied for all input parameters at once. Each algorithm was trained and tested using 5-fold cross-validation with folds created by the `caret` package [33].

2.1.6 Evaluating performance

After training the algorithms, we assessed how well they performed in predicting the parameters in order to answer research question 2 (which algorithm most successfully calibrates ABMs). The error computations were performed on each of the 5 test folds separately, and then averaged. To facilitate comparisons among ABMs and algorithms, a fixed set of error metrics was computed. There are many error metrics available, but here we chose to use relatively simple ones. This is because we want to be able to compare error metrics across all algorithms, and some error metrics require additional information on for example class membership probabilities or prediction intervals, which not all algorithms produce. We were interested in the ability of the algorithms to correctly retrieve the input parameters corresponding to the outputs in the test data and therefore computed error metrics comparing the predicted to the true parameters. Although points close to each other in parameter space may produce very different outputs (e.g., around tipping points) the computational burden of re-running the ABMs with the predicted parameters prevented the computation of error metrics comparing the true output and the output generated by the predicted parameters.

Both the Fire ABM and the Wolf Sheep Predation ABM contain continuous parameters. For these parameters we computed four different error metrics:

1. *Root Mean Squared Error (RMSE)* represents the standard deviation of the prediction errors. Smaller values indicate smaller differences between the predicted and true parameters. It is computed as
$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{\theta}_i - \theta_i)^2}{n}} \tag{2.1}$$
with n the sample size of the test data.

2. *Normalized Root Mean Squared Error (NRMSE)* allows for comparisons between different ABMs. It is computed as
$$\text{NRMSE} = \frac{\text{RMSE}}{\sigma_{\text{obs}}} \tag{2.2}$$
with σ_{obs} the standard deviation of the test data.

3. *Point prediction performance* is a measure that indicates how much better the algorithm performs than just taking the average value of the parameter in the training data. A score of 1 indicates perfect estimation, whereas 0 indicates unidentified parameters, and a negative value indicates mis-identified parameters. It is computed as
$$\text{point prediction performance} = 1 - \frac{\sum_{i=1}^{n}\sqrt{(\hat{\theta}_i - \theta_i)^2}}{\sum_{i=1}^{n}\sqrt{(\bar{\theta}_{\text{train}} - \theta_i)^2}} \tag{2.3}$$
where $\bar{\theta}_{\text{train}}$ is the average value of the parameter in the training data [9].

4. *Percentage estimated correctly* is computed as the percentage of parameters in the test set that were estimated correctly as
$$\frac{\sum_{i=1}^{n} I(\theta_i = \hat{\theta}_i)}{n}. \tag{2.4}$$
with $I = 1$ if $\theta_i = \hat{\theta}_i$ and $I = 0$ if $\theta_i \neq \hat{\theta}_i$.

The Fire ABM also contains a binary *density* parameter (see Table 2.1). For this parameter, we computed four error metrics:

1. *Cohen's kappa* (also called the kappa score), measures the proportion of correctly estimated parameters, while taking into account that random guessing would also produce the correct value in 50 % of cases. It is computed as

$$\kappa = \frac{2 * (TP * TN - FN * FP)}{(TP + FP) * (FP + TN) + (TP + FN) * (FN + TN)} \tag{2.5}$$

   with TP true positives, FP false positives, TN true negatives, and FN false negatives from a confusion matrix.

2. *F1 score* measures accuracy by combining precision (the proportion of true identifications that was correct) and recall (the proportion of trues identified as such) using their harmonic mean. Based on a confusion table, it is computed as

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}. \tag{2.6}$$

   Scores closer to 1 are better.

3. *Matthews Correlation Coefficient (MCC)* represents the correlation between the predicted and the true values. Based on a confusion table, it is computed as

$$\text{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{2.7}$$

   Scores vary symmetrically from - 1 to 1, with a score of 0 indicating that the algorithm performs no better than random guessing, which is useful information since even random guessing will correctly guess approximately 50% of the cases for a binary parameter. We computed MCC using the `mltools` package [34].

4. *Percentage estimated correctly* was computed as for the continuous case.

Confusion matrices and associated error metrics were computed using the `caret` package [33]. The error metrics for this parameter could not be directly compared to those for the continuous parameters, and were therefore only used to evaluate the calibration performance within the Fire ABM and not to be compared to the Wolf Sheep Predation ABM.

2.2 Calibration algorithms

All algorithms used in this study are reference-table algorithms, and were fitted using the same training data.

Simple linear / logistic regression A common approach to ABM calibration is to train a simple Linear Regression with the ABM outputs as predictors and the input parameter as the response variable [12]. A separate regression is trained for each input parameter. In the case of binary input parameters, a logistic regression is used.

The linear regressions take the following shape

$$\theta_{q,p} = \beta_0 + \beta_1 * y_{q,1} + ... + \beta_r * y_{q,r} + \epsilon \tag{2.8}$$

and for a logistic regression

$$\theta_{q,p} = \frac{\exp \beta_0 + \beta_1 * y_{q,1} + ... + \beta_r * y_{q,r}}{1 + \exp \beta_0 + \beta_1 * y_{q,1} + ... + \beta_r * y_{q,r}} + \epsilon. \tag{2.9}$$

The parameter estimates $\hat{\theta}$ are produced by inputting the output values from the test set in the obtained formula, i.e.

$$\beta_0 + \beta_1 * y_{q,1} + ... + \beta_r * y_{q,r} = \hat{\theta_{q,p}}. \tag{2.10}$$

We included a Linear Regression (LR) because it is the simplest and most well-known algorithm commonly used for ABM calibration. It is not expected to perform very well since it fits a linear relationship and the relationship between ABM input parameters and outputs is often non-linear. It will, however, serve as a good baseline to compare the other algorithms to. As used here, it fits all parameter separately, but one could adapt it to instead fit a weighted average. The main advantage of LR is that any ABM developer is likely familiar with the method and its interpretation.

Partial Least Squares When using a Linear Regression to calibrate an ABM, a separate model has to be made for each parameter. Partial Least Squares is a multivariate method and can therefore fit multiple outcome variables at the same time [35]. It is also better able to handle collinear datasets than Linear Regression. PLS is based on the sample principles as Principle Component Analysis. However, where Principle Component Analysis constructs the Principle Components based on which variables explain most of the variation within the variables themselves, PLS constructs components based on which predictors explain most of the covariance between the predictors and the response variables. We start by finding a component that explains most of the variation in the response variables, and then find a corresponding component in the predictors that best explains this variation. This ensures that the components are relevant to the regression that then follows. The fitted model takes the following shape:

$$\theta_{qp} = \sum_{i=0}^{k} \beta_i y_{q,i} + \epsilon_{qp}. \tag{2.11}$$

In order to fit a PLS for binary variables, they have to be turned into a dummy variable. The main advantage of PLS is that it fits all parameters at the same time, and therefore produces parameter sets that are a good fit, instead of individual parameters that may not work well together. Note that this is not something that we can study using the error metrics defined above. The construction of the components helps deal with collinear data, and PLS may therefore need less data to be fitted than Linear Regression does. The underlying relationship that is being fitted is still a linear one, which is potentially not ideal for ABMs. We fitted the PLS using the using the `pls` package [36]. It also takes longer to fit than a simple Linear Regression. And lastly, PLS is not a standard statistical technique and ABM developers may therefore not be familiar with it.

Random Forest The RF is another commonly used calibration algorithm [9]. A random forest is an aggregation of decision trees [37]. Whereas in a simple decision tree, all predictors are considered for each split, the RF only considers a subset so that it produces a multitude of uncorrelated decision trees. For continuous variables, the prediction is the average prediction over all trees, whereas for discrete variables the prediction is obtained by a majority vote. We trained 500 decision trees with minimum node size 1 (for classification) or 5 (for regression) and number

of sampled predictors $\sqrt{p}$ (for classification) or $p/3$ (for regression), using the `randomForest` package [38].

Random Forests are well-suited to situations where there are many more predictors than response variables, meaning we can use a RF with less data and not run into issues of collinearity like with LR. They also do not necessarily fit a linear relationship between predictors and response variable. Although RFs are a well-known algorithm, their interpretation can be more difficult because they do not output a single decision tree. As with LR, the random forest does not fit multiple parameters simultaneously unless we create a weighted average.

Multivariate Random Forest The MRF is an expansion of the RF algorithm that allows for fitting multiple response variables at the same time [39]. In a regular uni-variate RF, decision trees are made by at each node choosing the split that minimises the sum of squared error within the resulting two populations. This sum of squared errors is between predicted and actual response variable, and is usually measured as the Euclidian distance between those two. By using the Mahalanobis distance instead, it becomes possible to optimise each choice for not one but multiple response variables at the same time.

Since MRFs are very computationally intensive to train, we only trained 10 trees per MRF. The minimum node size and number of sampled predictors were the same as for the RF. We fitted the MRFs using the `MultivariateRandomForest` package [40].

The MRF algorithm has largely the same advantages as the RF algorithm, although it is much slower to train. In addition it can be trained on the full parameter set at the same time, although the more parameters are added the slower the algorithm becomes. Lastly, MRFs are probably the least well-known out of all four algorithms discussed here and their interpretation is just as difficult as for the RF.

Chapter 3: Results

Two main issues prevented the execution of the methodology for the Wolf Sheep Predation ABM as described in Chapter 2. First, when fitting the linear regression, all full time series as well as the summarised data with a point every 10 time steps had to be shortened (in the temporal dimension) in order to prevent rank-deficient fits. Second, a number of runs of the PLS and MRF algorithms took well over 24 hours. Therefore, the non-summarised data with a point every 10 and 20 time steps were shortened and the full non-summarised time series were excluded completely for PLS. The MRF had such excessively long run times that only 2 out of 7 parameters were fitted (initial number of sheep and sheep gain from food) and only 100 (randomly chosen) out of the 200 parameter combinations in the training data were used. For further details, see Appendix B.

3.1 An overview of Agent-Based Model calibration issues

Table 3.1 shows an overview of all issues mentioned in the Introduction (see Chapter 1) and highlights which were and were not be addressed in this study.

Table 3.1: Overview of calibration issues. Issues not addressed in this study are italicised.

Step	Issues	Solution
General	Calibration steps are combined and not studied separately	Study steps and combinations of options separately
	Cannot use traditional likelihood	Transform into optimisation problem
Sampling	Cannot run full parameter space	*Use HPC / parallel computing*
		Sample parameter space
	Many options for sampling strategies, unclear which should be used	*Study effect of different sampling designs on calibration performance*
	Unclear how to decide on a sample size	*Study effect of different sample sizes on calibration performance*
	Models are stochastic	Run each parameter combination multiple times
Summarising model output	ABMs produce a lot of output data	Summarise output data so that it is easier / faster to work with
	Implicit spatial summarising	*Study effect of spatial summarising on calibration performance*
		Study options for spatial calibration
	Summarising stochastic runs	Study effect of run summarising
	Explicit temporal summarising	Study effect of temporal summarising on calibration performance
		Study options for temporal calibration
	Many possible summary statistics, unclear how to select and what the consequences are	*Study effect on calibration performance*
Obtaining and summarising real data	Uncertainty in real data	Study effect of noise on calibration performance
	Data is unavailable	
	Mismatch between scale of parameters and scale of data	*Sensitivity analysis to indicate parameters importance, which warrant further study*
	Lack of observational data on what parameter values should be	*Sensitivity analysis to indicate parameter importance, which warrant further study*
	Real-world data and ABM output are on different scales	*Summarise either one so that they are on the same scale*
Choosing a distance function	Optimising multiple outputs simultaneously	*Study effect of (weighted) summarising on calibration performance*
		Study calibration performance of multiple multivariate algorithms
Minimising distance function	Algorithms not all accessible to non-statisticians	Provide clear overview of optimal calibration methodology
		Develop software to aid in calibration
	Local minima in search space	*Use algorithms with perturbance / multiple start options*
	Over-calibration / over-fitting	Study effect of noise on calibration performance
	Interdependent parameters	Use efficient algorithms to lessen computing time
	Non-linear relationships between parameters and output	Study calibration performance of linear vs. non-linear algorithms
	Parameter identifiability	Study relationship between parameters and model output to find unidentifiable parameters
	Equifinality	Acknowledge and take into account when calibrating
	Large amount of parameters requires large training samples / large amounts of run time	Use efficient algorithms to reduce computation time

3.2 The influence of data on calibration performance

3.2.1 Time series vs. time points

There was no discernible difference between including only the endpoint or also including the midpoint when calibrating the Fire ABM, although the estimation of the *directions* parameter was slightly less consistent when including both mid- and endpoint (see Figure 3.1 and Table 3.2a). The Wolf Sheep Predation NRMSE increased with an increasing number of time steps and then dropped back when considering the full time series (see Table 3.2b). The same worsening trend showed in the Point Prediction Performance (PPP). Noticeably though, all the worst scores were the results of LR and PLS (see Figure 3.1.B). After removing these there was a small but consistent improvement in the NRMSE when adding more time points (see Table 3.2b).

Table 3.2: Time series vs. time points

(a) Fire ABM

	mid- and endpoint μ (σ)	endpoint only μ (σ)
F1 score	0.71 (0.08)	0.71 (0.04)
MCC	0.40 (0.15)	0.40 (0.09)
Kappa	0.40 (0.15)	0.40 (0.09)
NRMSE	0.05 (0.03)	0.05 (0.02)
PPP	-23.20 (19.00)	-24.10 (19.70)

(b) Wolf Sheep Predation ABM

	endpoint μ (σ)	50 time steps μ (σ)	20 time steps μ (σ)	10 time steps μ (σ)	time series μ (σ)
NRMSE	0.09 (0.06)	0.14 (0.14)	0.92 (1.79)	1.66 (2.88)	0.62 (1.58)
PPP	0.25 (0.25)	0.08 (0.36)	-0.91 (2.42)	-3.59 (6.81)	-2.51 (7.75)
NRMSE (excluding LR and PLS)	0.079 (0.06)	0.078 (0.05)	0.077 (0.065)	0.076 (0.07)	0.068 (0.07)

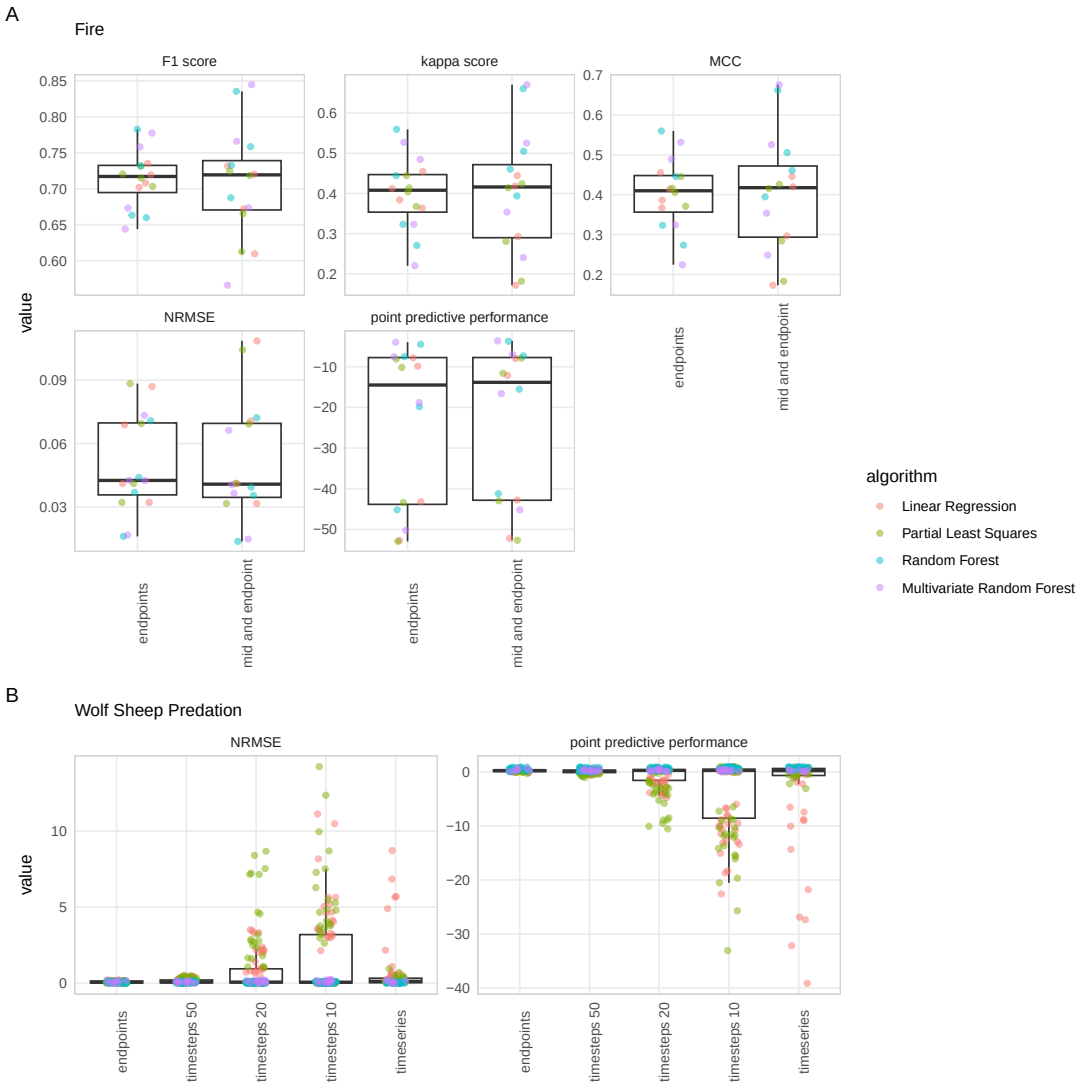


Figure 3.1: Comparison of the effect of the number of included time points on the calibration of the A) Fire and B) Wolf Sheep Predation ABMs. Every panel shows a different error metric. Each data point represents a dataset-algorithm combination. Here, the points are coloured by algorithm and split by the number of time points included in the data. Differences in summarising (see Figure 3.2) and noise (see Figure 3.3) are thereby not shown here. Boxplots show the median, quartiles, and overall range of the overlying points. Means and standard deviations are reported in corresponding Table 3.2. Note that the bimodal nature of the data in the continuous parameters for A) is caused by the split between data with summarised repeated runs vs. non-summarised repeated runs (see Figure 3.2).

3.2.2 Summarising repeated runs of the same parameter combination

Summarising the data made little difference for the estimation of the discrete *directions* parameter in the Fire ABM (see Figure 3.2.A and Table 3.3a), but slightly worsened the estimation of the continuous *density* parameter in terms of NRMSE (see Table 3.3a). Although the differences in mean performance for the Wolf Sheep Predation ABM were not much larger than for the Fire ABM (see Table 3.3), the summarised data yielded many more of the extremely poor performances, especially in combination with LR and PLS (see Figure 3.2 and Table 3.3b).

Table 3.3: Summarising vs. not summarising repeated runs

(a) Fire ABM

	not summarised μ (σ)	summarised μ (σ)
F1 score	0.72 (0.71)	0.71 (0.05)
MCC	0.41 (0.14)	0.39 (0.10)
Kappa	0.41 (0.14)	0.39 (0.10)
NRMSE	0.03 (0.01)	0.07 (0.02)
PPP	-39.7 (13.80)	-7.54 (2.62)

(b) Wolf Sheep Predation ABM

	not summarised μ (σ)	summarised μ (σ)
NRMSE	0.04 (0.09)	1.40 (2.39)
PPP	0.20 (1.36)	-2.81 (6.16)
NRMSE (excluding LR and PLS)	0.02 (0.01)	0.13 (0.04)
PPP (excluding LR and PLS)	0.61 (0.25)	0.28 (0.17)

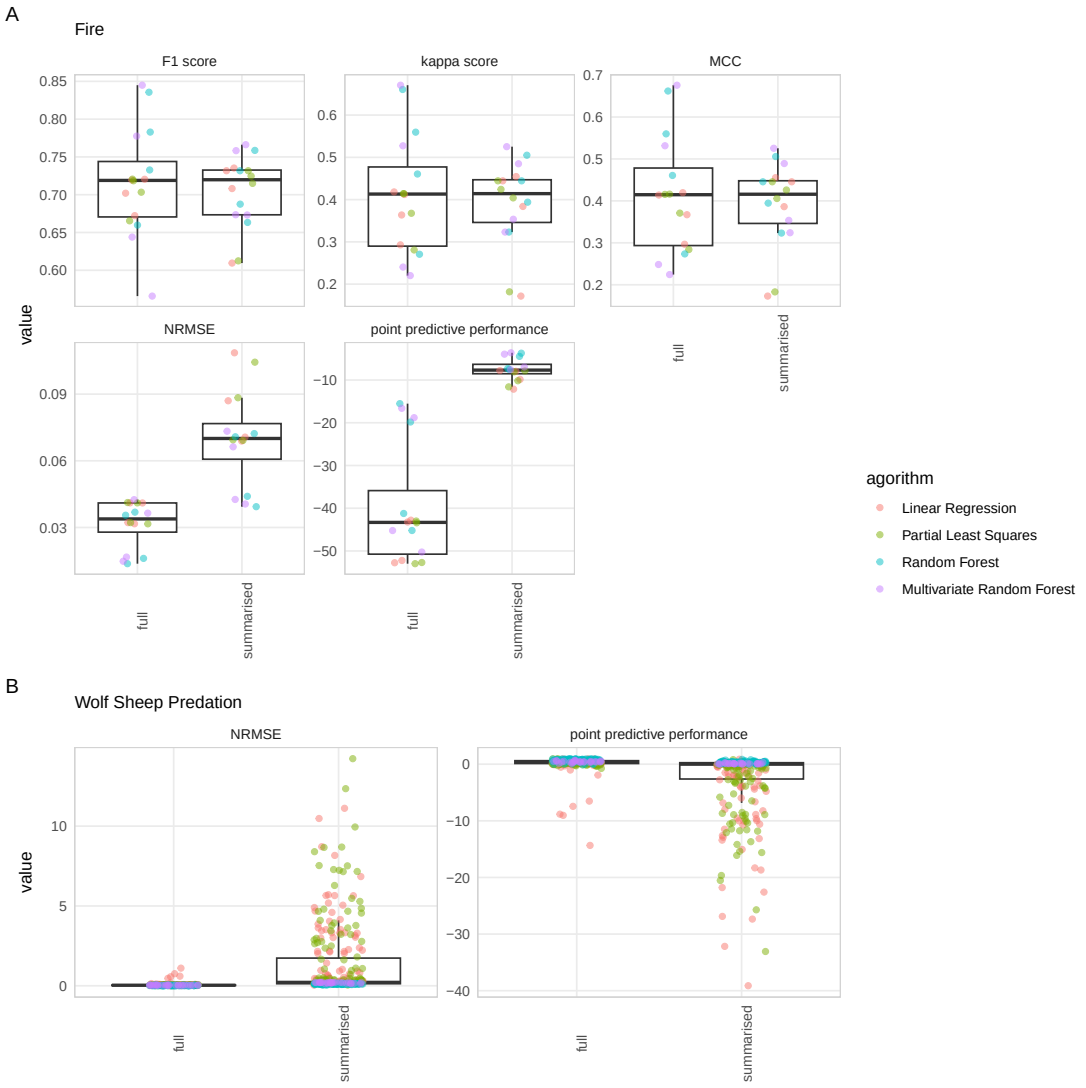


Figure 3.2: Comparison of the effect of summarising repeated runs on the calibration of the A) Fire and B) Wolf Sheep Predation ABMs. Every panel shows a different error metric. Each data point represents a dataset-algorithm combination. Here, the points are coloured by algorithm and split by whether the data was summarised per repeated run or not. Differences in time points (see Figure 3.1) and noise (see Figure 3.3) are thereby not shown here. Boxplots show the median, quartiles, and overall range of the overlying points. Means and standard deviations are reported in corresponding Table 3.3.

3.2.3 Noise in the test data

In the Fire ABM, adding noise to the data deteriorated the calibration outcomes (see Figure 3.3.A and Table 3.4a). For the Wolf Sheep Predation ABM, the NRMSE went down with increasing noise levels, but so did the PPP (see Figure 3.3 and Table 3.4b). When removing the

LR and PLS, we saw a clearer trend where the NRMSE increased when adding noise and the PPP decreased (see Table 3.4b).

Table 3.4: Clean vs. noisy data

(a) Fire ABM

	clean μ (σ)	noise μ (σ)
F1 score	0.75 (0.04)	0.67 (0.04)
MCC	0.49 (0.08)	0.32 (0.08)
Kappa	0.49 (0.08)	0.32 (0.08)
NRMSE	0.04 (0.02)	0.06 (0.03)
PPP	-18.20 (15.8)	-29.10 (20.90)

(b) Wolf Sheep Predation ABM

	clean μ (σ)	output noise μ (σ)	temporal and output noise μ (σ)
NRMSE	0.64 (1.75)	0.71 (1.52)	0.86 (2.10)
PPP	-0.67 (3.32)	-1.40 (3.98)	-1.98 (6.13)
NRMSE (excluding LR and PLS)	0.07 (0.06)	0.08 (0.06)	0.08 (0.06)
PPP (excluding LR and PLS)	0.55 (0.29)	0.42 (0.22)	0.32 (0.23)

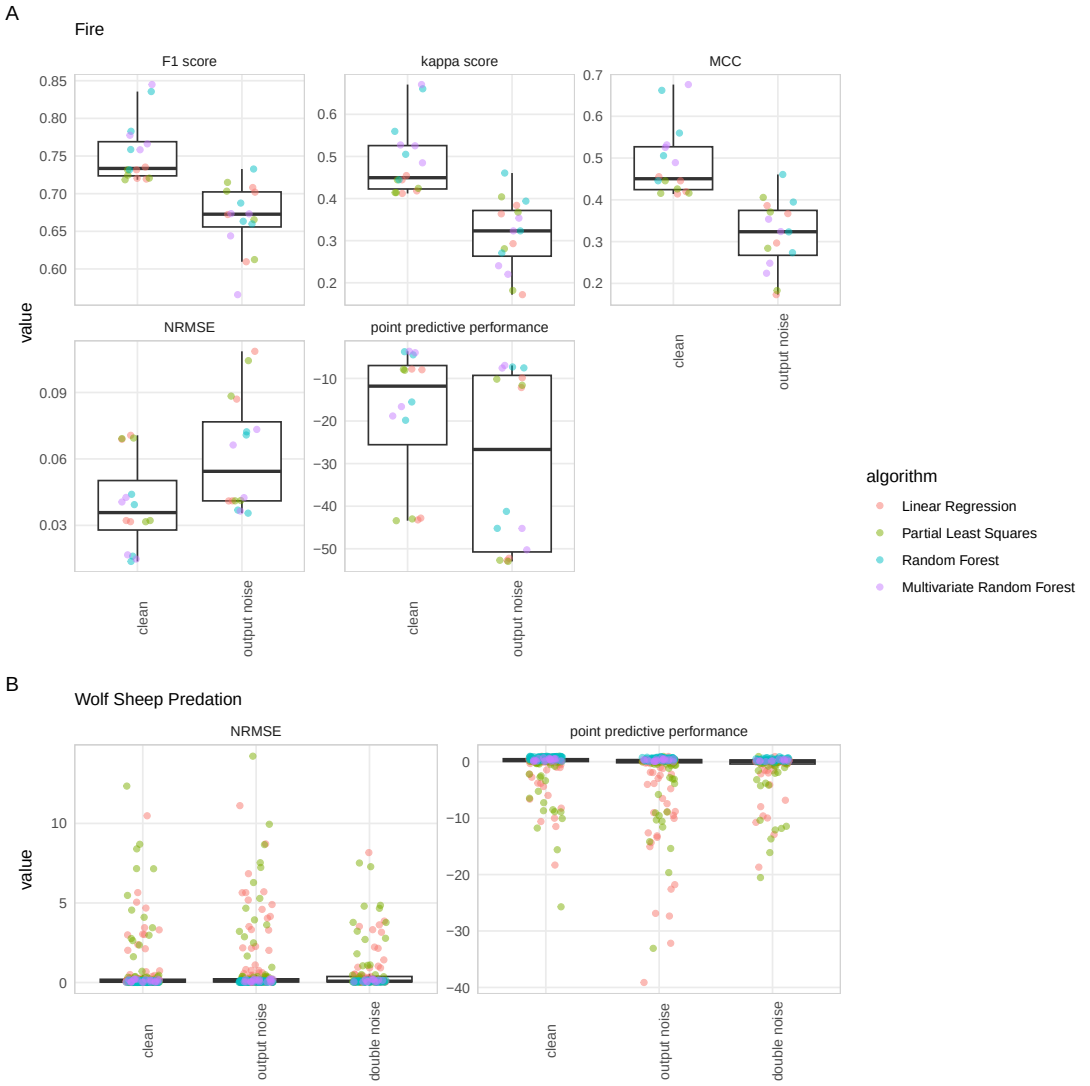


Figure 3.3: Comparison of the effect of noise on the calibration of the A) Fire and B) Wolf Sheep Predation ABMs. Every panel shows a different error metric. Each data point represents a dataset-algorithm combination. Here, the points are coloured by algorithm and split by noise levels. Differences in time points (see Figure 3.1) and summarising (see Figure 3.2) are thereby not shown here. Boxplots show the median, quartiles, and overall range of the overlying points. Means and standard deviations are reported in corresponding Table 3.4. Note that the bimodal nature of the data in the continuous parameters for A) is caused by the split between data with summarised repeated runs vs. non-summarised repeated runs (see Figure 3.2).

3.3 Comparing calibration algorithms

3.3.1 Uni-variate vs. multivariate algorithms

As seen in Figure 3.4 the difference between the uni-variate linear regression and the corresponding multivariate PLS were so small they are negligible (see also Table 3.5). The difference between the uni-variate RF and the corresponding multivariate MRF was also very small (see Figure 3.4). Noticeably, the multivariate methods appeared to show a slightly larger variation in performance than the uni-variate methods (see Table Table 3.5).

Table 3.5: Uni-variate vs. multivariate algorithms

(a) Fire ABM

	linear / logistic regression μ (σ)	partial least squares μ (σ)	random forest μ (σ)	multivariate random forest μ (σ)
F1 score	0.70 (0.04)	0.70 (0.04)	0.73 (0.06)	0.71 (0.09)
MCC	0.37 (0.09)	0.37 (0.09)	0.45 (0.13)	0.42 (0.16)
Kappa	0.37 (0.09)	0.37 (0.09)	0.45 (0.12)	0.42 (0.16)
NRMSE	0.06 (0.03)	0.06 (0.03)	0.45 (0.03)	0.04 (0.02)
PPP	-28.60 (20.80)	-28.70 (21.00)	-18.10 (16.50)	-19.10 (18.5)

(b) Wolf Sheep Predation ABM

	linear regression μ (σ)	partial least squares μ (σ)	random forest μ (σ)	multivariate random forest μ (σ)
NRMSE	1.05 (1.98)	1.33 (2.56)	0.06 (0.05)	0.11 (0.08)
PPP	-2.69 (6.28)	-2.37 (5.33)	0.50 (0.26)	0.27 (0.19)

3.3.2 Linear vs. non-linear algorithms

The linear algorithms linear / logistic regression and PLS were consistently outperformed by the non-linear RF and MRF approaches, given the same data (see Figure 3.4).

3.3.3 Best and worst algorithm

When estimating discrete parameters, the MRF scored best on all error metrics, although it also yielded the worst score in one of the three cases (see Table 3.6a). When estimating continuous parameters the RF scored best on the NRMSE measure, but not on the PPP. The linear / logistic regression approaches dominated the worst scores (see Table 3.6).

Table 3.6: The best and worst algorithms (as determined by the minimum and maximum of all scores) for each error metric.

(a) Fire ABM

Error metric	Best Algorithm (Score)	Worst Algorithm (Score)
F1	Multivariate Random Forest (0.85)	Multivariate Random Forest (0.57)
MCC	Multivariate Random Forest (0.68)	Logistic Regression (0.173)
Kappa	Multivariate Random Forest (0.67)	Logistic Regression (0.172)
NRMSE	Random Forest (0.01)	Linear Regression (0.11)
PPP	Multivariate Random Forest (-3.63)	Partial Least Squares (-53.00)

(b) Wolf Sheep Predation ABM

Error metric	Best Algorithm (Score)	Worst Algorithm (Score)
NRMSE	Random Forest (< 0.01)	Partial Least Squares (14.20)
PPP	Partial Least Squares (0.95)	Linear Regression (-39.10)

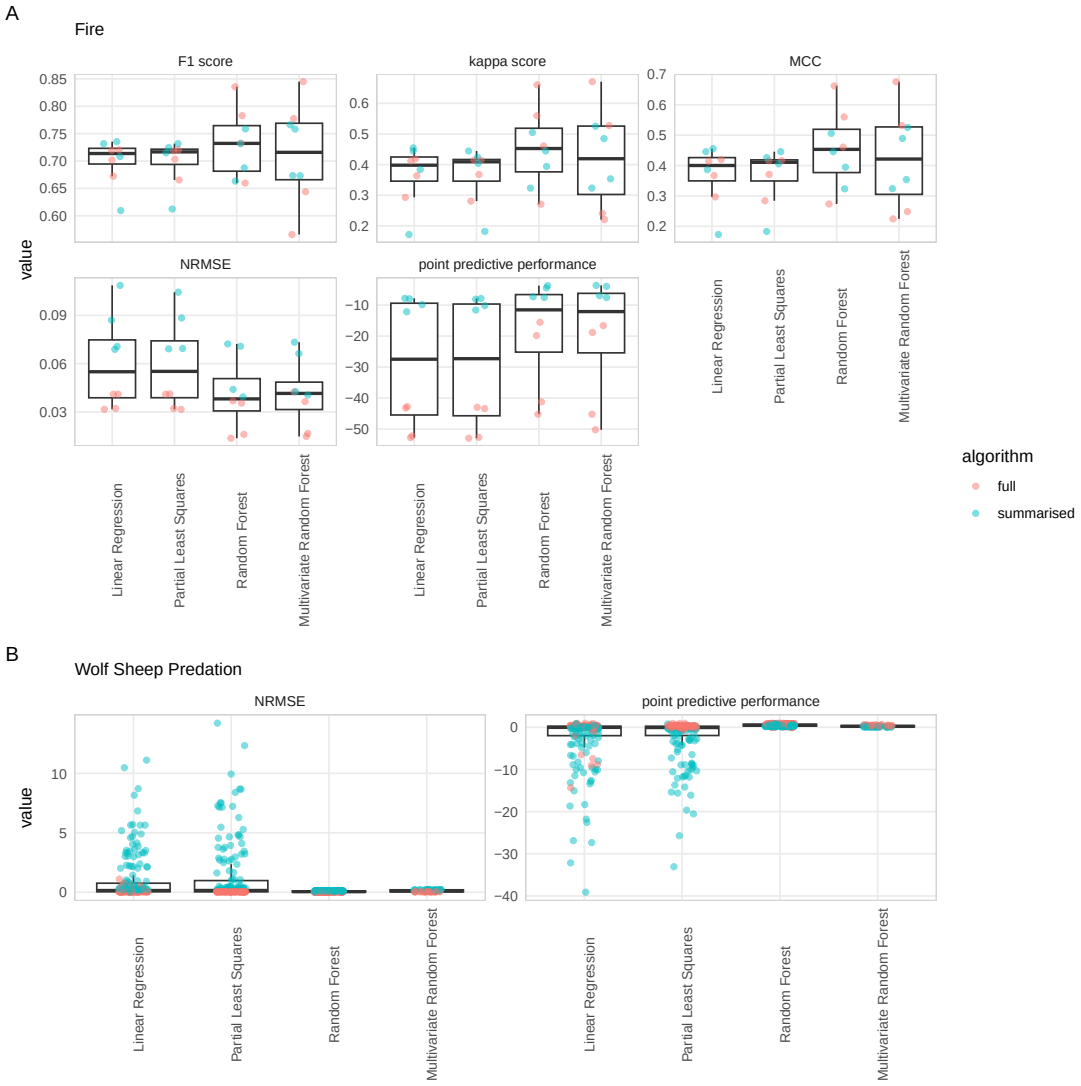


Figure 3.4: A comparison of calibration results for the four different algorithms. A) The performance for the discrete *directions* (F1, kappa, MCC) and *density* (NRMSE, PPP) parameters of the fire ABM. B) The performance for the 7 continuous parameters of the Wolf Sheep Predation ABM. Every panel shows a different error metric. Each point represents a dataset and is coloured based on whether that dataset contained summarised repeated runs or not. Points are divided by the 4 different algorithms. Boxplots show the median, quartiles and, and overall range of the overlying points. Corresponding means and standard deviations for each algorithm are shown in Table 3.5.

3.4 Observations

The two parameters from the Fire ABM were estimated most successfully (see Figure 3.5). The performance for the Wolf Sheep Predation ABM was especially poor, given that the highest percentage of correctly estimated parameters did not exceed three.

Computation times were not formally recorded, but we observed that LR was fastest, followed by RF, PLS, and finally MRF, in that order. It should however be noted that in order to be able to fit the LR, data with long time series and small sample sizes may have to be truncated to prevent rank-deficient fitting.

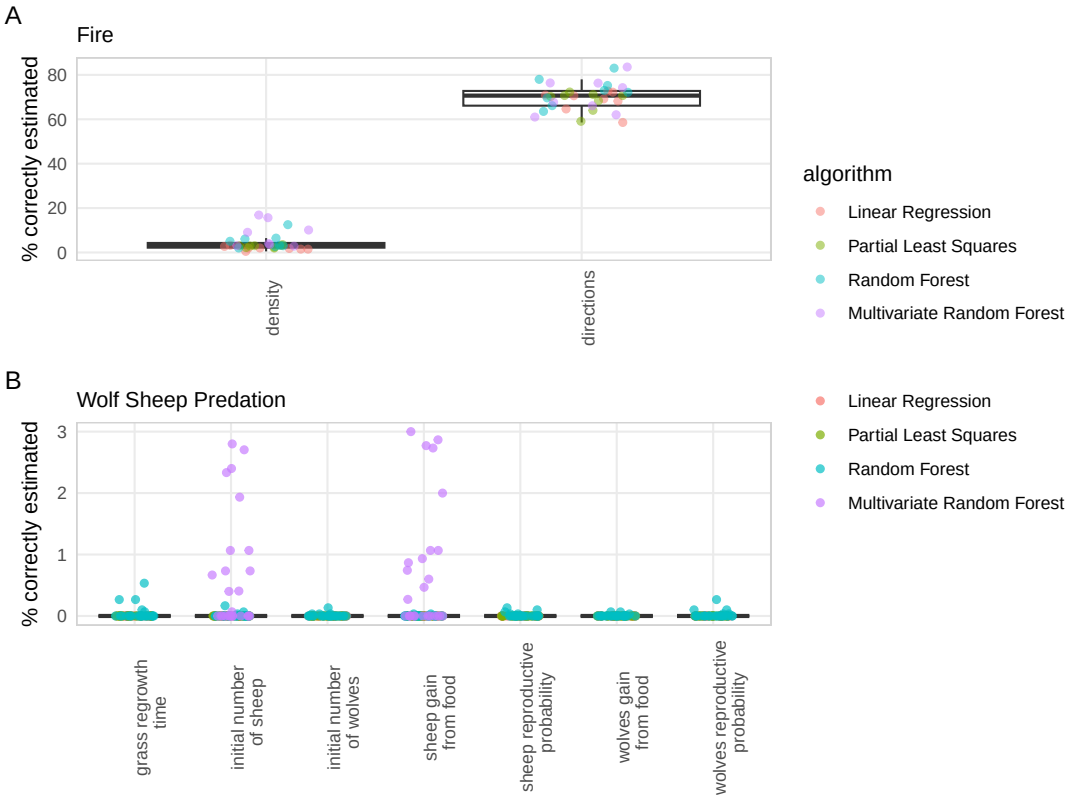


Figure 3.5: Percentages of correctly estimated parameters. The percentage of correctly estimated parameters is shown per dataset combination and coloured according to the algorithm that produced it. Points are shown for each parameter in the Fire (A) and Wolf Sheep Predation (B) ABMs separately. Boxplots show the median, quartiles and, and overall range of the overlying points.

Chapter 4: Discussion and Conclusion

ABM calibration methodology has not yet been standardised, and many new methods are introduced without proper comparison to existing methods [5, 6, 9]. In addition, many aspects of calibration methodology such as the influence of sampling, summarising ABM output, and the uncertainty in calibration data, are currently understudied. As a result, no single best calibration strategy has been identified [9]. This study investigated the performance of different calibration methods, including using time series vs. time points, using summarised vs. non-summarised repeated runs, the influence of noise in test data, uni-variate vs. multivariate algorithms, and linear vs. non-linear algorithms. By studying the influence of these factors on calibration performance, this study aids in the path towards a standardised calibration strategy for ecological ABMs.

The Fire (see Appendix A) and Wolf Sheep Predation (see Appendix B) ABMs were used to investigate calibration performance. Using the approximately 200 samples, a data set was created with and without summarised repeated runs. The ABMs were inspected regarding parameter space, equifinality, unidentifiable parameters, and temporal auto-correlation. Training data was created and used to train a simple linear / logistic regression (LR), partial least squares (PLS), random forest (RF), and multivariate random forest (MRF) algorithm. The results of these calibrations were then compared using testing data with and without noise.

The calibration of the Fire ABM showed no improvement by including more time steps, but this is likely due to the small difference (1 time step) between the datasets used. The Wolf Sheep Predation ABM shows a small improvement in calibration with the addition of more time steps, but only when considering the RF and MRF algorithms. The LR and PLS algorithms appear to be unable to make use of the additional data, instead showing deteriorating performance with the addition of extra time steps. Although summarising repeated runs of the same parameter combination is a very common occurrence, this study shows that this can be detrimental to the calibration of the ABM. The summarising does not affect the discrete parameter much, but the continuous parameters are better calibrated with non-summarised data. This difference is the largest observed out of all the data variations. Part of this result can most likely be explained by the fact that the non-summarised data frames are simply larger and thereby contain more information. Another cause of the poorly performing summarised data, especially for the Wolf Sheep Predation ABM, is likely the fact that the mean is not always a good summary measure. When standard deviations are large and values are not normally distributed around a single mean, using the mean to summarise repeated runs leads to the loss of large amounts of information. Both ABMs showed worse calibration results when noise was added to the test data. The difference for the fire ABM was larger than that for the Wolf Sheep Predation ABM, which may be explained by the much poorer calibration results for the Wolf Sheep Predation ABM in general. In terms of calibration algorithms, there were few notable performance differences between univariate (LR and RF) and multivariate (PLS and MRF) algorithms, but computational times for training were much longer when using multivariate algorithms. Notably though, combining LR

or PLS with data with summarised repeated runs resulted in some of the worst performances. Overall, the MRF performed best, followed by the RF. These non-linear algorithms outperformed the linear PLS and especially LR in almost all cases. The parameters in the Fire ABM were predicted correctly in less than 20% of cases for the density, and in less than 80% of cases for the directions. Calibration for the Wolf Sheep Predation ABM was even poorer, with all parameters being predicted correctly in a maximum of 3% of cases. This may be caused by the fact that the parameters in the Wolf Sheep Predation ABM are barely identifiable, as was discovered during the preliminary analyses steps (see Appendix B). Similarly, the equifinality in the Wolf Sheep Predation ABM likely also contributed to the poor performances.

Strengths and limitations The main issue encountered during this study was computational limitations. Both when running the ABMs and when training the calibration algorithms, some experiments were prohibited by excessive run times. For these reasons, output errors were not computed because that would involve running the ABMs for every value predicted for the test data. Additionally, it is difficult to draw conclusions about the influence of the number of time points in the training data because some data had to be shortened in the time dimension in order to be able to train with it. As a result, the data with more frequent time points did not always contain more information than the data with fewer time points, likely distorting the results.

In contrast to the findings by Carella [9], this study did find differences in performance between different calibration algorithms. Perhaps more importantly, this study found that those differences were smaller than the difference made by summarising vs. not summarising the repeated runs produced by the stochastic ABMs. We therefore recommend to not summarise repeated runs when moving forward, especially when this is not computationally prohibitive. In addition to these results, this study created a structured methodological approach that can be used when further investigating calibration methodology and when introducing new calibration algorithms. At the same time, it also provides a clear overview of which steps in the calibration process (i.e., sampling, pre-training investigations) one should pay attention to when introducing a new calibration algorithm. Lastly, we made a start in identifying which steps in the calibration process are most influential on calibration performance. This is useful information to have when further developing calibration methodology.

This study shows that to make ABM calibration feasible for researchers without excessive computational power available to them, it is very important that any new calibration algorithms are efficient and work well with small sample sizes. Search-based algorithms (such as Genetic Algorithms) do not require the double computational burden of first running the ABM and then training the calibration algorithm. They do, however, need to be re-trained for every case in the test set. Although potentially prohibitive for a study like this one, in practice there likely is just one validation data set (the real-world data). Another option would be to train a surrogate model to recognise the different outcome regions in the parameter space of an ABM, such that subsequent sampling can be more efficient and the calibration algorithm may be trained with a smaller sample size. This may be especially powerful when identifying what type of behaviour is required to replicate the real-world system, such that the calibration algorithm may only need to be trained on a portion of the ABM outcome space. One more idea to focus the sampling effort would be to set the distributions in the Latin Hypercube Sampling such that they reflect the beliefs about what the parameter values should be. Switching coding platforms may also significantly reduce run times. NetLogo is relatively slow, especially when connected to Python via PyNetLogo. Instead, ABMs may be coded directly in Python.

Ideas for future research The calibration performances obtained in this study were sub-optimal. This was likely at least partially due to the unidentifiable parameters and equifinality

in the ABMs. Future studies should include error measures that indicate not just whether parameters are estimated completely correctly, but also how wrong those estimates are. For example, the Fire ABM contains a tipping point, meaning that for parameters around that tipping point a 10% difference between the estimated and the true parameter is much more problematic than a similar difference for parameters further away from the tipping point. For a relatively simple ABM such as the Fire ABM, it is straightforward to come up with an error measure that takes this into account by using different weights. Developing a similar measure for a more complex ABM such as the Wolf Sheep Predation ABM is much less straightforward since the influence of each parameter also depends on the values of the other parameters. We also did not take into account the possibility that in cases where the estimated parameters did not correspond to the true parameters, this may be because they do produce output that corresponds to the output that the predictions were based on, as a consequence of equifinality. Additionally, we did not check how the parameters estimated by the uni-variate algorithms performed when considered as a set. Again, appropriate error measures have to be introduced to further investigate these issues. Introducing approaches more suited to ABM output data may also improve performance. Instead of a standard linear regression, one could use a piece-wise / broken regression to account for some of the non-linearity in the data. If computational times prohibit the use of multivariate algorithms, one can consider using multi-criterion optimisation to ensure that not just the individual parameters but also the parameter combination yields the desired outcome. Although auto- and temporal correlation are considered nuisances, they are are what makes ABMs interesting. An approach that utilises these correlations to inform the calibration may be interesting. The spatial component of ABMs is usually summarised out when calibrating, but especially for movement or migration ABMs it may be desirable to calibrate the spatial component as well. One could do this by training a Convoluted Neural Network on the images of the model space (in time series format) and using aerial photographs as real-world data. Although computationally expensive, this approach would be especially interesting in combination with a digital twin approach to study, for example, changing migration routes in response to increasing human infrastructure.

Within the framework of this study, there is ample room for expansion of the research questions. The following questions can easily be answered using the methodology outlined here:

- What is the effect of sample size on calibration performance. Is there a minimum sample size required to obtain satisfactory results?
- It is recommended to repeatedly run the same parameter combinations due to the stochastic nature of most ABMs. To what extend does it harm the calibration if one chooses to not do this, or does not use enough repeats?
- Would summarising repeated runs be more successful if using a different summary measure than the mean? Not every ABM lends itself to summarising using the default mean, because distributions may, for example, be bimodal. Using, for example, the median instead may still help reduce computational times when training the calibration algorithm, while yielding a more faithful representation of the data.
- When using time points instead of the full time series, can we enhance performance by using a (sliding) window average instead of taking the value at each exact time point?

In addition, more ABMs can be added to allow for a more complete comparison. Since the Fire ABM is fully random and the Wolf Sheep Predation ABM is semi-random, a logical inclusion would be a non-random ABM, i.e. one where the agents move with purpose based on information about their environment. Adding an error measure for the discrepancy between the true outcome

and the outcomes resulting from the predicted parameters will give insight in the extend to which the algorithm predicts parameter combinations that will actually reproduce the desired real-world data. Lastly, this study did not take into account the fact that real-world data is often not available in the shape of (long term) time series.

To conclude, in order to contribute to the work towards a more standardised calibration methodology, this study investigated the effects of summarising repeated runs, using few vs. many time points, the noise in test data, and various algorithms on the calibration of two simple ABMs. The effects of the inclusion of more or fewer time points and / or adding noise to the data were either non-existent or small, but the common practice of summarising repeated runs by taking their mean proves detrimental. In contrast to previous findings, this study did see a small but promising difference in performance between the different algorithms included, indicating that a search for more efficient calibration algorithms may not be in vein. Specifically, the non-linear multivariate MRF performed best, although the results were still relatively poor. The methodology used by this study may be used to further investigate issues in calibration methodology, or when introducing new calibration algorithms.

Future studies using this framework could focus on further investigating the effect of summarising repeated runs, including summary functions other than the mean, the effect of sample sizes on calibration performance, as well as investigating new calibration algorithms such as Convoluted Neural Networks and approaches making use of the temporal- and auto-correlation features common in ABMs.

Bibliography

[1] Elske van der Vaart, Alice S.A. Johnston, and Richard M. Sibly. Predicting how many animals will be where: How to build, calibrate and evaluate individual-based models. *Ecological Modelling*, 326:113–123, 4 2016.

[2] Matthew Oremland and Reinhard Laubenbacher. Optimization of agent-based models: Scaling methods and heuristic algorithms. *JASSS*, 17:6, 2014.

[3] Raphaël Duboz, David Versmisse, Morgane Travers, Eric Ramat, and Yunne Jai Shin. Application of an evolutionary algorithm to the inverse parameter estimation of an individual-based model. *Ecological Modelling*, 221:840–849, 3 2010.

[4] C. C.M. Chen, C. C. Drovandi, J. M. Keith, K. Anthony, M. J. Caley, and K. L. Mengersen. Bayesian semi-individual based model with approximate bayesian computation for parameters calibration: Modelling crown-of-thorns populations on the great barrier reef. *Ecological Modelling*, 364:113–123, 11 2017.

[5] J. Gareth Polhill, Jiaqi Ge, Matthew P. Hare, Keith B. Matthews, Alessandro Gimona, Douglas Salt, and Jagadeesh Yeluripati. Crossing the chasm: a 'tube-map' for agent-based social simulation of policy scenarios in spatially-distributed systems. *GeoInformatica*, 23:169–199, 4 2019.

[6] Steven Manson, Li An, Keith C. Clarke, Alison Heppenstall, Jennifer Koch, Brittany Krzyzanowski, Fraser Morgan, David O'sullivan, Bryan C. Runck, Eric Shook, and Leigh Tesfatsion. Methodological issues of spatial agent-based models. *JASSS*, 23, 1 2020.

[7] Josie McCulloch, Jiaqi Ge, Jonathan A Ward, Alison Heppenstall, J Gareth Polhill, Nick Malleson, Alan Turing Institute, John Dodson House, Euston Rd, and London Nww Wdb. Calibrating agent-based models using uncertainty quantification methods. *JASSS*, 25:1, 2022.

[8] Jan C Thiele, Winfried Kurth, and Volker Grimm. Facilitating parameter estimation and sensitivity analysis of agent-based models: A cookbook using netlogo and r. *JASSS*, 17:11, 2014.

[9] Ernesto Carrella. No free lunch when estimating simulation parameters. *JASSS*, 24:7, 2021.

[10] Jennifer Badham, Chipp Jansen, Nigel Shardlow, and Thomas French. Calibrating with multiple criteria: A demonstration of dominance. *JASSS*, 20:11, 2017.

[11] Juan Francisco Robles, Enrique Bermejo, Manuel Chica, and Óscar Cordón. Multimodal evolutionary algorithms for easing the complexity of agent-based model calibration. *JASSS*, 23:4, 2021.

[12] Ernesto Carrella, Richard Bailey, and Jens Madsen. Calibrating agent-based models with linear regressions. *JASSS*, 23:7, 2020.

[13] Romain Reuillon, Clara Schmitt, Ricardo De Aldama, and Jean-Baptiste Mouret. A new method to evaluate simulation models: The calibration profile (cp) algorithm. *JASSS*, 18:12, 2015.

[14] Francesco Lamperti, Andrea Roventini, and Amir Sani. Agent-based model calibration using machine learning surrogates. *Journal of Economic Dynamics and Control*, 90:366–389, 5 2018.

[15] Guus Ten Broeke, George Van Voorn, Arend Ligtenberg, and Jaap Molenaar. The use of surrogate models to analyse agent-based models. *JASSS*, 24:3, 2021.

[16] Ju Sung Lee, Tatiana Filatova, Arika Ligmann-Zielinska, Behrooz Hassani-Mahmooei, Forrest Stonedahl, Iris Lorscheid, Alexey Voinov, Gary Polhill, Zhanli Sun, and Dawn C. Parker. The complexities of agent-based modeling output analysis. *JASSS*, 18, 10 2015.

[17] Maksat Ashyraliyev, Yves Fomekong-Nanfack, Jaap A. Kaandorp, and Joke G. Blom. Systems biology: Parameter estimation for biochemical models. *FEBS Journal*, 276:886–902, 2 2009.

[18] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[20] U Wilensky. Netlogo, 1999.

[21] Marc Jaxa-Rozen and Jan H. Kwakkel. Pynetlogo: Linking netlogo with python. *Journal of Artificial Societies and Social Simulation*, 21, 2018.

[22] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[23] Ken Arnold, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

[24] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, PBC., Boston, MA, 2020.

[25] U Wilensky. *NetLogo Fire model*, 1997.

[26] U Wilensky. *NetLogo Wolf Sheep Predation model*, 1997.

[27] Iris Lorscheid, Bernd Oliver Heine, and Matthias Meyer. Opening the 'black box' of simulations: Increased transparency and effective communication through the systematic design of experiments. *Computational and Mathematical Organization Theory*, 18:22–62, 3 2012.

[28] Michael D Byrne. How many times should a stochastic model be run? an approach based on confidence intervals. In *Proceedings of the 12th International conference on cognitive modeling, Ottawa*, 2013.

[29] Alboukadel Kassambara and Fabian Mundt. *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*, 2020. R package version 1.0.7.

[30] Danny Matthew Saputra, Daniel Saputra, and Liniyanti D Oswari. Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method. In *Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)*, pages 341–346. Atlantis Press, 2020.

[31] D Joubert, JD Stigter, and J Molenaar. Determining minimal output sets that ensure structural identifiability. *PLoS One*, 13(11):e0207334, 2018.

[32] Thorsten Pohlert. *trend: Non-Parametric Trend Tests and Change-Point Detection*, 2023. R package version 1.1.5.

[33] Kuhn and Max. Building predictive models in r using the caret package. *Journal of Statistical Software*, 28(5):1–26, 2008.

[34] Ben Gorman. *mltools: Machine Learning Tools*, 2018. R package version 0.3.5.

[35] Michael Haenlein and Andreas M Kaplan. A beginner's guide to partial least squares analysis. *Understanding statistics*, 3(4):283–297, 2004.

[36] Kristian Hovde Liland, Bjørn-Helge Mevik, and Ron Wehrens. *pls: Partial Least Squares and Principal Component Regression*, 2022. R package version 2.8-1.

[37] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

[38] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[39] Mark Segal and Yuanyuan Xiao. Multivariate random forests. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, 1(1):80–87, 2011.

[40] Raziur Rahman. *MultivariateRandomForest: Models Multivariate Cases Using Random Forests*, 2017. R package version 1.1.5.

Appendix A: Fire

The fire ABM is a relatively simple application from the NetLogo Models Library [25]. It models the spread of fire through a stand of trees. Based on the density of trees, the ABM fills patches with trees. The fire starts on the left side of the simulated space and spreads through the stand by expanding from one patch with trees to another. In the original ABM, the fire can spread in 4 directions (east, west, north, south). Here, we add an option for fire spreading in 8 directions (i.e. to all surrounding patches). The input parameters therefore become the tree density $(0 - 99\%)$ and the number of directions that the fire can spread in $(\{4, 8\})$. The output we use is the percentage of burned trees at each tick of the simulation. This is computed as

$$\% \text{ burned trees} = \frac{\text{number of burned trees}}{\text{initial number of trees}} * 100. \tag{A.1}$$

This ABM exhibits tipping point behaviour with regard to the fire reaching the right side of the simulation space. There is a tree density (depending on the number of directions of fire spread) below which the fire is almost certain not to reach the right side of the simulation space, and above which the fire is almost certain to reach the right side of the simulation space.

A number of issues arises from the fire ABM data. First, the fire ABM produces simulations of varying lengths, according to how many time steps it takes for the embers to die out. Since most algorithms cannot deal with missing data, we artificially lengthen the time series to equal lengths by copying the last value of the simulation. This creates complete, but nearly perfectly collinear data because the data of the later time steps will only differ in a few simulations. Additionally, the multi-collinearity of the extended data makes it difficult to fit time steps data using a linear regression. Doing so leads to rank-deficient fits, causing the subsequent predictions to be unreliable. This issue persists when taking time steps every 10 or 20 ticks. The issue lessens when taking time steps every 50 ticks, but since around 40% of the simulations has fewer than 50 ticks this is not seen as a viable option. Second, the minimum and maximum burn percentages are not included as a variable. The minimum burn percentage is 0 for all simulations (the burn percentage at the start). The maximum burn percentage is always identical to the burn percentage at the last time step. These variables are therefore not informative and are excluded from the analysis. Third, by running each parameter combination 5 times we obtain a maximum of 990 time series. This implies that it will not be possible to fit regressions with close to or more than 990 parameters. This eliminates the possibility of fitting the full time series using a linear regression. Therefore, we do not run the options with data every 10, 20, and 50 time steps because the training sample is not large enough to uniquely identify the parameters of such datasets in, for example, the linear regression. It may be possible to solve this issue by creating more training data, but since the outputs do not differ widely from one simulation to the next and we are already using the full parameter space, doing so would most likely create collinear data.

To avoid these issues, the fire model regressions are only fit with the endpoint, middle point,

mean, standard deviation, and trend as predictors. There is a clear relationship between initial tree density and resulting burn percentage at the end of the simulation (see Figure A.3), suggesting that not much information is lost by only using the final time step. However, this relationship is less clear for the number of directions. An alternative approach would be to fit a ridge or elastic net regression, which can deal with multi-colinearity.

A.1 Preliminary

A.1.1 Running the Agent-Based Model

Determining the number of model runs To determine the number of runs per parameter combination we run the fire ABM 50 times with densities 5, 50, and 95 and number of directions 4 and 8. This gives 6 parameter combinations to inspect. All computations are performed on the burn percentage at the end of the simulations. The change in mean from one simulation to the next immediately settles below 1 (see Figure A.1.A) and is therefore not very informative. The coefficient of variation settles below 0.1 within 4 runs for all parameter combinations (see Figure A.1.B). The relative outer variance fails to converge within 50 runs for 3 of the parameter combinations. For the other 3 it only dips below 0.2 and then increases again or is 0 the entire time because there's no variation in the runs. Since it is unfeasible to run the ABM more than 50 times, we will use the outcomes of the mean convergence and coefficient of variation experiments only and use 5 runs per parameter combination.

Sampling parameter space Since the parameter space of the fire ABM is small, we can simply run the full parameter space without any need for sampling. There are 198 parameter combinations created by setting the density between 1 and 99 for both 4 and 8 directions. The ABM allows for setting the density to 0 but since that yields a simulation of 0 ticks (there is no fire to spread because there is no fuel for the fire) we do not include this setting.

Summarising repeated runs Inspecting the standard deviation of the burn percentage per parameter combination shows that most parameter combinations show little variation (a small standard deviation), with the exception of a few with very large standard deviations (see Figure A.2.A). A number of parameter combinations (notably all with mid-range tree densities) produce large standard deviations ($sd > 10$) and are inspected individually (see Figure A.2). This shows that in most cases using the mean to summarise the runs will still produce values that are in line with the actual outputs, although some information will be lost about the spread of the data. We summarise using the mean.

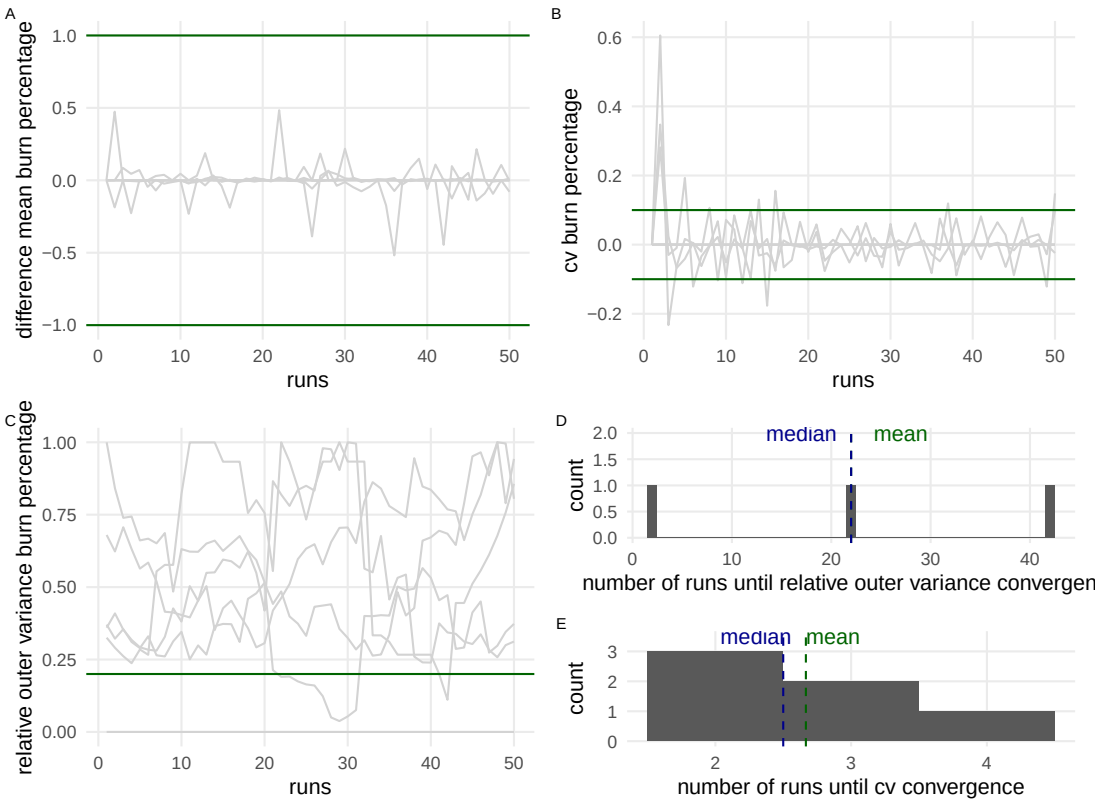


Figure A.1: Determining number of model runs for fire ABM: Figure shows the mean convergence (A), coefficient of variation (B & D), and relative outer variance (C & E) that inform the number of repetitions per parameter combination.

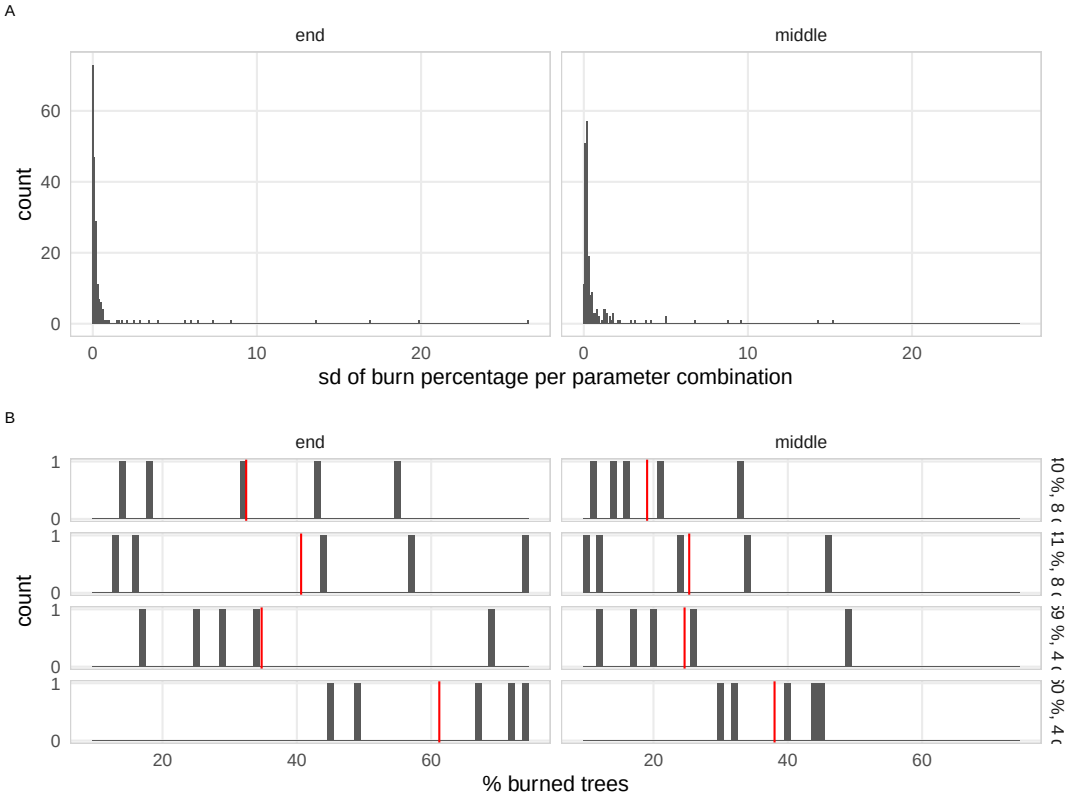


Figure A.2: Inspecting summarising options for fire ABM: The figure shows the standard deviations of the burn percentage within each parameter combination for the middle and end of the simulations (A). Additionally, those parameter combinations with standard deviations larger than 10 are shown separately (B).

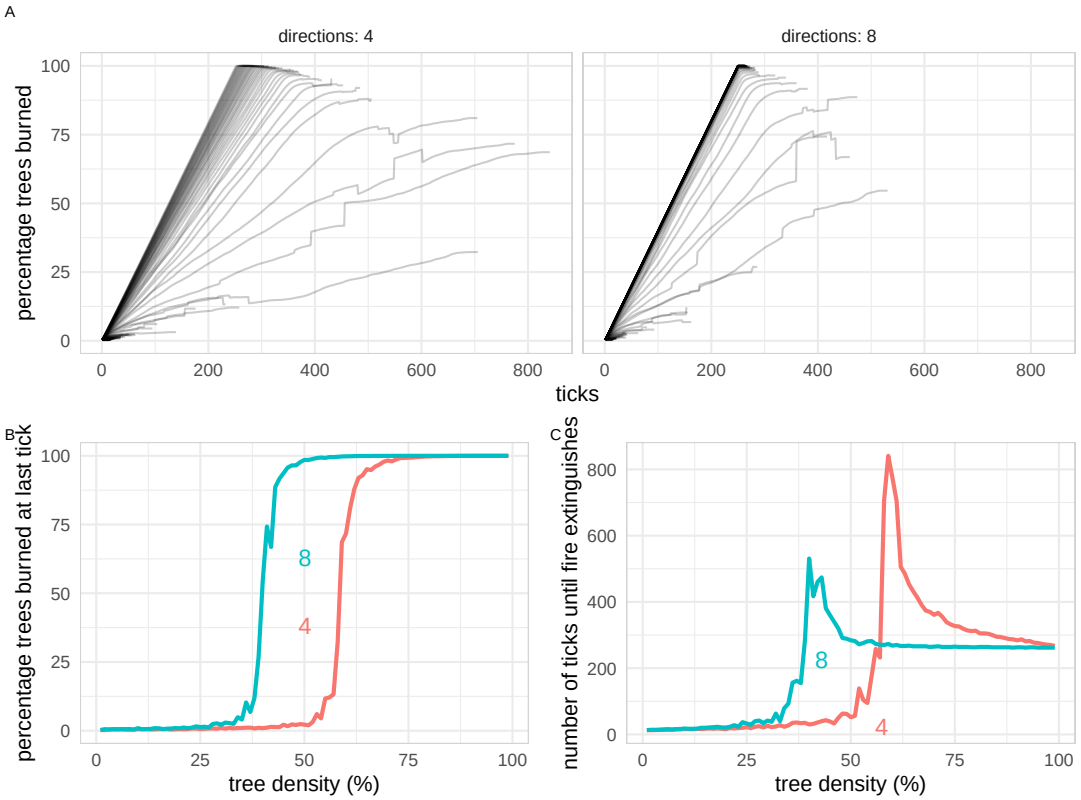


Figure A.3: Fire ABM parameter space: Plot showing the speed of burning (A), the total burned trees at the end of the simulation (B), and the length of the simulation (C) for different parameter settings of the initial tree density and the number of directions the fire can spread in.

A.1.2 Agent-Based Model inspection

Visualising parameter space The parameter space visualisation show that, as expected, with the fire spreading in 8 instead of 4 directions, the percentage of burned trees increases more quickly and the simulations are generally shorter (see Figure A.3.A and C). Additionally, at low (< 30) and high (> 75) tree densities, the number of directions does not matter for the final burn percentage. At intermediate ($30 - 75$) tree densities, however, the total burn percentage is larger for the simulations where the fire spreads in 8 directions compared to those where in the fire spreads in just 4 directions (see Figure A.3.B). The same figure shows that there is a tipping point in the ABM where before the tipping point the percentage of burned trees remains low, whereas after the tipping point that same percentage is very high.

Identifying equifinality 2-means clustering based on the percentage of burned trees at the middle and final steps of the simulation yields clusters that are fairly distinctive in terms of tree density (see Figure A.4.A) but not so much in terms of directions of fire spread (see Figure A.4.B). This points towards equifinality in terms of the number of directions of fire spread. We can also see this in the parameter space visualisations (Figure A.3). It is clear that higher tree densities lead to higher burn percentages throughout the simulation, but the number of directions

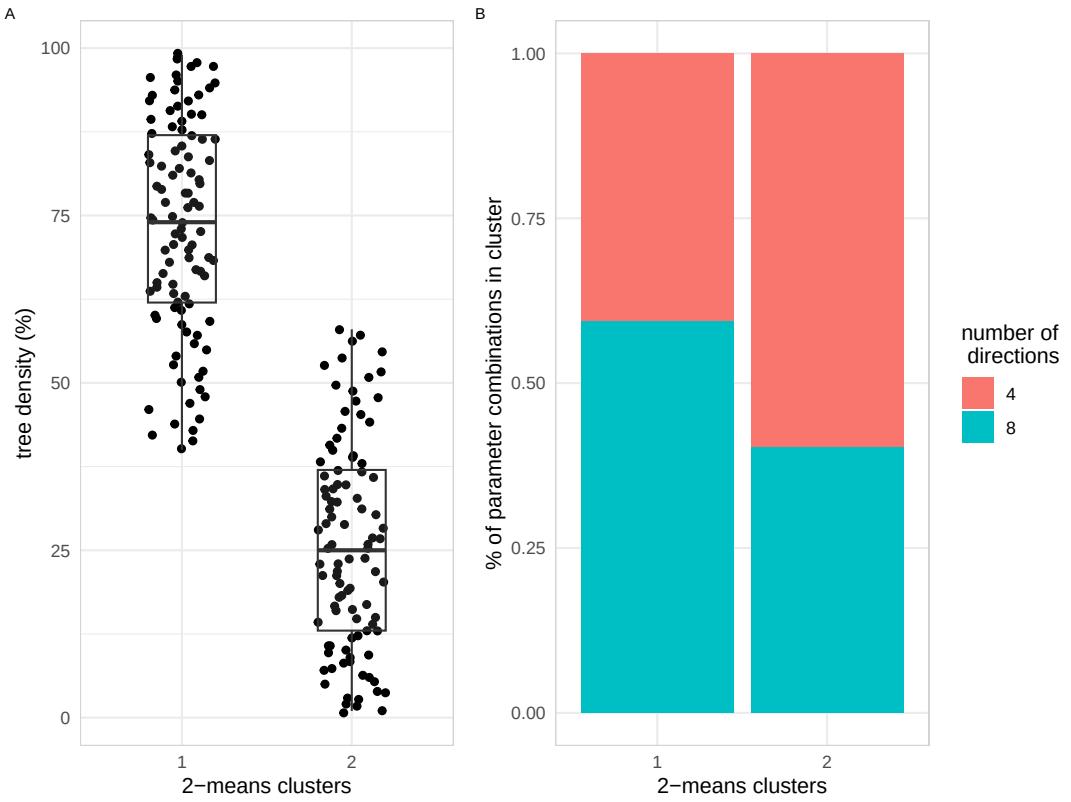


Figure A.4: K-means clustering for fire ABM outputs: Results of 2-means clustering based on the percentage of burned trees at the middle and final steps of the simulation. Shown are the distribution of tree density (A) and directions of fire spread (B) in both clusters.

of fire spread has a less pronounced effect on the burn percentages. It does affect the speed of the simulation, but since we do not include length of simulation as a variable, that effect is lost in the current analysis.

Finding unidentifiable parameters The slopes for the association between tree density and burn percentage are not flat, indicating that the tree density parameter is identifiable with regards to the output variable (see Figure A.5.A). Similarly, the boxplots for 4 and 8 directions are different (see Figure A.5.B). However, both an increase in tree density and an increase in directions leads to an increase in burn percentage, indicating that any algorithm may have trouble distinguishing between the effects of these two parameters.

Temporal auto-correlation plots Figure A.6.A shows that there is considerable auto-correlation for most time series created by the fire ABM. However, most of this disappears when looking at the partial auto-correlation (see Figure A.6.B). Nonetheless, auto-correlation is present, suggesting an algorithm that can take this into account may be expected to perform better when calibrating the fire ABM.

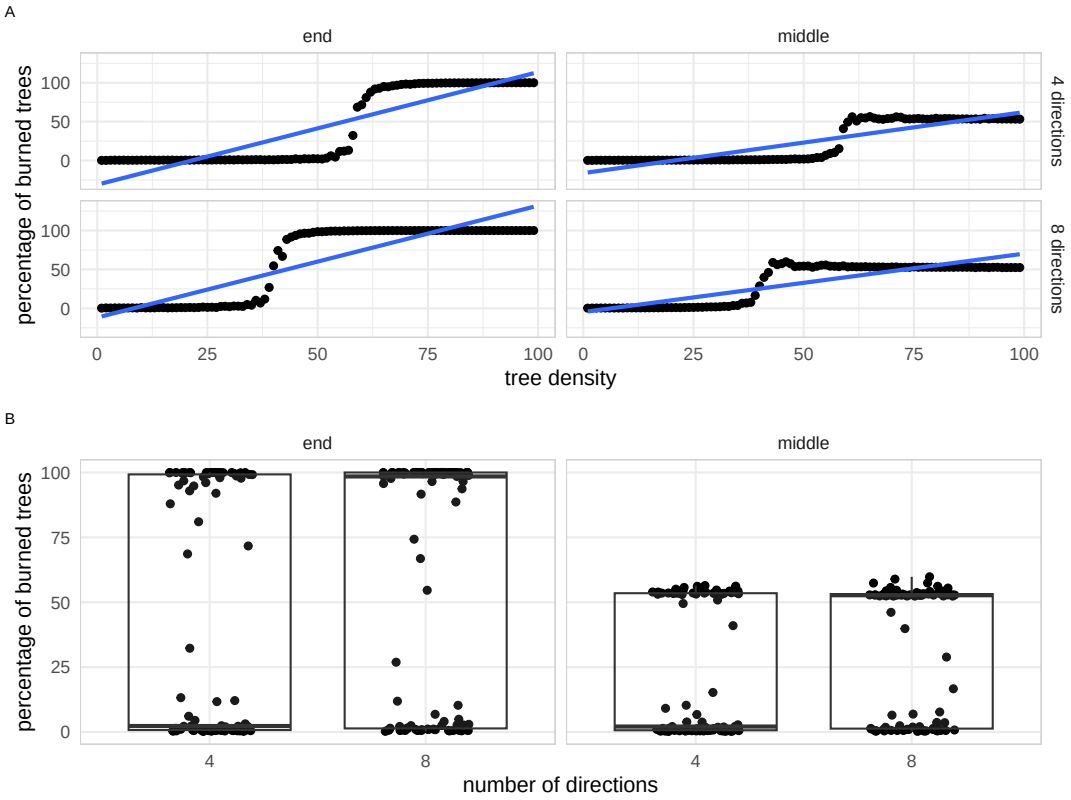


Figure A.5: Finding unidentifiable parameters in the fire ABM: Similar slopes / trends indicate unidentifiable parameters.

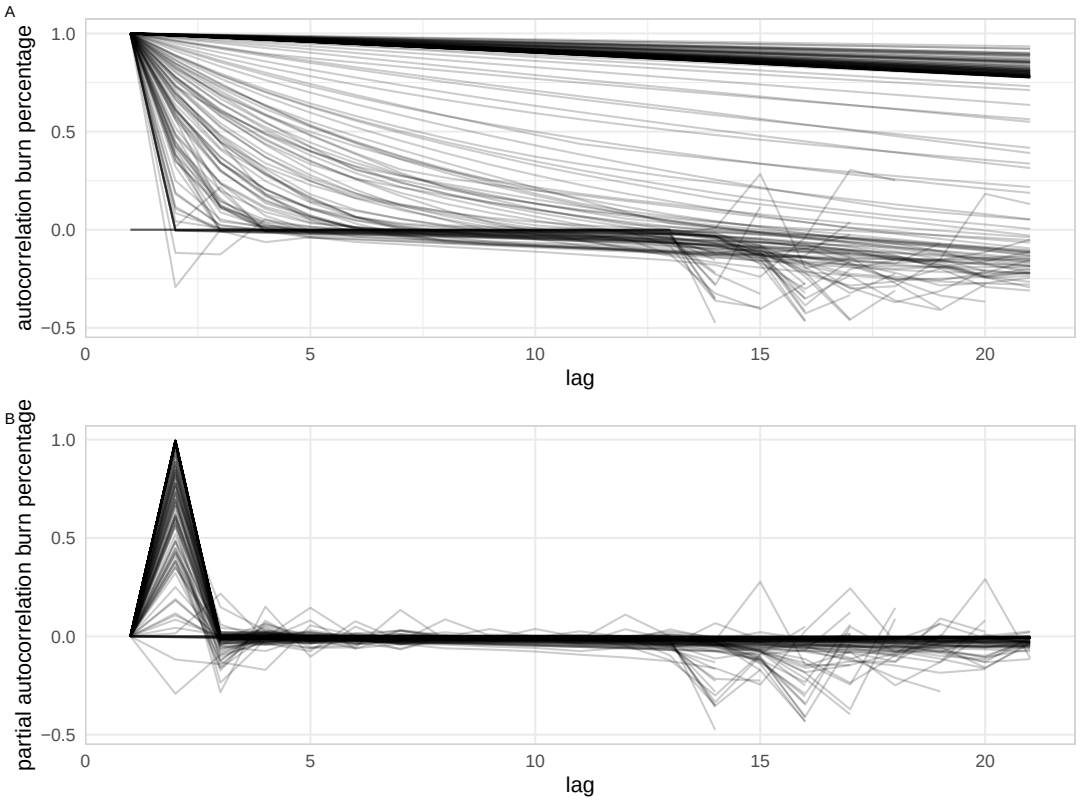


Figure A.6: Auto-correlation for the fire ABM: Figure shows the auto-correlation (A) and partial auto-correlation (B) for the fire ABM.

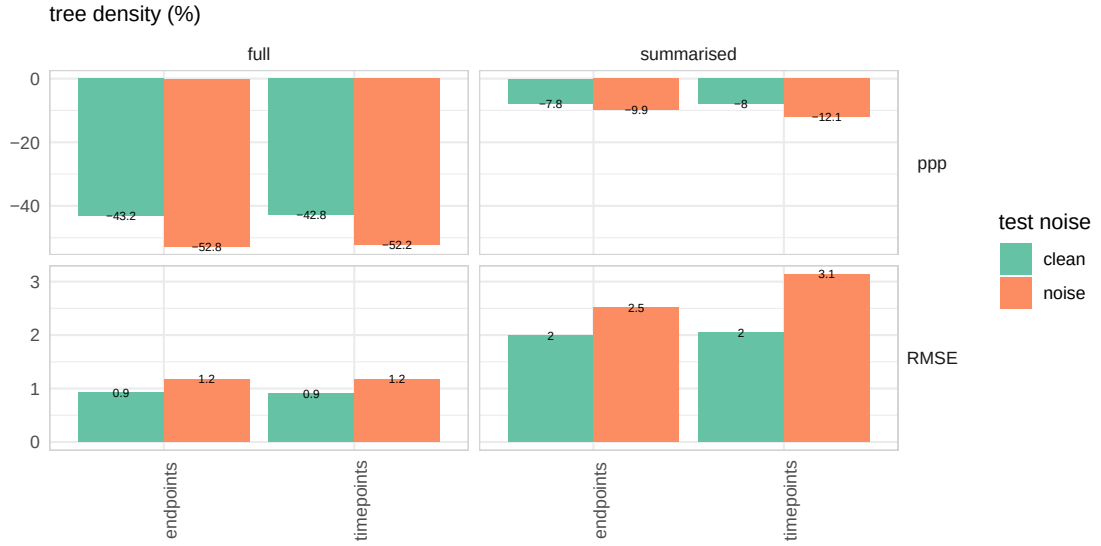


Figure A.7: Fire ABM linear regression errors on the density parameter: The errors as made by the linear regression for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

A.2 Linear regression

Density The regression supplied with clean full data mid and end points yields the highest percentage of correctly estimated densities, as well as the lowest RMSE for the density. This is closely followed by the same data but with endpoints only. The clean data with only endpoints but averaged runs comes close in terms of the percentage of correct densities, but performs worse in terms of the density RMSE. In almost all cases, the clean data outperforms the corresponding experiment with output noise, except for the mid and endpoint data with averaged runs. Here, the noisy data gets more parameters correct (although the difference is still small), but the clean data does perform better in terms of RMSE. The percentage of correctly estimated density parameters is overall very low. The RMSE is tolerable, indicating that although the linear regression does not predict the exact correct density parameter, it is not far off. The point prediction values all indicate mis-identification.

Directions The logistic regression supplied with endpoints, clean data, and averaged runs performs best in terms of percentage of directions correctly predicted, the kappa score, and the F1 score. It is, however, outperformed by the endpoints output noise full data in terms of the MCC score. Kappa scores vary between fair ($(0.2, 0.4)$) to moderate ($(0.41, 0.6)$) agreement. The F1-scores indicate decent precision and recall. The MCC scores show that the logistic regression performs better than random chance, but by how much differs from barely ($MCC = 0.20$ for mid-and endpoint noisy data with averaged runs) to a fair amount ($MCC = 0.53$ for endpoints with output noise and full runs). Especially interesting is the fact that for the clean full data, it makes almost no difference whether the logistic regression is supplied with only endpoints or with both mid- and endpoints.

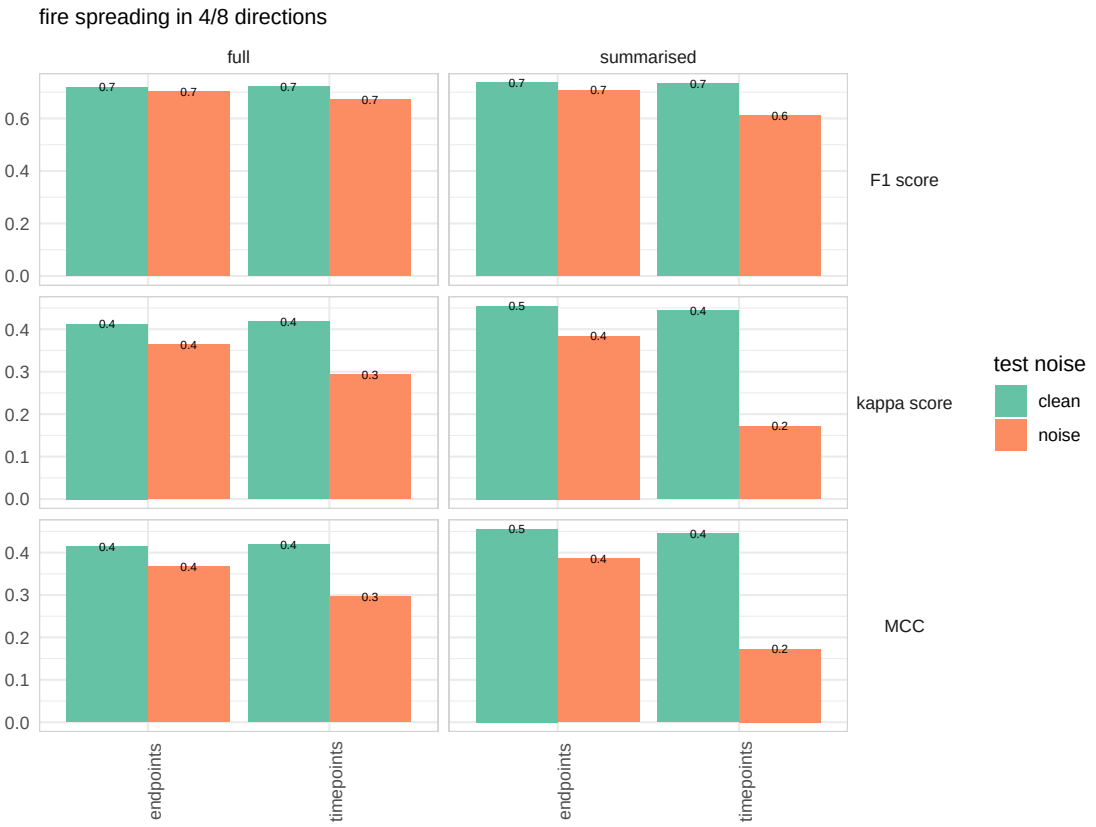


Figure A.8: Fire ABM linear regression errors on the directions parameter: The errors as made by the linear regression for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

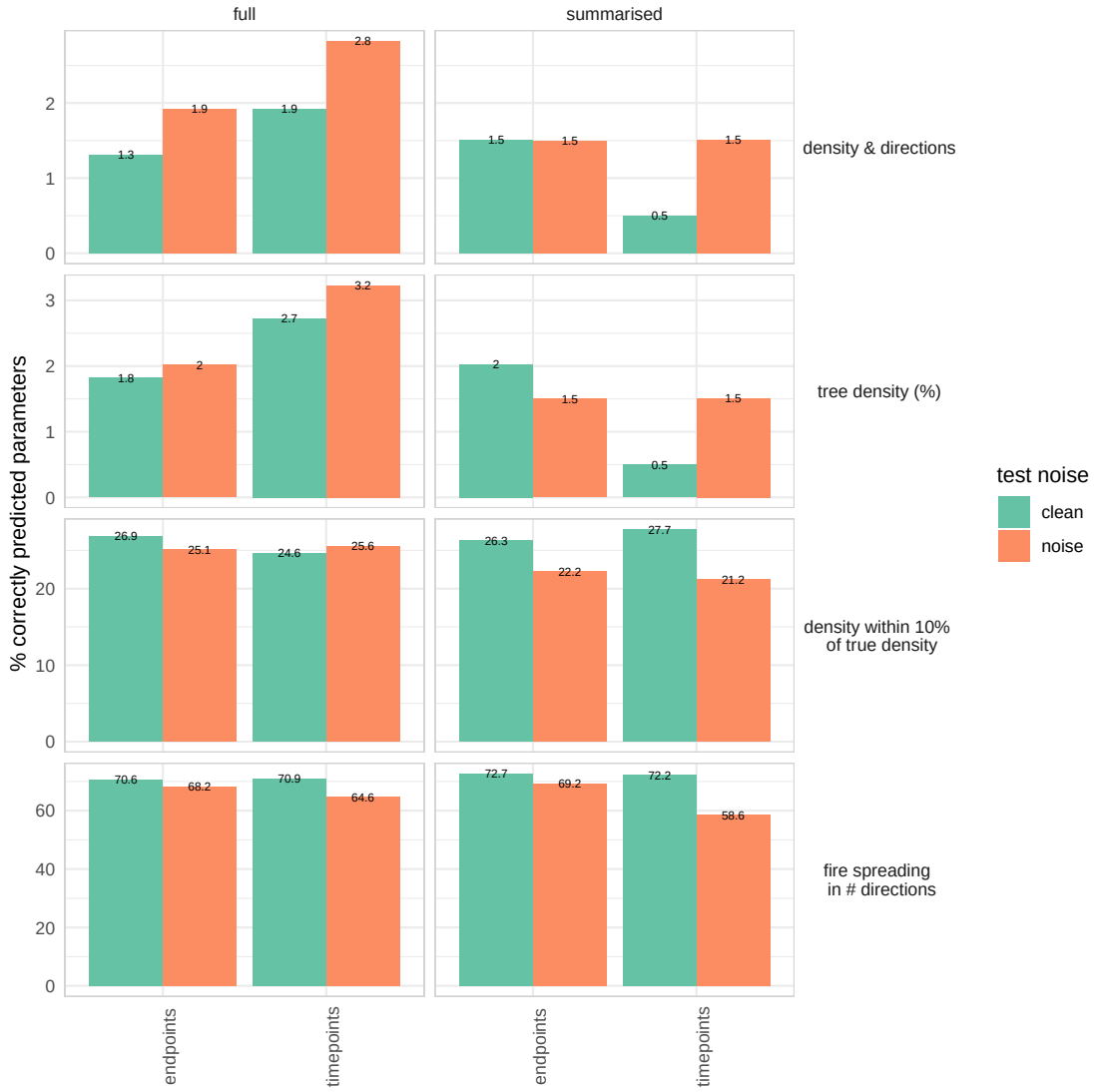


Figure A.9: Fire ABM linear regression errors on both parameters: The errors as made by the linear regression for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

Both parameters Very few parameter combinations are guessed correctly in all cases. The full data with output noise and mid- and endpoints seems to outperform the other datasets very slightly. However, the differences are in all cases very small. Interesting to see here also, is that it apparently makes no difference whether the clean full data comes with endpoints only or with both end- and midpoints.

A.3 Partial Least Squares

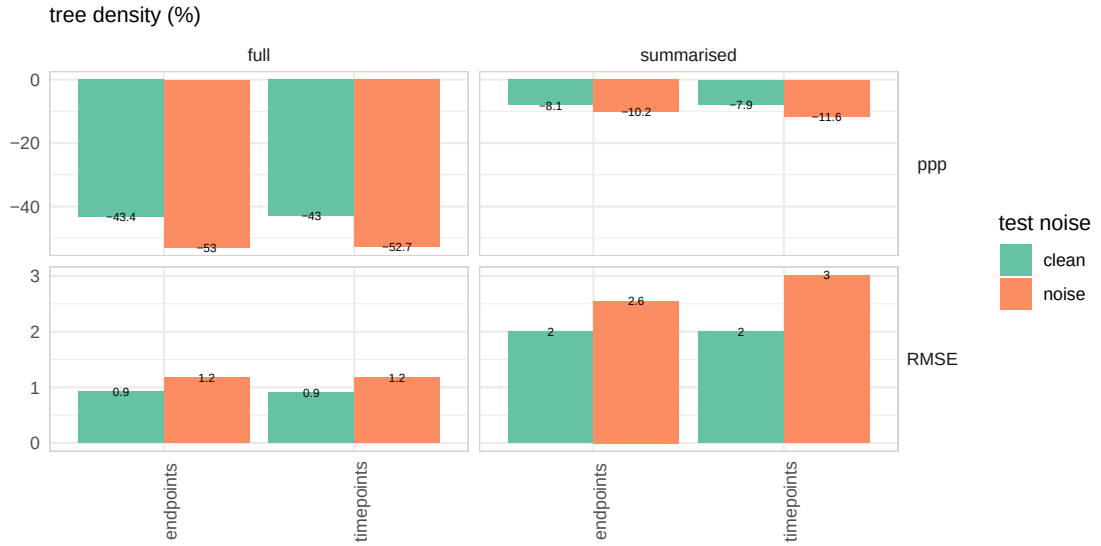


Figure A.10: Fire ABM partial least squares errors on the density parameter: The errors as made by the partial least squares for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

Density None of the data performs particularly well in terms of the percentage of correctly guessed parameters, but both the clean non-summarised data does perform better than the rest in terms of RMSE. The mid- and endpoint data with output noise and averaged runs performs worst in terms of RMSE. The point prediction values all indicate mis-identification.

Directions Performance is relatively constant across all data sets. In terms of the percentage of correctly predicted parameters, both the endpoints only and mid- and endpoints data without noise and with averaged runs perform best. This data also performs the best in terms of kappa, F1, and MCC scores, although the difference with most of the rest of the data is very small. The mid- and endpoint data with output noise and averaged runs performs worst, only slightly better than random guessing would. This data also performs noticeably worse in terms of the kappa, F1, and MCC scores compared to the other data.

Both parameters None of the data sets performs well, but the endpoints only clean non-summarised data performs worst. The endpoints, clean, averaged runs data performs best.

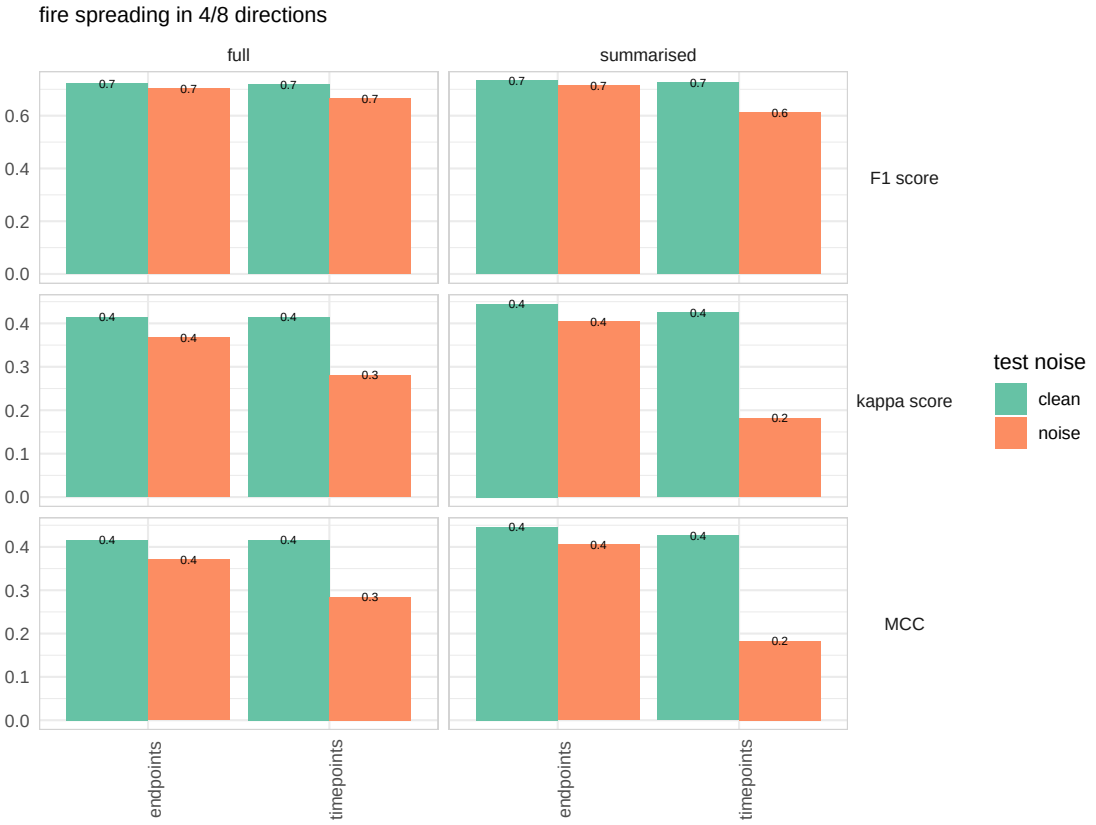


Figure A.11: Fire ABM partial least squares errors on the directions parameter: The errors as made by the partial least squares for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

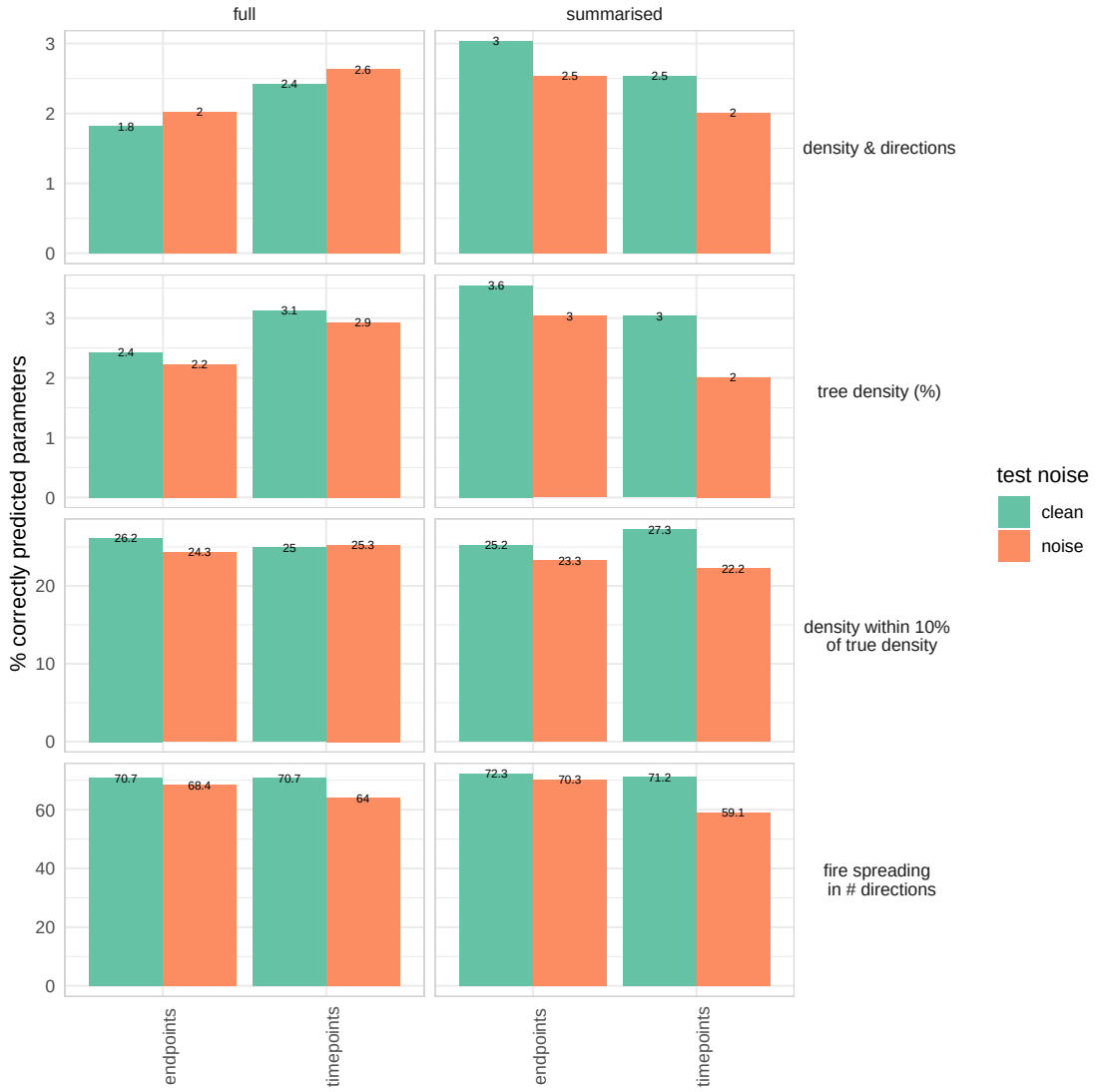


Figure A.12: Fire ABM partial least squares errors on both parameters: The errors as made by the partial least squares for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

A.4 Random Forest

The random forests were trained with the standard settings as per the randomForest R package [38], except the number of trees was set to 1000. The standard parameters were used to prevent the need of determining them via the common visual inspection methods, which hinders the automation of the training process.

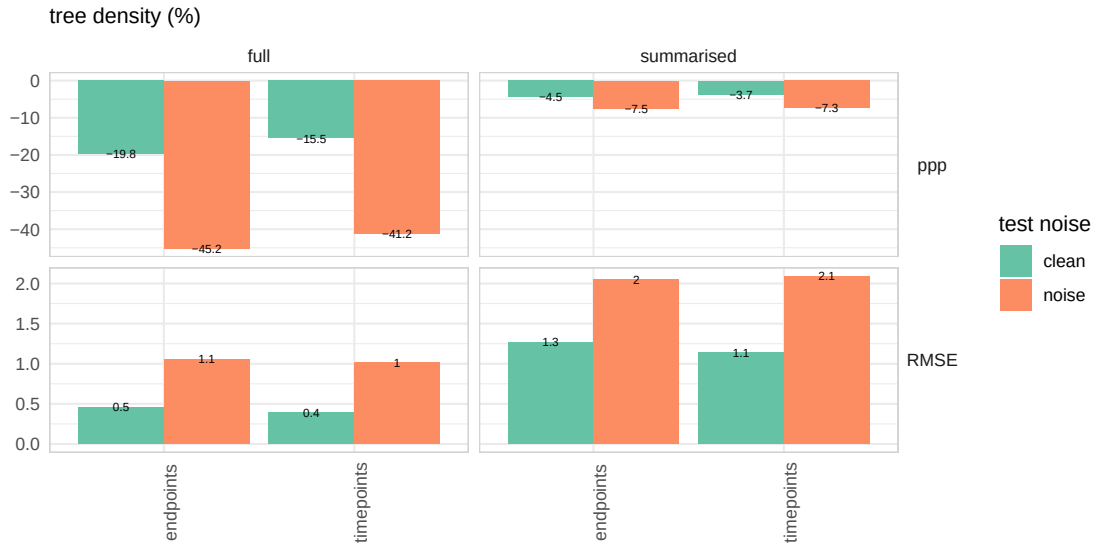


Figure A.13: Fire ABM random forest errors on the density parameter: The errors as made by the random forest for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

Density The clean, non-summarised data with both middle- and endpoints clearly performs best in terms of the percentage of correctly predicted densities. The RMSE scores are best for the non-noisy, non-summarised data. Noticeably, the RMSE suffers more from summarising the data than from adding noise. The point prediction performance scores all indicate mis-identification.

Directions None of the kappa scores are particularly admirable, although the clean non-summarised data with mid- and endpoints achieves a substantial score. The F1 scores indicate fairly good precision and recall. The MCC scores show that the random forest performs better than random guessing, although this varies from barely better (noisy data with only endpoints) to definitively better (non-summarised data without noise).

Both parameters The clean, non-summarised data with both middle- and endpoints clearly performs best in terms of the percentage of correctly predicted densities and directions. The summarised version of the same data performs on the same level as the clean data with only time points. All datasets with noise perform noticeably worse. A 12% performance is still not great, but already much better than what has been obtained so far.

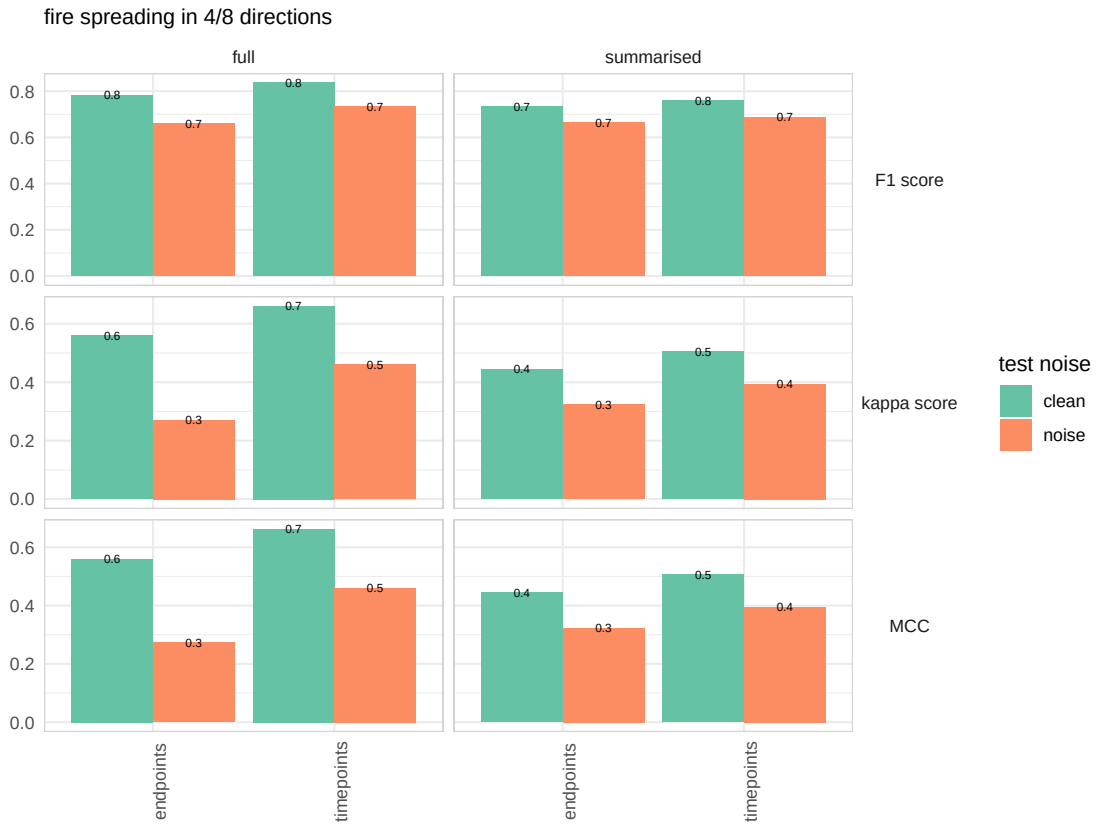


Figure A.14: Fire ABM random forest errors on the directions parameter: The errors as made by the random forest for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

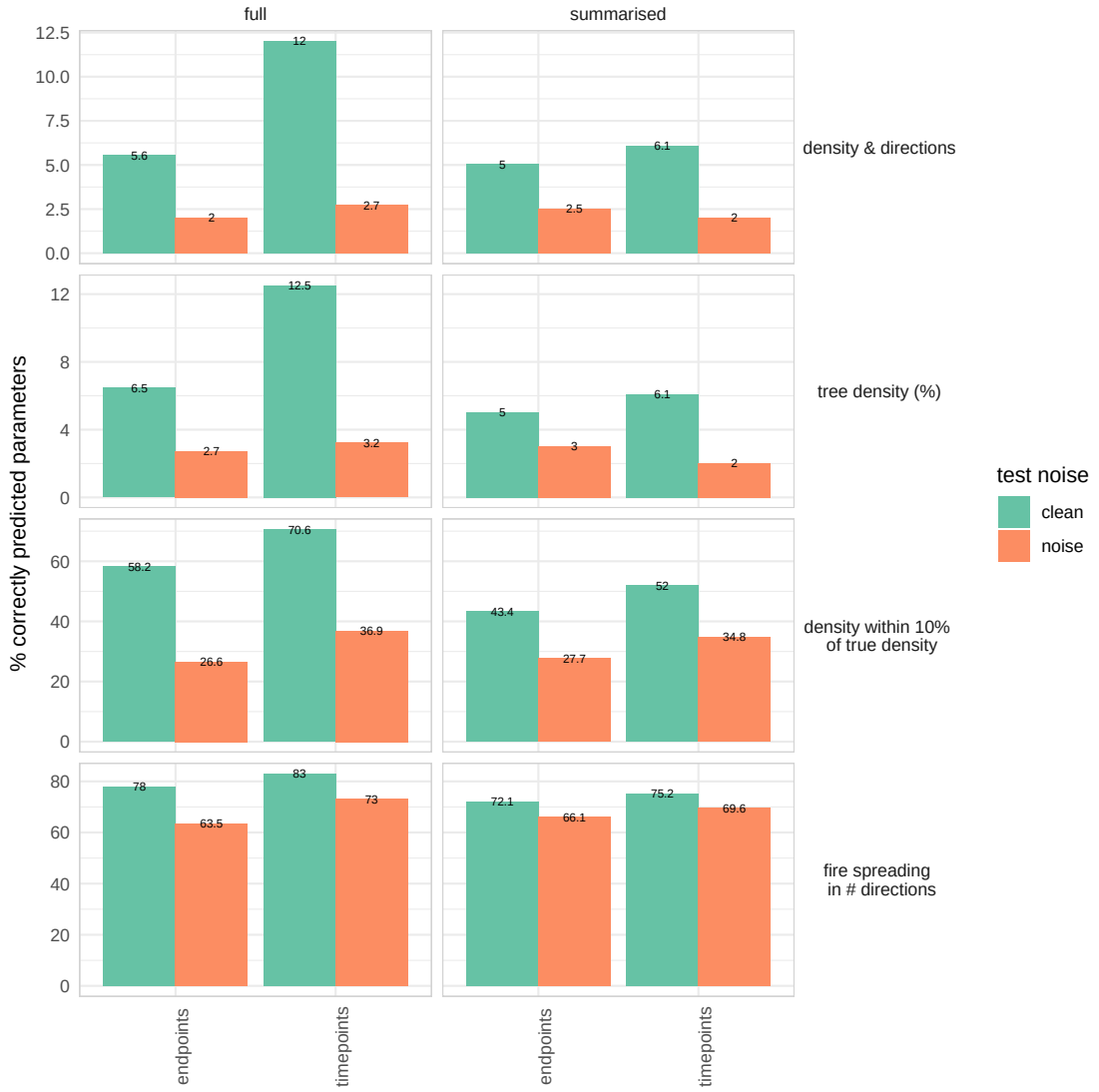


Figure A.15: Fire ABM random forest errors on both parameters: The errors as made by the random forest for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

A.5 Multivariate Random Forest

The point prediction performance indicates misidentified density parameters in all cases (see Figure A.16). Adding noise to the test data clearly increases the RMSE, but there is no clear difference between the endpoints only and mid- and endpoints data. The non-summarised data performs better than the summarised data, but the difference is small.

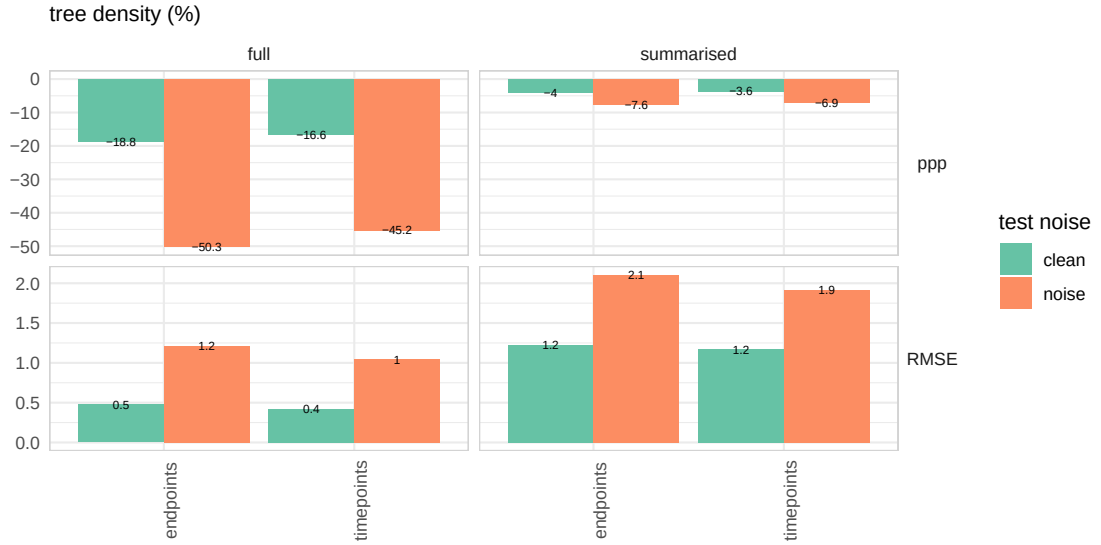


Figure A.16: Fire ABM multivariate random forest errors on the density parameter: The errors as made by the multivariate random forest for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

Matthews Correlation Coefficient indicates that in all cases the Multivariate Random Forest performs better than random guessing (see Figure A.17. The clean, non-summarised data with mid- and endpoints clearly performs best.

The MRF guesses the correct density and directions parameters in 16.9 % of cases, given clean non-summarised data with endpoints only (see Figure A.18). Providing additional midpoints seems to deter the performance regarding estimating the density parameter but not the directions parameter. These differences are, however, very small. Adding noise to the data is clearly detrimental to the parameter estimation.

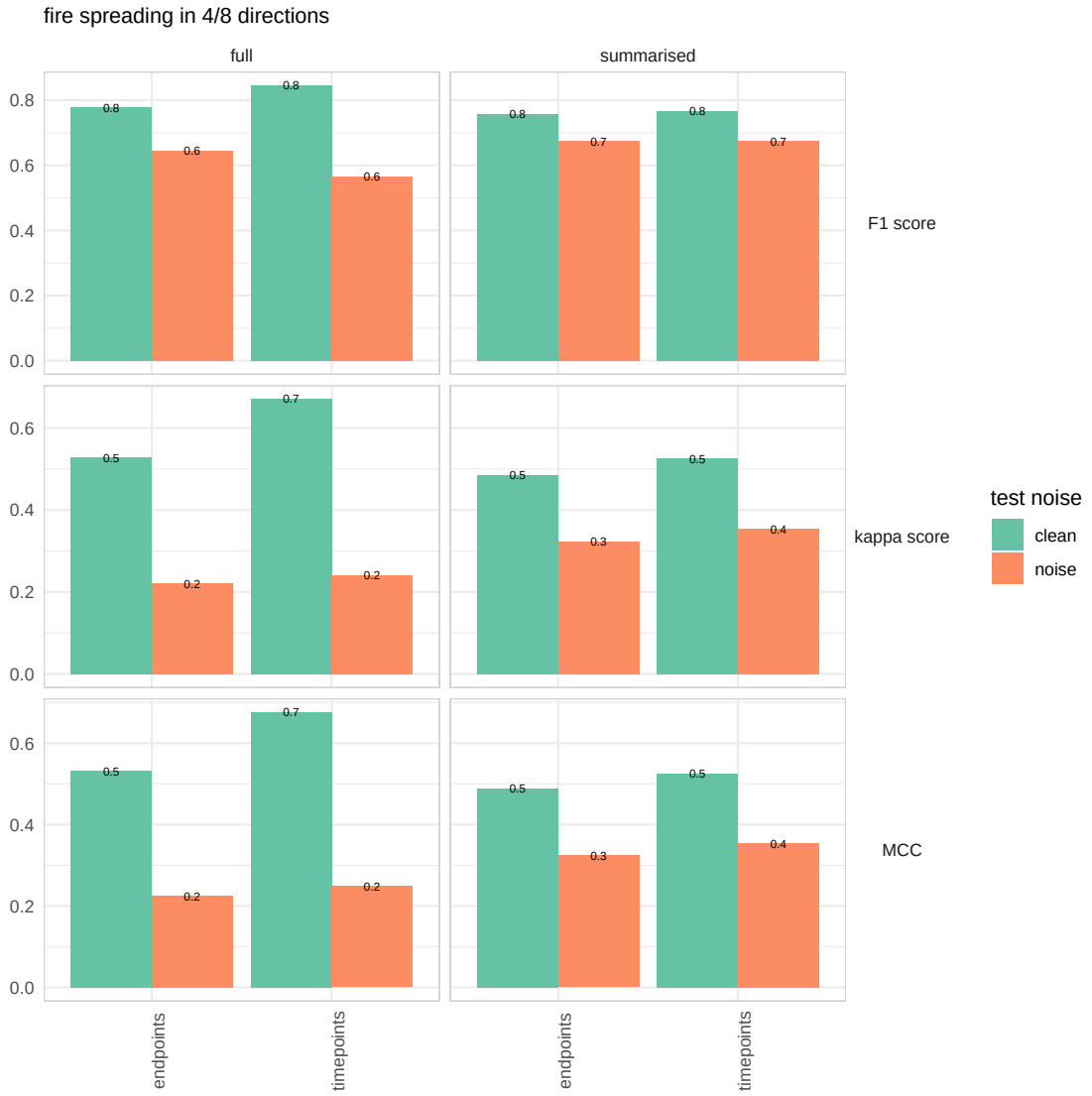


Figure A.17: Fire ABM multivariate random forest errors on the directions parameter: The errors as made by the multivariate random forest for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

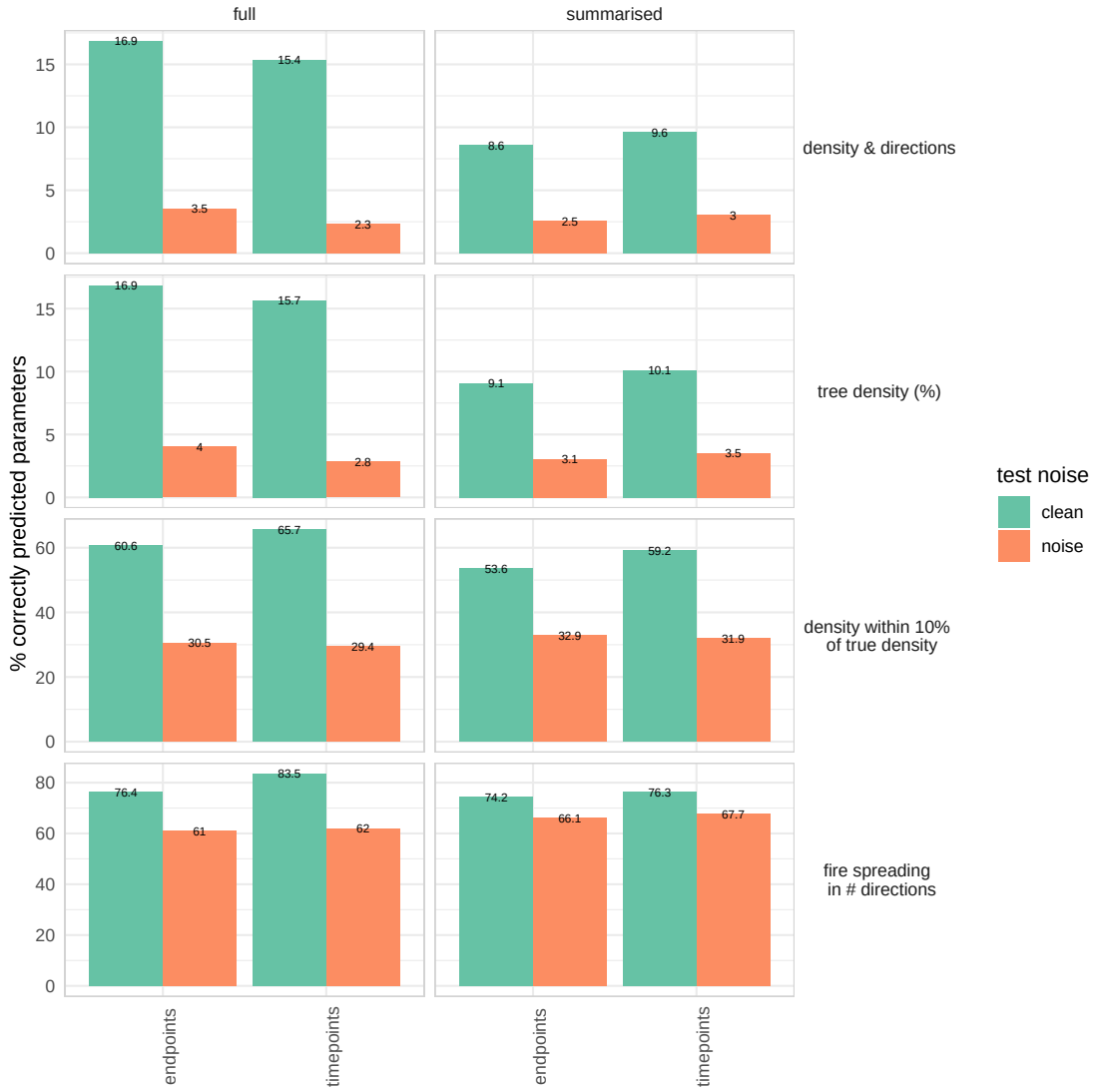


Figure A.18: Fire ABM multivariate random forest errors on both parameters: The errors as made by the multivariate random forest for the fire ABM. All results were obtained with a sample size of 198 parameter combinations (the full parameter space), each ran 5 times.

A.6 Compare algorithms within the Fire Agent-Based Model

A.6.1 Algorithms

The direction parameter is best predicted by the RF algorithm. The RF has the highest mean score in terms of Cohen's kappa, F1 score, Matthews Correlation Coefficient and the percentage of correctly predicted direction parameters (see Figure A.19). The MRF only marginally outperforms the linear regression and partial least squares algorithms, but also consistently has the largest range of performance. Meaning some of the best, but also some of the worst performances. The partial least squares algorithm yields a slightly higher mean performance than the linear regression, but its range is nearly equivalent.

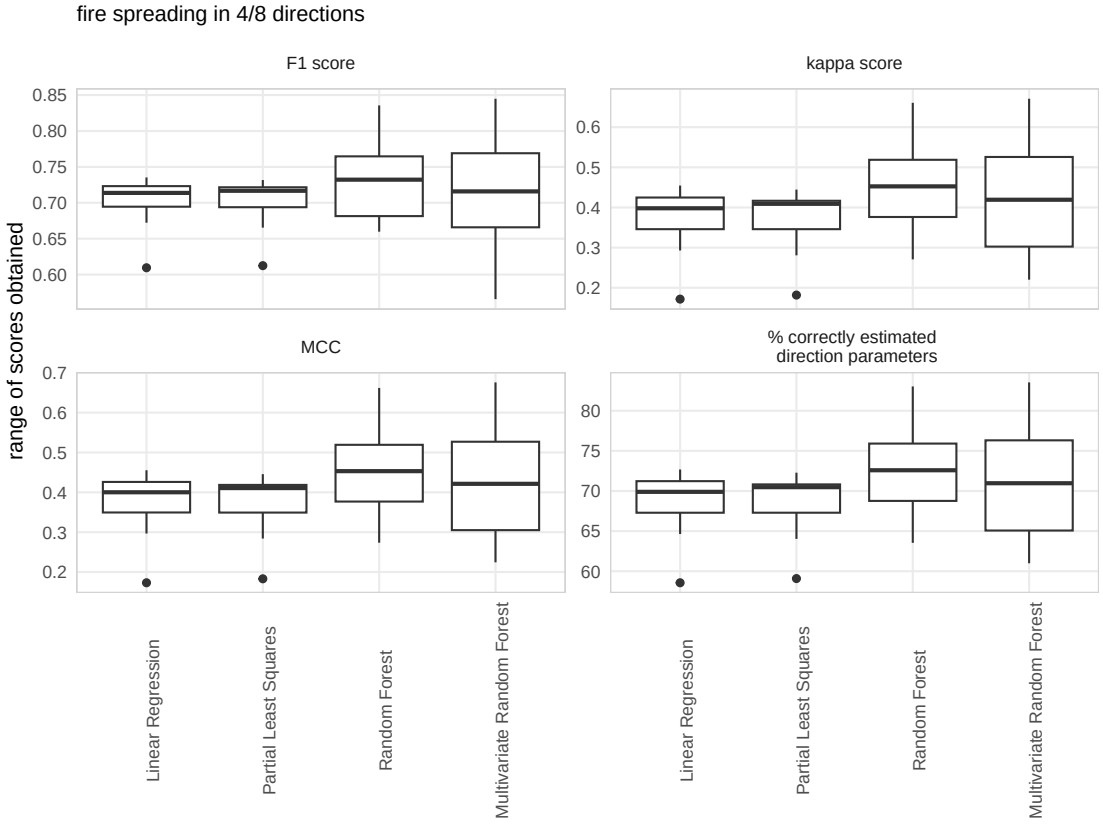


Figure A.19: Kappa scores, F1 scores, Matthew's Correlation Coefficient, and the percentage correctly predicted directions parameters for the fire ABM, per algorithm. Boxplots represent the range of results obtained with different noise levels, numbers of data points, and summarised vs. non-summarised repeated runs.

The density parameter is rarely estimated correctly, although the RF and MRF algorithms get within 10% of the correct value in around 40% of cases, sometimes up to 70% (see Figure A.20). The linear regression and partial least squares algorithms are outperformed by the RF and MRF algorithms, but the differences within those two groups are less pronounced.

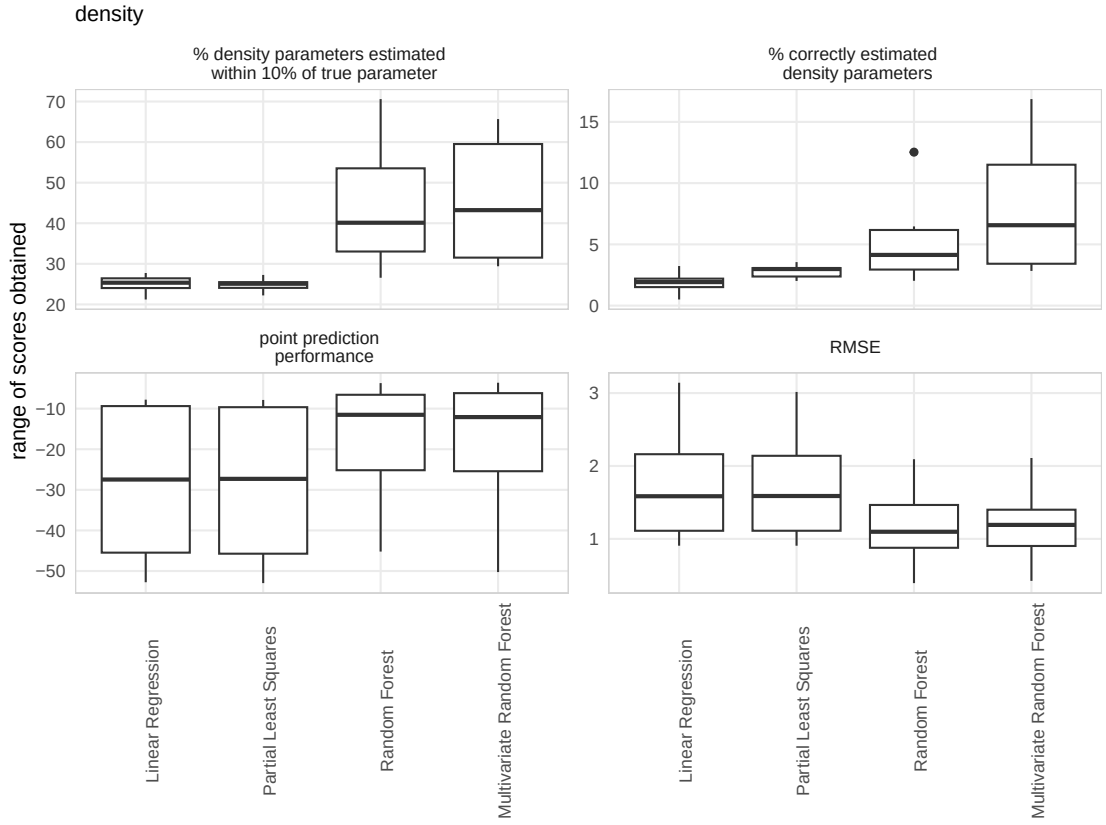


Figure A.20: Percentage of density parameters predicted within 10% of their true value, the percentage correctly predicted density parameters, point prediction performance, and the RMSE between the true and predicted density parameters for the fire ABM, per algorithm. Boxplots represent the range of results obtained with different noise levels, numbers of data points, and summarised vs. non-summarised repeated runs.

A.6.2 Data

The difference between using endpoints only, or using additional midpoints is small in terms of predicting the directions parameter (see Figure A.21). Most notably, the clean data consistently outperforms the data with noise. Although there are differences between the summarised and non-summarised data, no clear trend can be discerned.

The difference between using endpoints only, or using additional midpoints is small in terms of predicting the density parameter (see Figure A.21). Most notably, the clean data consistently outperforms the data with noise. The summarised data performs better in terms of point prediction performance, but worse in terms of RMSE. There only small differences when it comes to the percentages of (almost) correctly predicted densities, so it remains unclear which approach is favourable.

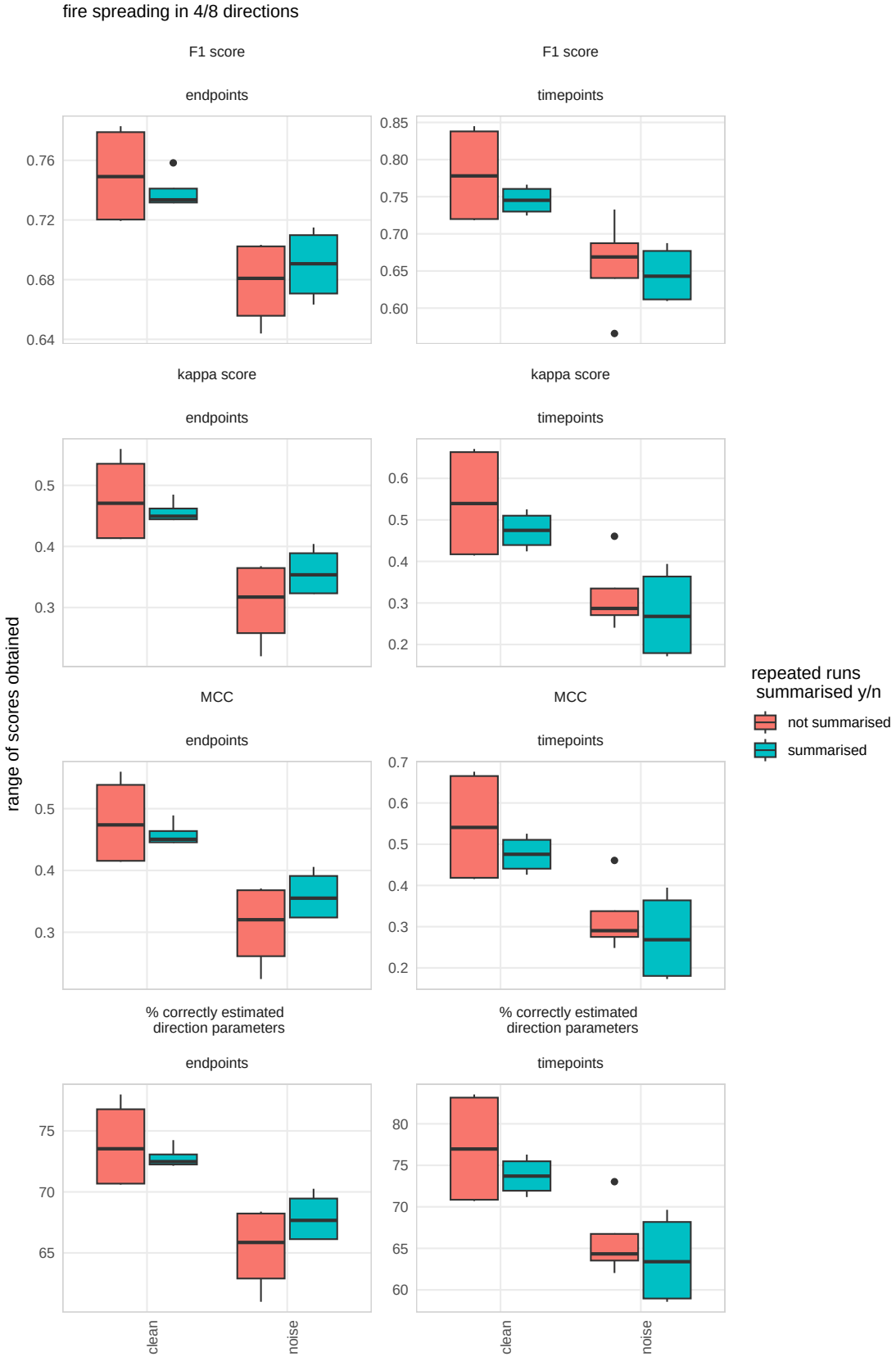


Figure A.21: Kappa scores, F1 scores, Matthew's Correlation Coefficient, and the percentage correctly predicted directions parameters for the fire ABM, per algorithm. Boxplots represent the range of results obtained with different algorithms.

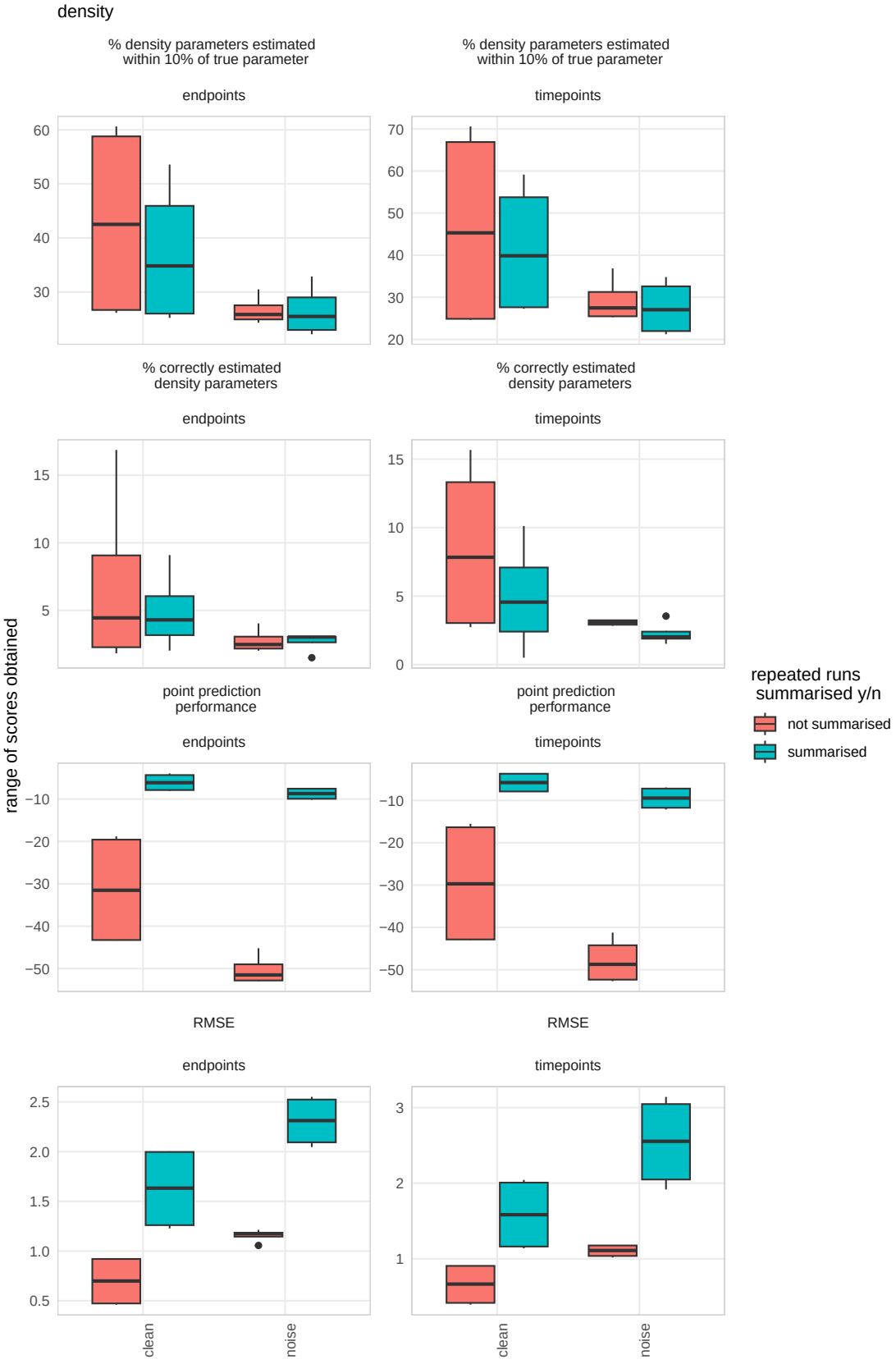


Figure A.22: Percentage of density parameters predicted within 10% of their true value, the percentage correctly predicted density parameters, point prediction performance, and the RMSE between the true and predicted density parameters for the fire ABM, per algorithm. Boxplots represent the range of results obtained with different algorithms.

Appendix B: Wolf Sheep Predation

The Wolf Sheep Predation ABM is a predator-prey model from the NetLogo Model Library [26]. The ABM comes in two versions. Here we only use the version which contains 3 entities: grass, sheep, and wolves. Wolves and sheep move randomly through the landscape, with each step costing energy. Sheep eat grass when they encounter it, to replenish their energy . After being eaten, the grass takes a set amount of time to grow back. Wolves eat sheep when they encounter them and also gain energy by doing so. Sheep and wolves die when they run out of energy. At each timestep, both wolves and sheep reproduce with a set probability. The ABM contains the following input parameters:

- initial-number-sheep: the initial number of sheep in the simulation $\{0 - 250\}$;
- initial-number-wolves: the initial number of wolves in the simulation $\{0 - 250\}$;
- sheep-gain-from-food: the amount of energy sheep gain by eating a patch of grass $\{0 - 50\}$;
- wolves-gain-from-food: the amount of energy wolves gain by eating a sheep $\{0 - 100\}$;
- sheep-reproduce: the probability that sheep reproduce at any tick $\{1 - 20\}$;
- wolves-reproduce: the probability that wolves reproduce at any tick $\{0 - 20\}$;
- grass-regrowth-time: the number of ticks it takes for grass to grow back after it has been eaten $\{0 - 100\}$.

The simulation produces 3 outputs:

- sheep: the number of sheep at each tick;
- wolves: the number of wolves at each tick;
- grass: the number of patches with grass at each tick.

The simulation runs for however many time steps are requested, with one exception: if there are no wolves left and the number of sheep exceeds 30,000 the simulation automatically stops. There are 2 other possible scenarios. When the wolf population becomes too large they eat all the sheep and thereby cause their own extinction as well. Alternatively, under a number of parameter settings, the ABM produces cyclical population dynamics.

B.1 Preliminary

B.1.1 Running the Agent-Based Model

Determining the number of runs To determine the number of runs per parameter combination we run the Wolf Sheep Predation ABM 50 times with all possible combinations of the following parameter settings

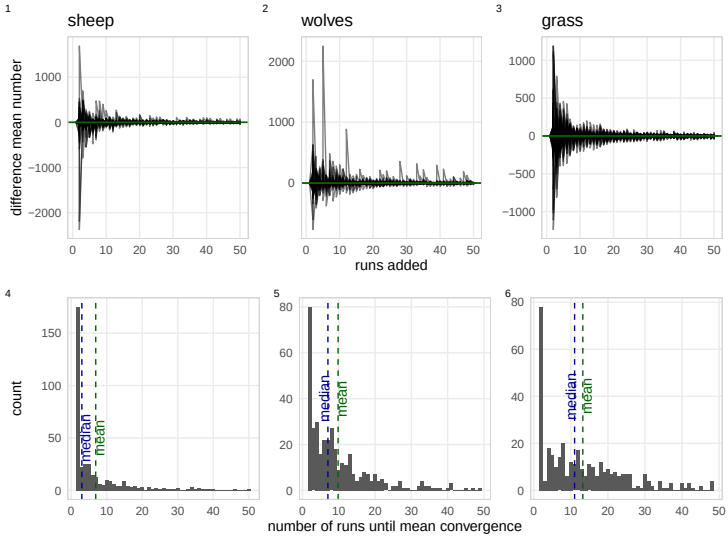
- initial-number-sheep: $\{62, 188\}$;
- initial-number-wolves: $\{62, 188\}$;
- sheep-gain-from-food: $\{12, 38\}$;
- wolves-gain-from-food: $\{25, 72\}$;
- sheep-reproduce: $\{5, 15\}$;
- wolves-reproduce: $\{5, 15\}$;
- grass-regrowth-time: $\{25, 72\}$.

This yields 128 parameter combinations to inspect. All computations are performed on the number of sheep, wolves, and grass at 50, 100, and 150 ticks. The change in mean from one simulation to the next settles below 1 within 10 runs for most parameter combinations (see Figure B.1a). There are however also simulations where it takes up to 50 runs to do so, as well as simulations where it takes more than 50 runs. The coefficient of variation settles below 0.1 within 10 runs for simulations, but for a large number it takes more than 50 runs (see Figure B.1b). Notably, there are simulations where additional runs increase the coefficient of variation when adding more runs. Finally, the relative outer variance fails to converge within 50 runs for many parameter combinations. Since it is unfeasible to run the ABM more than 50 times, we will use the outcomes of the mean convergence and coefficient of variation experiments and use 15 runs per parameter combination. This is a compromise between the simulations that require fewer than 10 runs and those that require many more.

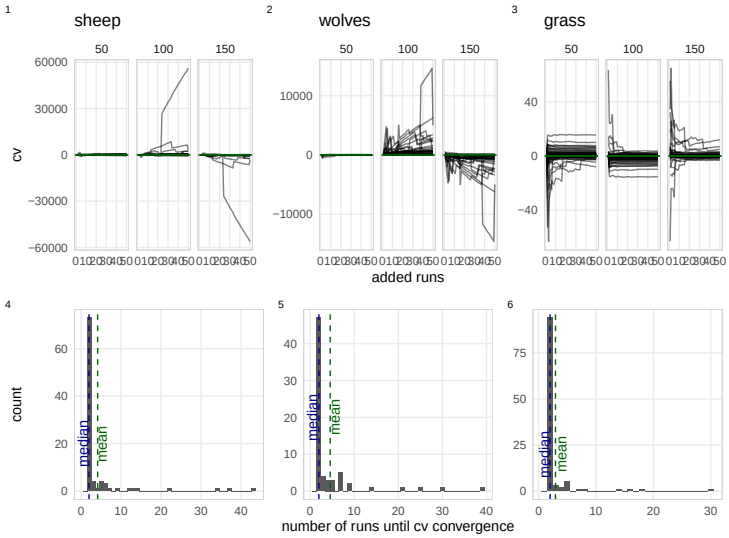
Sampling parameter space The Wolf Sheep Predation parameter space has over $1.3 * 10^{13}$ possible parameter combinations. Unfortunately, we encountered severe runtime limitations using this ABM. Therefore, we only sample 200 parameter combinations. This consitutes a meagre $1.5 * 10^{-9}\%$ of the parameter space. Each of the sampled parameter combinations is run for 500 timesteps and repeated 15 times.

Summarising repeated runs Inspecting the standard deviation of the number of sheep, wolves, and grass per parameter combination shows that for all variables the standard deviation increases as the simulation progresses (see Figures B.2, B.3, and B.4). Additionally, there are some very large standard deviations. Inspecting those further shows that for both the wolves and the grass the mean is a proper representation of the data (see Figures B.3.B and B.4.B). However, for the amount of sheep taking the mean appears to be a poor representation of the data in at least 4 of the 5 most extreme cases (see Figure B.2.B). We summarise using the mean.

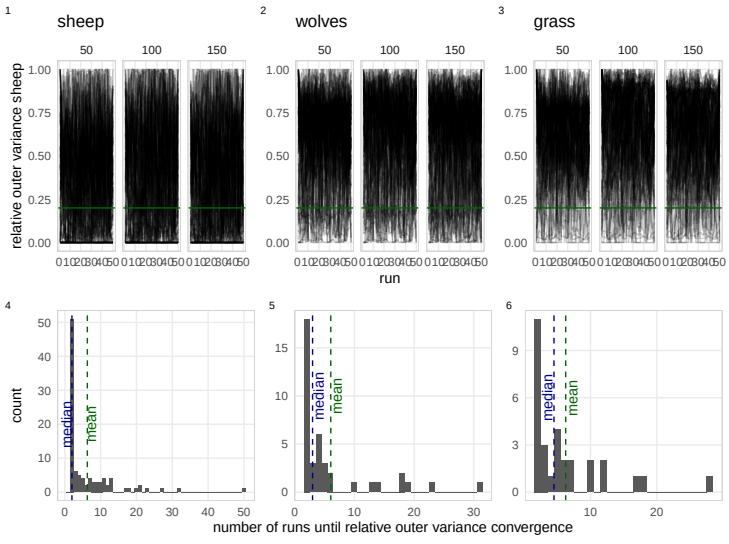
Creating training data When adding temporal noise, the full time series was skipped since adding temporal noise to it would make the length of the time series unequal.



(a) Mean convergence



(b) Coefficient of variation convergence



(c) Windowed variance convergence

Figure B.1: Determining number of model runs for wolf sheep ABM: Figure shows the mean convergence (A), coefficient of variation (B), and relative outer variance (C) that inform the number of repetitions per parameter combination.

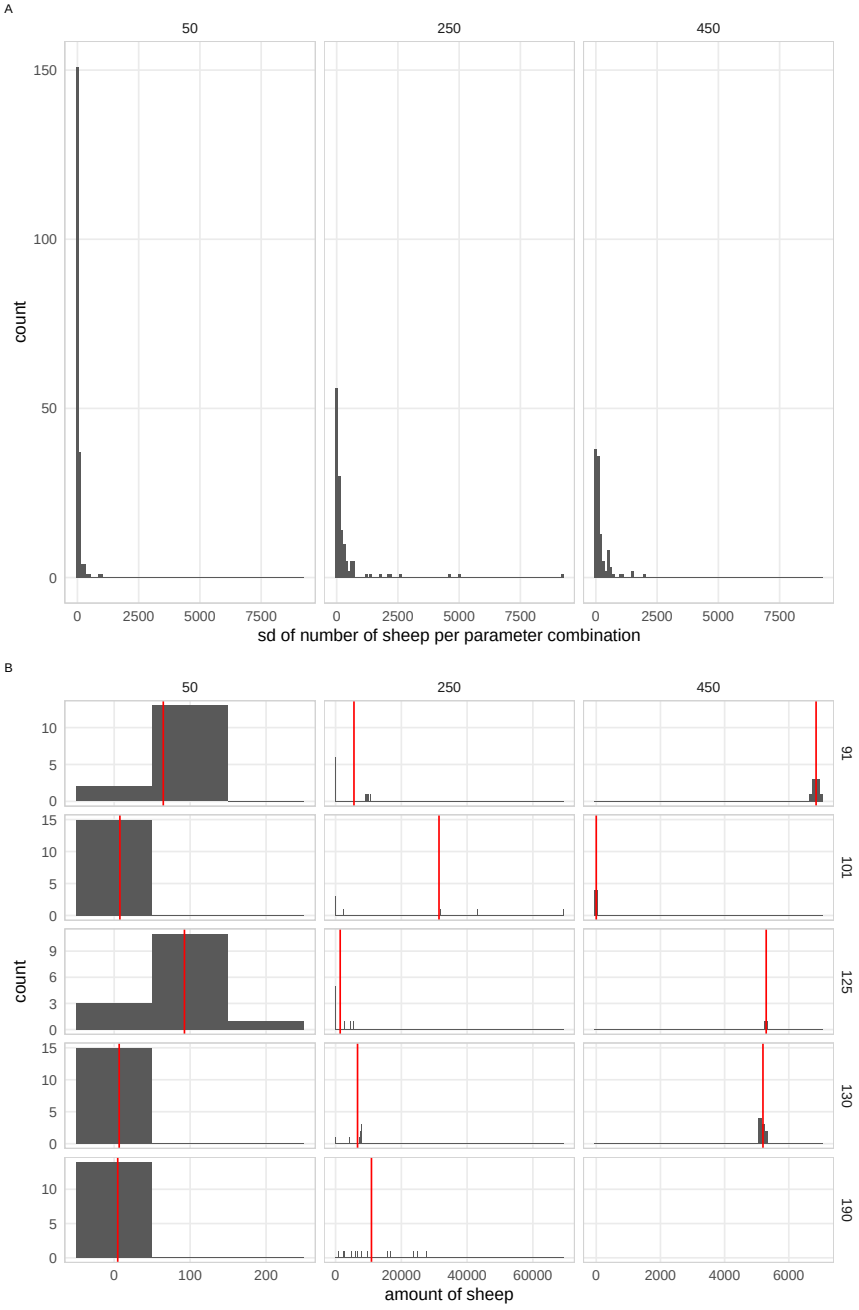


Figure B.2: Inspecting summarisating options for Wolf Sheep ABM: The figure shows the standard deviations of the number of sheep within each parameter combination for time steps 50, 250, and 450 (A). Additionally, the parameter combinations with the 5 largest standard deviations are shown separately (B).

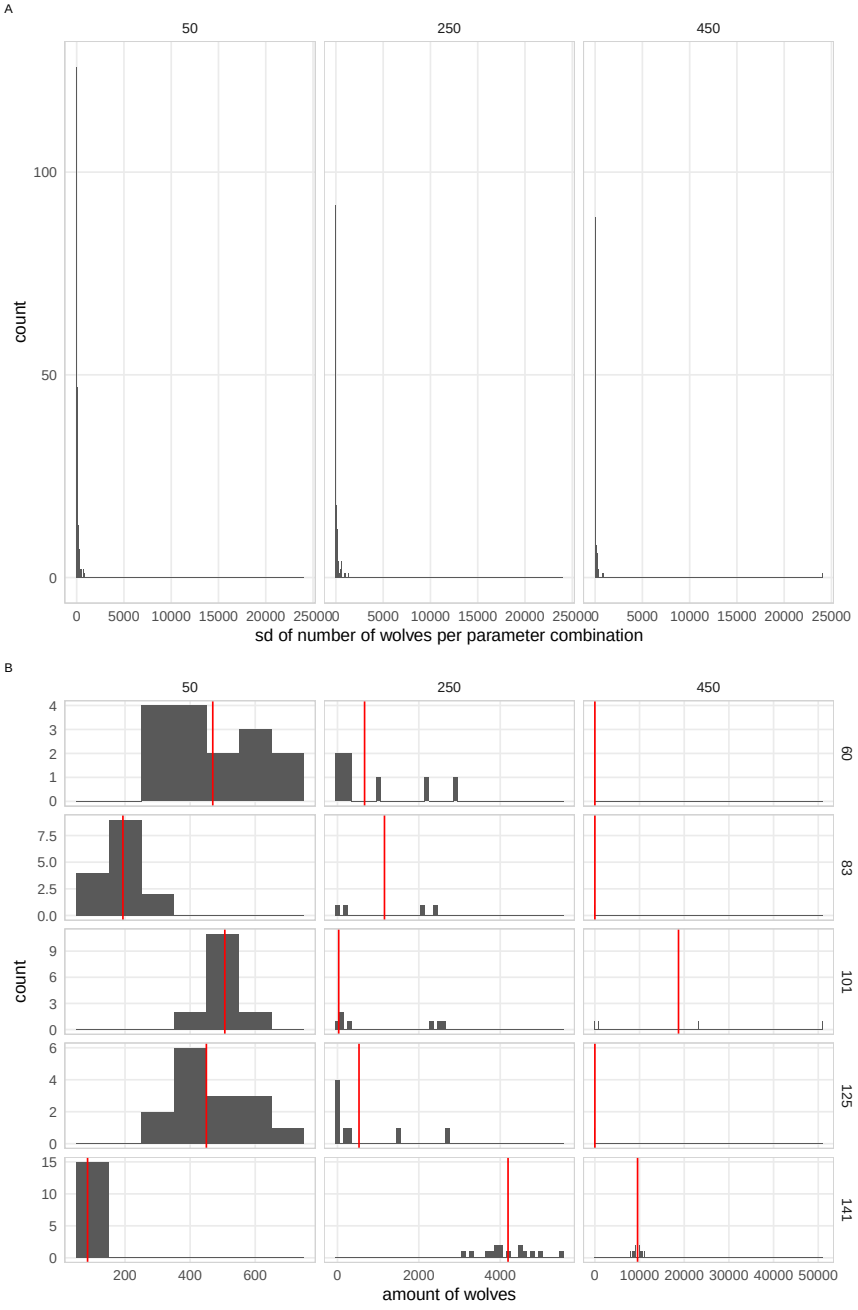


Figure B.3: Inspecting summarising options for Wolf Sheep ABM: The figure shows the standard deviations of the number of wolves within each parameter combination for time steps 50, 250, and 450 (A). Additionally, the parameter combinations with the 5 largest standard deviations are shown separately (B).

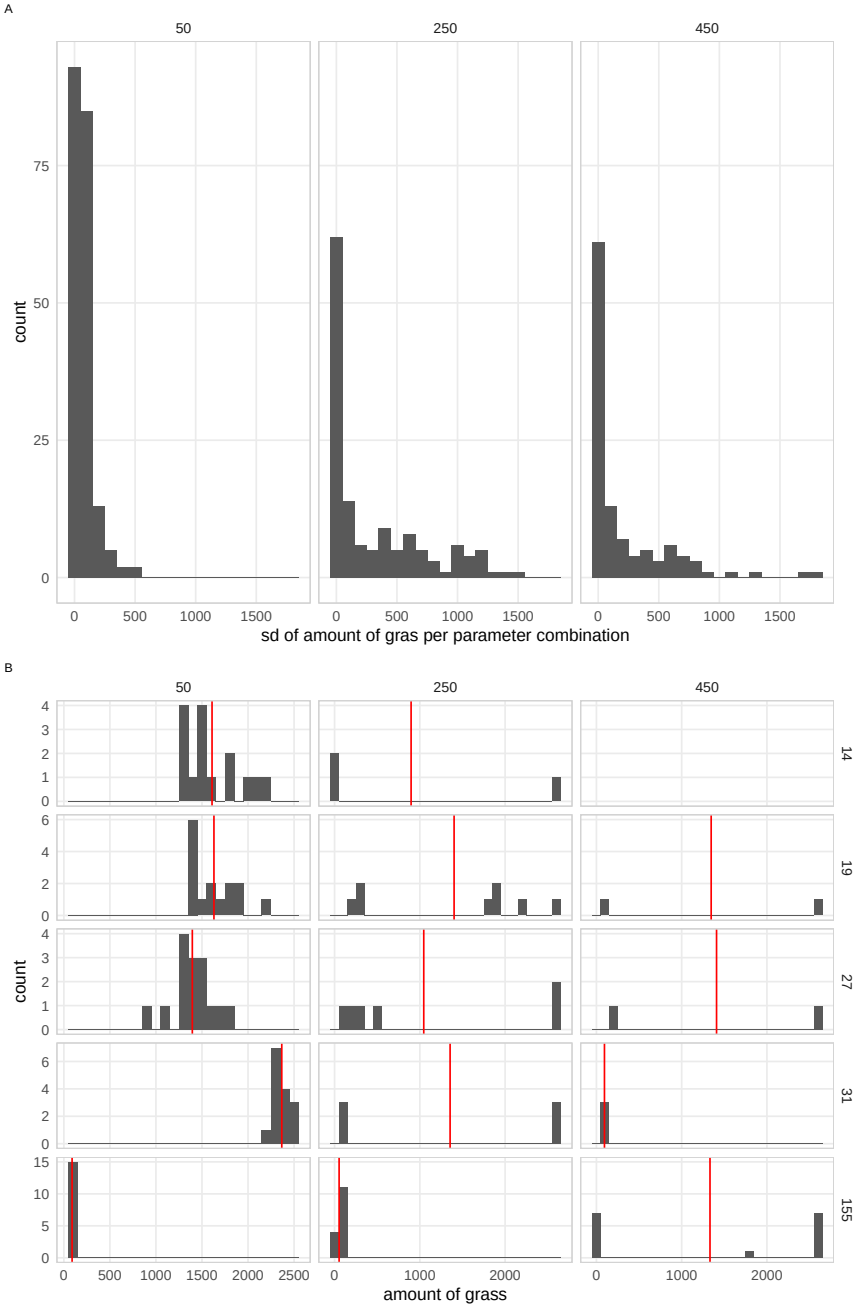


Figure B.4: Inspecting summarising options for Wolf Sheep ABM: The figure shows the standard deviations of the amount of grass within each parameter combination for time steps 50, 250, and 450 (A). Additionally, the parameter combinations with the 5 largest standard deviations are shown separately (B).

B.1.2 Agent-Based Model inspection

Visualising parameter space Because an image with all time series in the dataset of 200 samples quickly becomes illegible, Figure B.5 shows a selection of parameter sets that together form a representation of the possible behaviours of the ABM.

Identifying equifinality 5-means clustering based on the number of sheep, wolves, and grass at ticks 50, 100, 250, 450, and 500 are shown in Figure B.13. We see that with the exception of the reproductive probability of wolves (B.13.F), all clusters show large overlap in the input parameters that produced the outputs. This points towards equifinality for almost all input parameters.

Finding unidentifiable parameters None of the slopes relating input parameters to output parameters are completely flat, indicating that all input parameters have at least some influence on the output of the ABM (see Figure B.14). However, many of the parameters show similar relationships, indicating that it may be difficult to distinguish the effect of one parameter from another.

Temporal auto-correlation Figures B.15 A, C, and E show that there is considerable auto-correlation for most time series produced by the Wolf Sheep ABM. However, most of this disappears when looking at the partial auto-correlation (see Figures B.15B, D, and F). Since auto-correlation is present, algorithms taking this feature into account may be expected to perform better when calibrating the Wolf Sheep ABM.

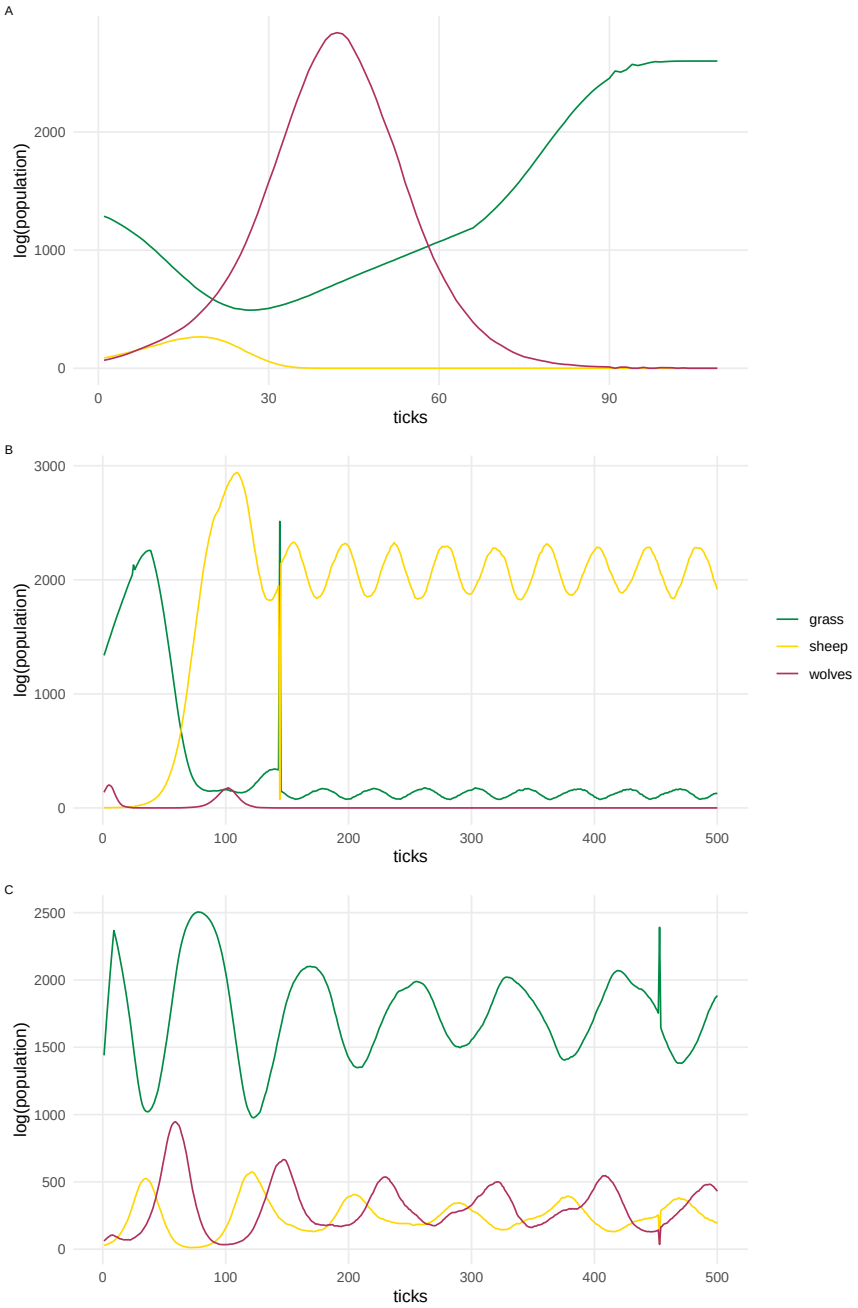


Figure B.5: Time series for the Wolf Sheep ABM: A representative sample of behaviours produced by the ABM. Plots show outputs for the following input parameter values: initial number of sheep (A:80, B:2, C:28), initial number of wolves (A:59, B:118, C:54), sheep gain from food (A:44, B:31, C:3), wolves gain from food (A:89, B:16, C:24), sheep reproduce (A:14, B:11, C:17), wolves reproduce (A:16, B:19, C:15), grass regrowth time (A:66, B:39, C:9). A) Both wolves and sheep go extinct, the grass reaches carrying capacity. B) Wolves go extinct and sheep and grass show cyclical behaviour. C) All three entities coexist, exhibiting cyclical behaviour.

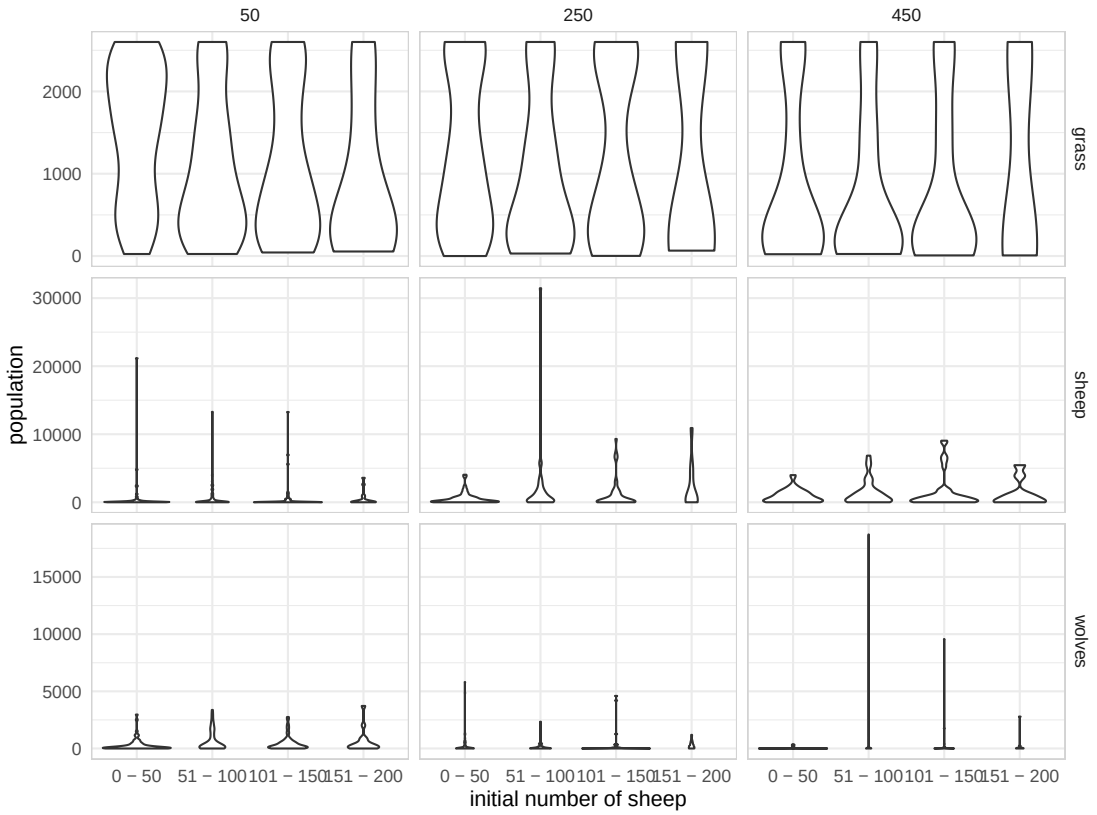


Figure B.6: Influence of initial number of sheep on sheep, wolves, and grass: Violin plots showing the distribution of sheep, wolves, and grass at ticks 50, 250, and 450 as a function of the initial number of sheep in the simulation.

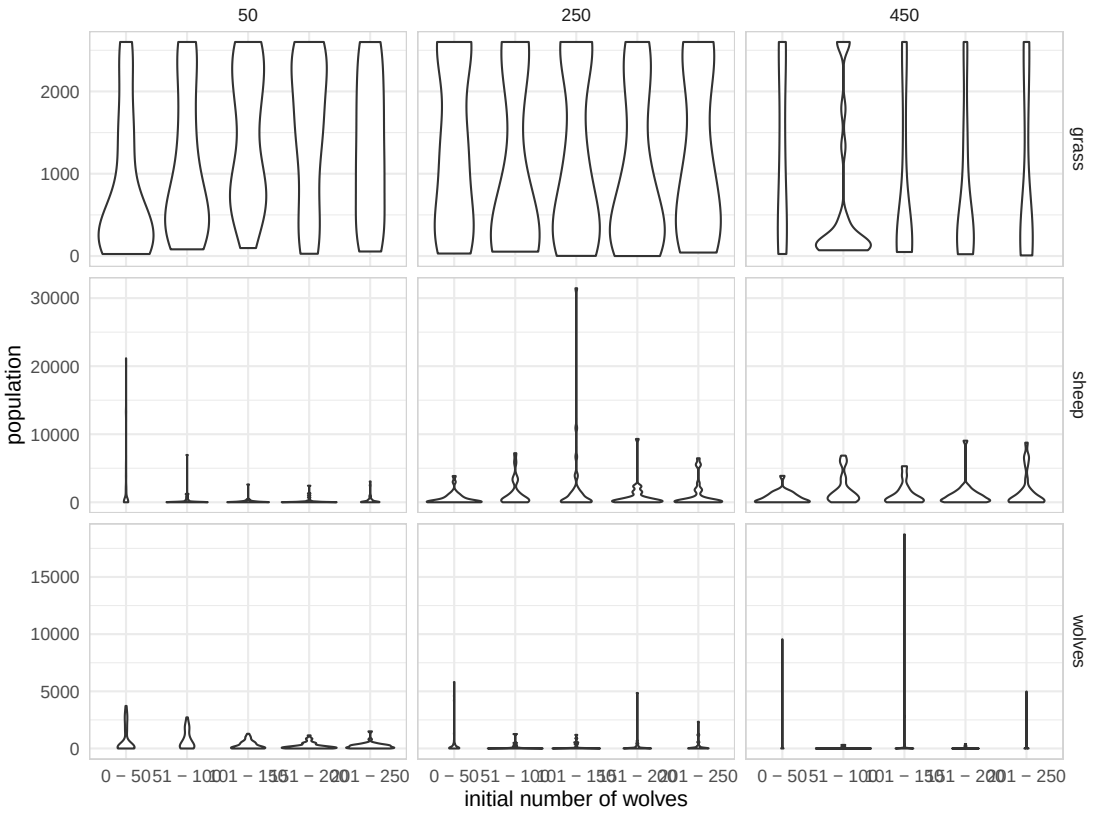


Figure B.7: Influence of initial number of wolves on sheep, wolves, and grass: Violin plots showing the distribution of sheep, wolves, and grass at ticks 50, 250, and 450 as a function of the initial number of wolves in the simulation.

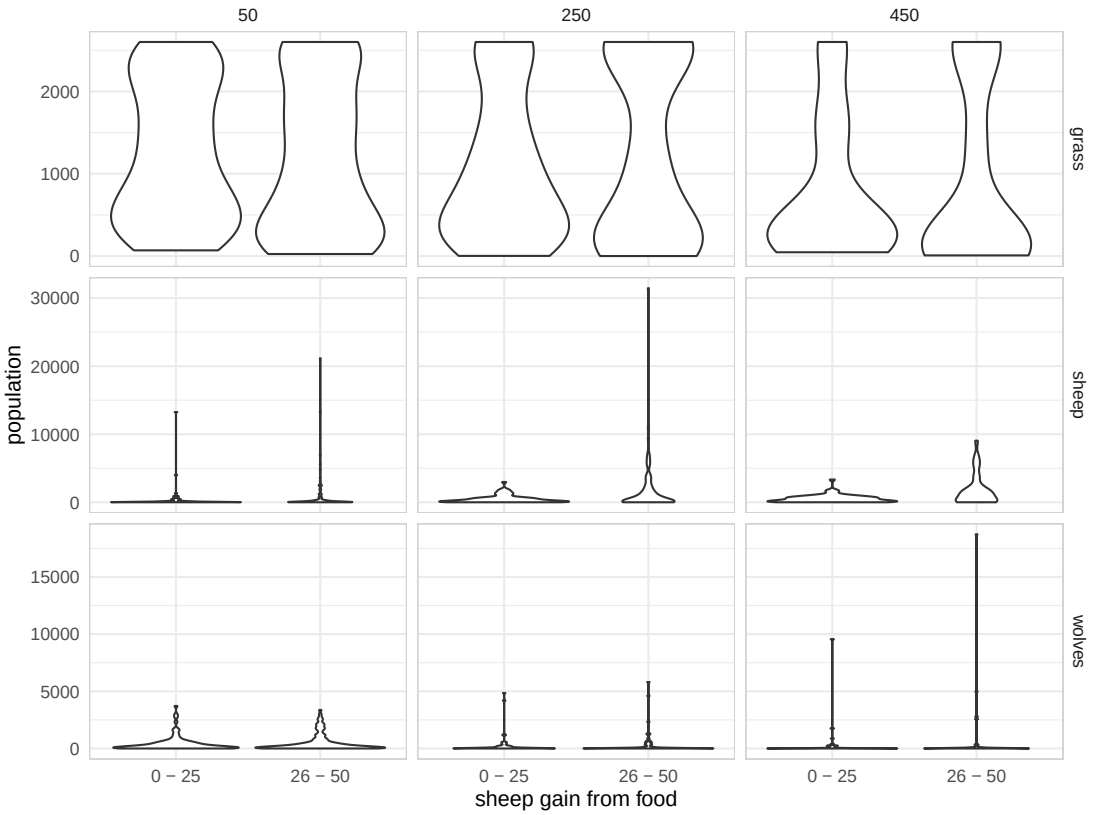


Figure B.8: Influence of sheep gain from food on sheep, wolves, and grass: Violin plots showing the distribution of sheep, wolves, and grass at ticks 50, 250, and 450 as a function of the amount of energy gained by sheep when eating grass in the simulation.

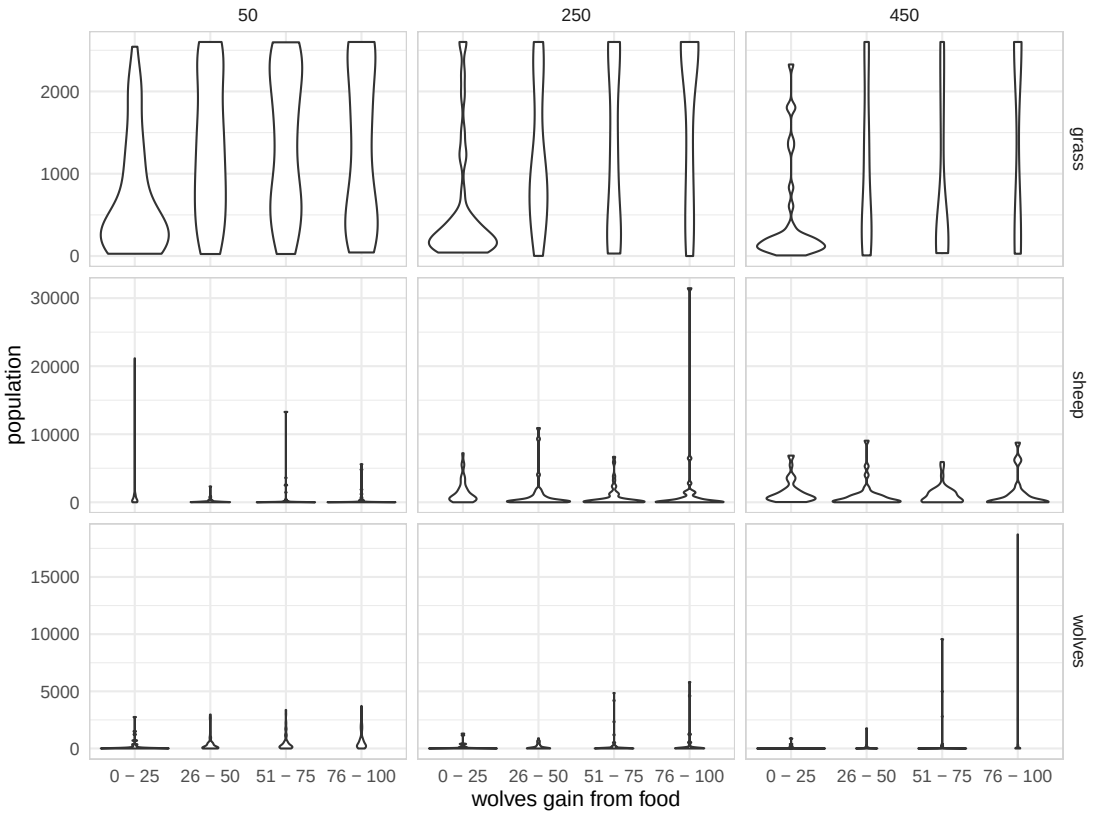


Figure B.9: Influence of wolves gain from food on sheep, wolves, and grass: Violin plots showing the distribution of sheep, wolves, and grass at ticks 50, 250, and 450 as a function of the amount of energy gained by wolves when eating sheep in the simulation.

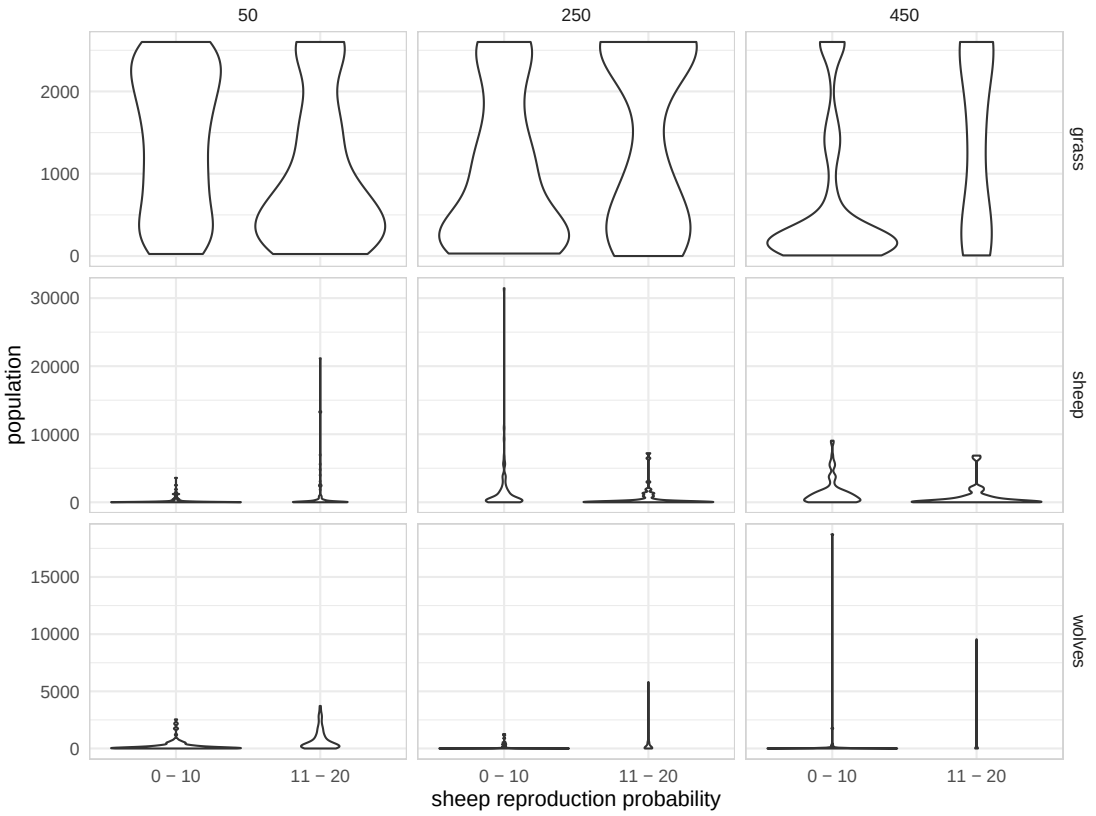


Figure B.10: Influence of probability of reproduction of sheep on sheep, wolves, and grass: Violin plots showing the distribution of sheep, wolves, and grass at ticks 50, 250, and 450 as a function of the probability for sheep to reproduce in the simulation.

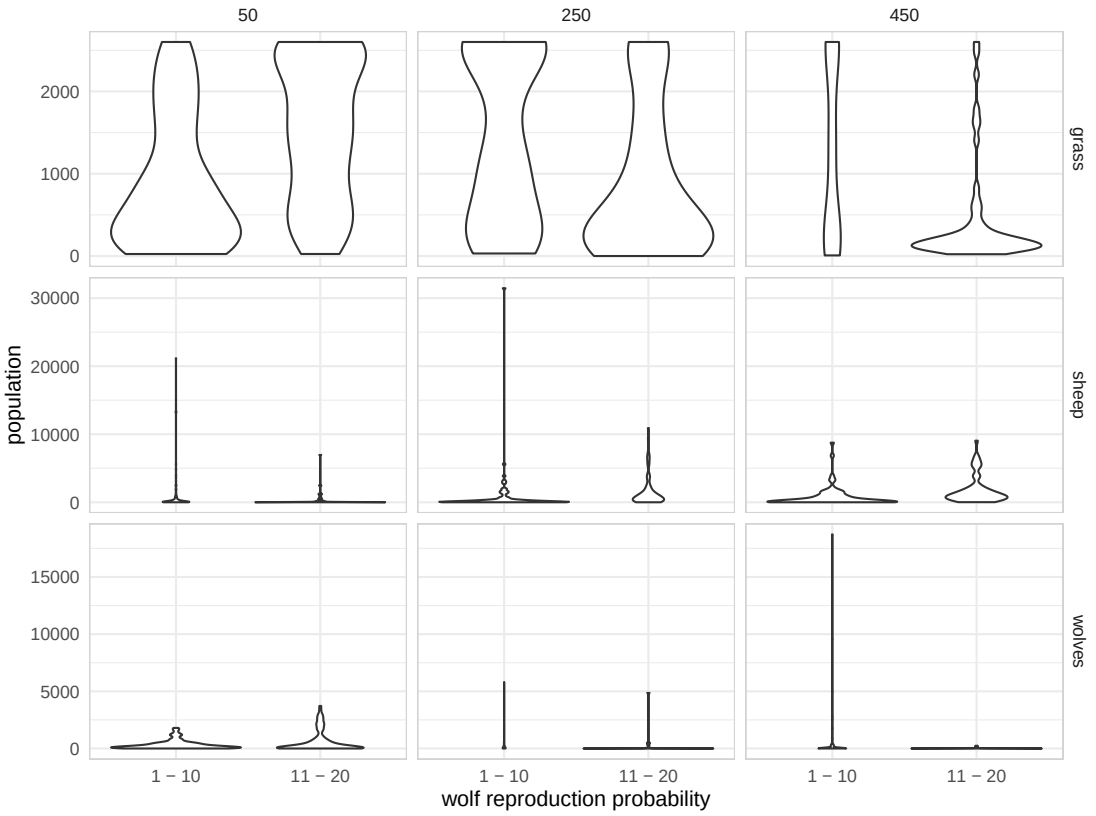


Figure B.11: Influence of probability of reproduction of wolves on sheep, wolves, and grass: Violin plots showing the distribution of sheep, wolves, and grass at ticks 50, 250, and 450 as a function of the probability for wolves to reproduce in the simulation.

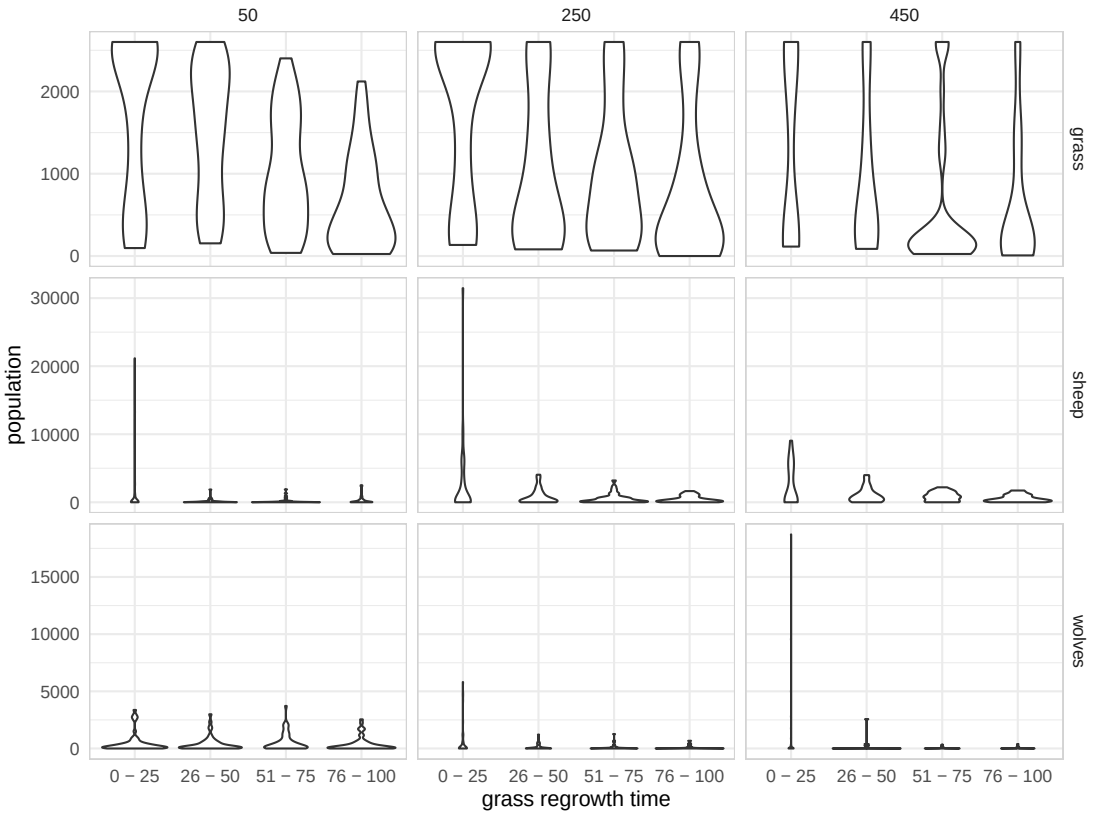


Figure B.12: Influence of grass regrowth time on sheep, wolves, and grass: Violin plots showing the distribution of sheep, wolves, and grass at ticks 50, 250, and 450 as a function of the grass regrowth time in the simulation.

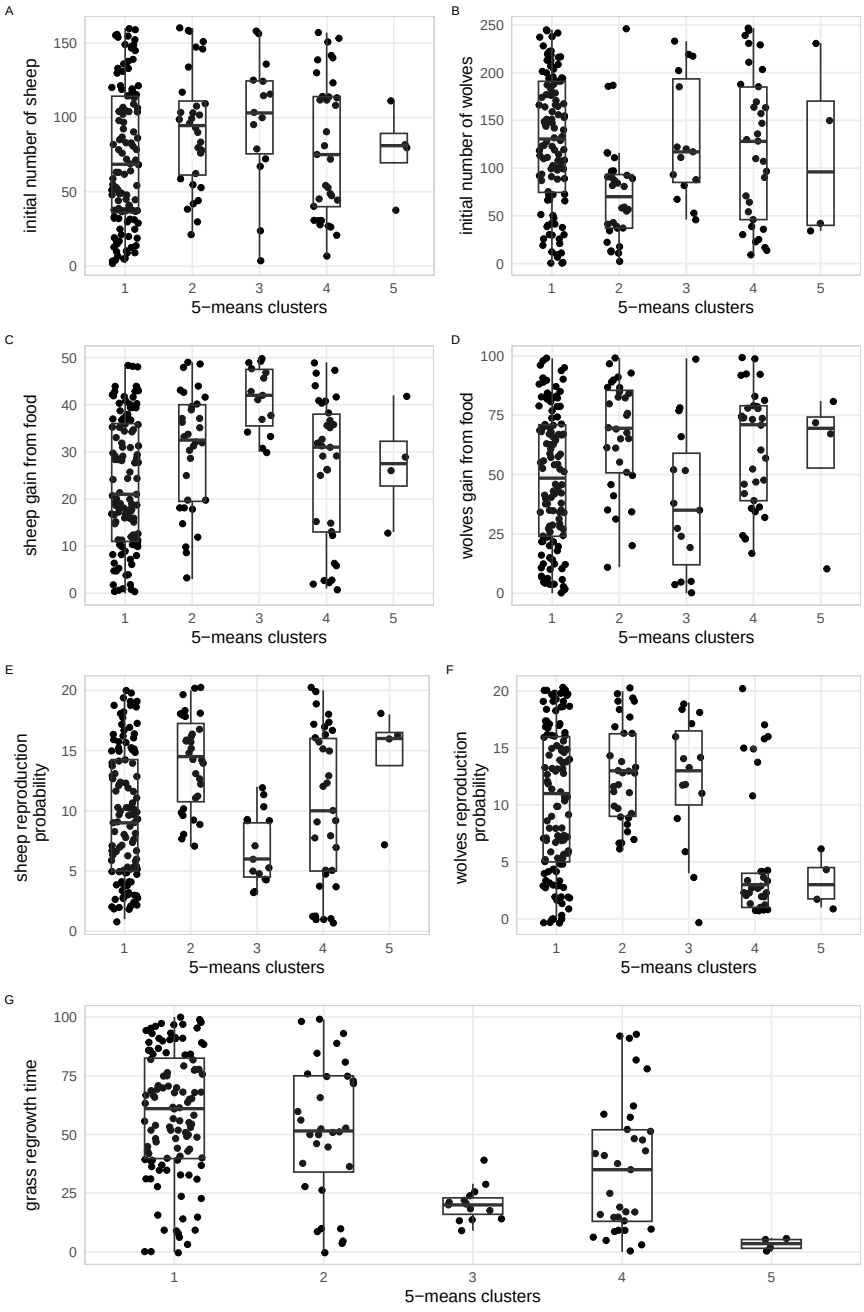


Figure B.13: K-means clustering of Wolf Sheep outputs: The results of a 5-means clustering technique on the sheep, wolf, and grass numbers at ticks 50, 100, 250, 450, and 500. Each panel shows the distribution of an input parameter within the clusters.

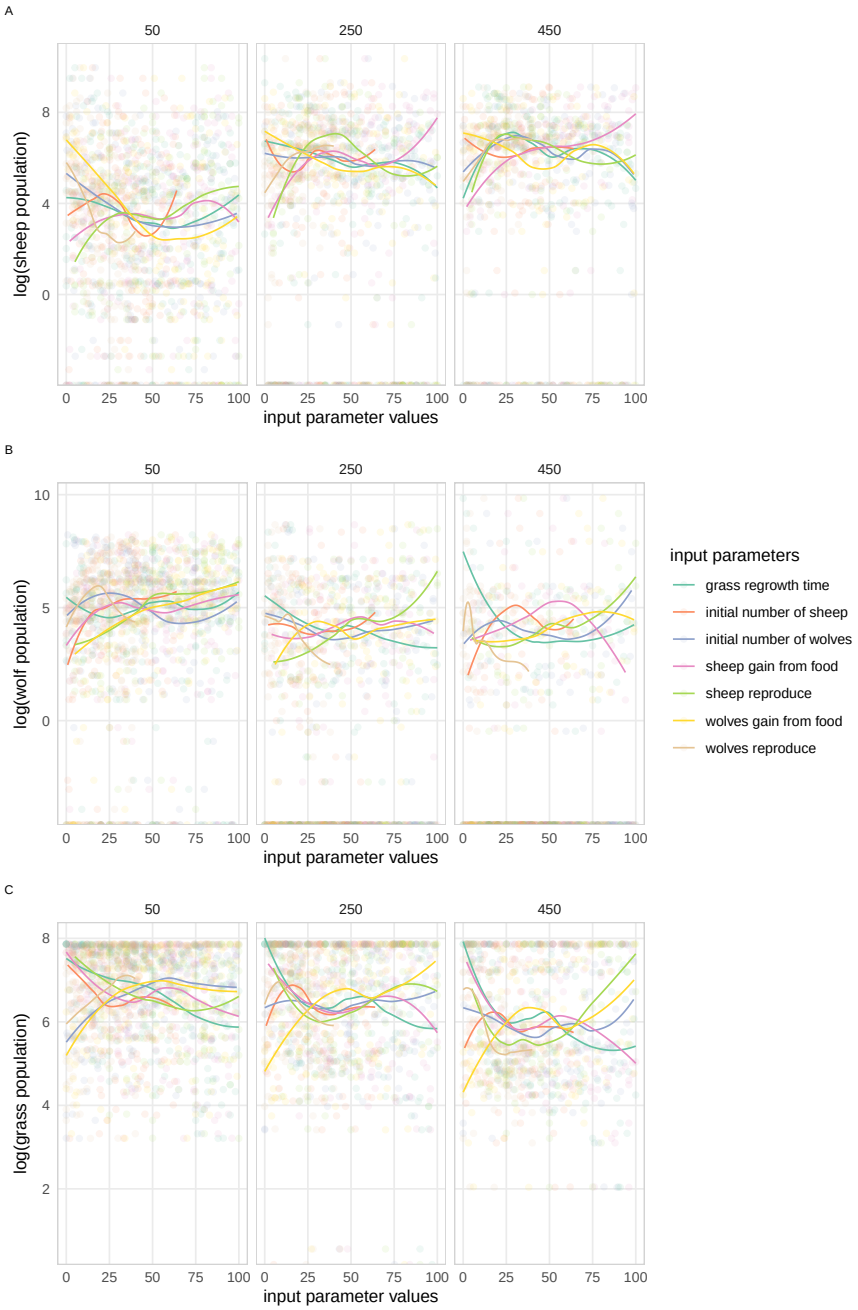


Figure B.14: Identifying unidentifiable parameters: LOESS-smoothed relationships between input parameter values and output population values (A: sheep, B: wolves, C: grass) at ticks 50, 250, and 450.

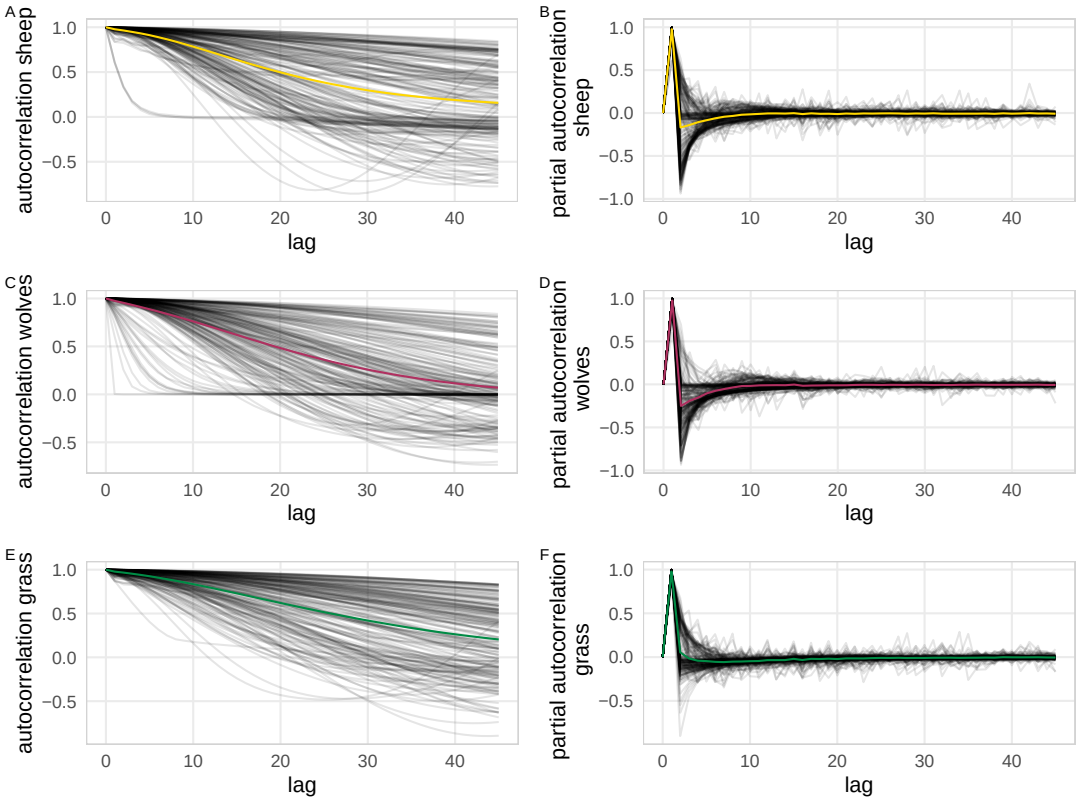


Figure B.15: (Partial) Auto-correlation in the Wolf Sheep ABM: (partial) auto-correlations are shown for the sheep (A), wolf (B), and grass (C) output variables. Means are plotted in colour.

B.2 Linear regression

In order to be able to fit the regression a number of time series had to be shortened to prevent rank-deficient fits. The summarised data with a time step every 10 ticks had to be shortened to ticks 0 to 400, the non-summarised full time series data had to be shortened to ticks 0 to 409, and the summarised full time series data had to be shortened to ticks 0 to 29.

In all cases, 0% of the input parameters are predicted correctly.

When looking at the root mean squared error for the 7 predicted input parameters (see Figure B.16), we can draw a number of general conclusions. First, the difference between summarised vs. non-summarised data is larger than the difference between noise vs. no noise and the difference between different numbers of time points in the data. The summarised data performs noticeably worse than the non-summarised data. In the non-summarised data, there is very little difference between the different noise levels or number of data points. This difference is larger within the summarised data. An exception is the full time series, which performs much worse when noise is added. Notably, for the sheep and wolf reproductive probabilities and the wolves gain from food parameter, within the non-summarised data, the full time series data are outperformed by data with fewer time steps. This may be due to the fact that a large part of the full time series had to be removed in order to be able to fit the regressions. This may have resulted in there being fewer information in the full time series data than we would have otherwise expected. This may also be the reason for the small observed differences in performance between the full time series and the data with fewer time points.

The point prediction performance (see Figure B.17) indicates mis-identification for most cases with the summarised data, and unidentified parameters in most cases with the non-summarised data. Noticeably, there appears to be some improvement when going from 10 to 50 time steps in the summarised data, although all time steps data are outperformed by the endpoint only data.

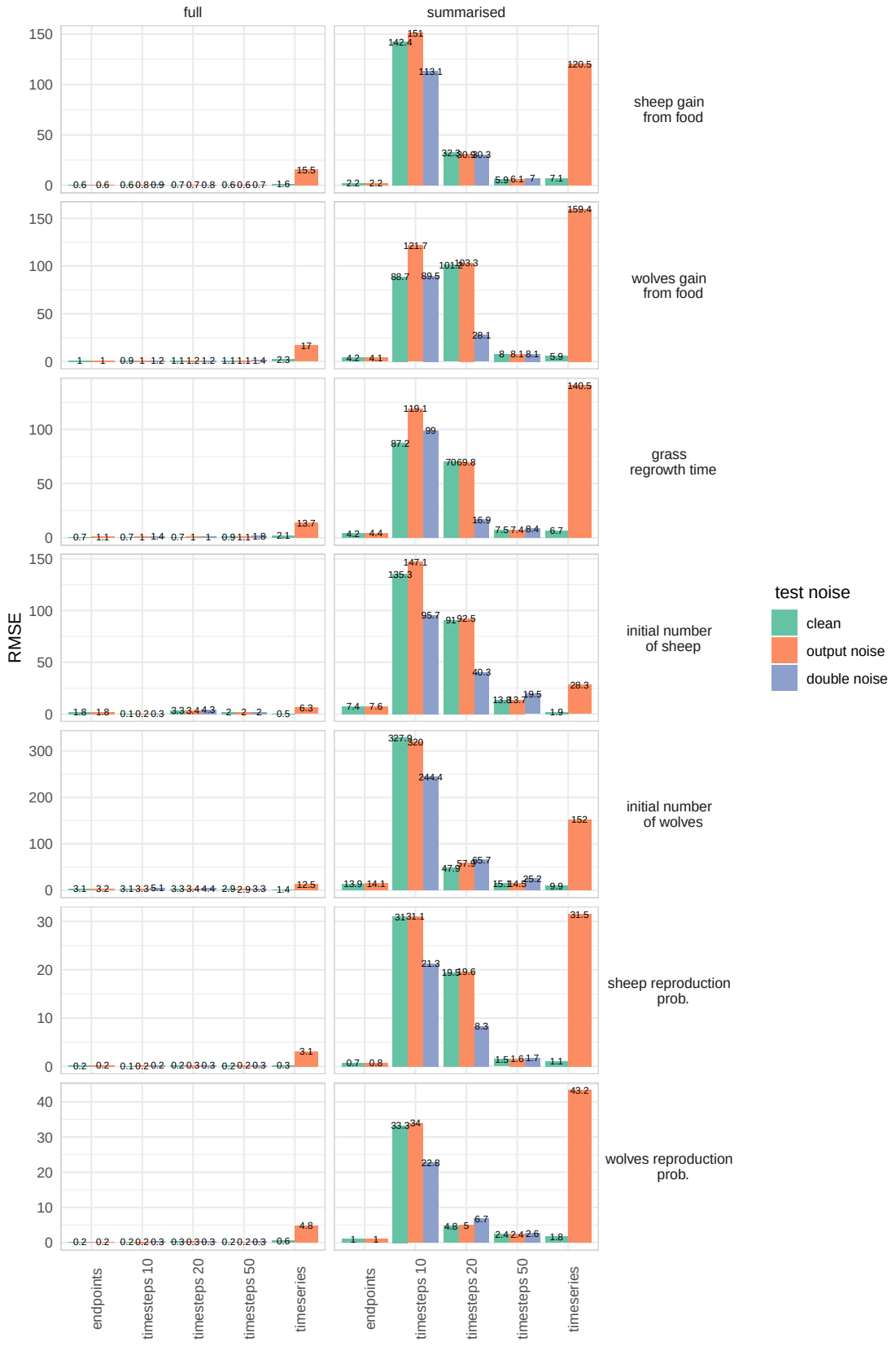


Figure B.16: Root mean squared error of the predicted vs. the true input parameters for the Wolf Sheep Predation ABM, using a simple linear regression.

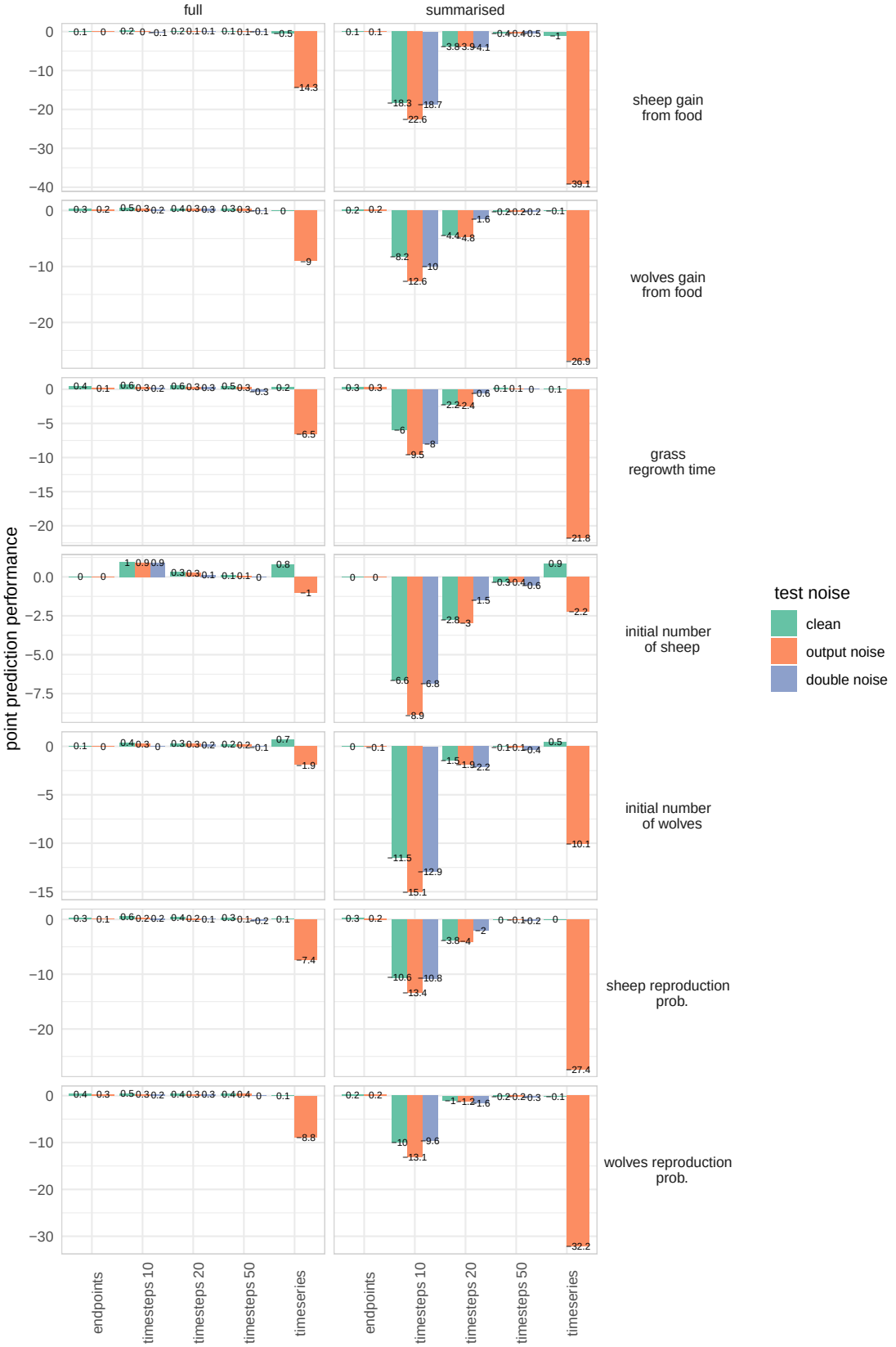


Figure B.17: Point prediction performance of the predicted input parameters for the Wolf Sheep Predation ABM, using a simple linear regression.

B.3 Partial Least Squares

In order to be able to fit the partial least squares a number of time series had to be shortened to prevent excessive run times (i.e. well over 24 hours for a single data set). The non-summarised data with a time step every 10 ticks had to be shortened to ticks 0 to 200 and the non-summarised data with a time step every 20 ticks had to be shortened to ticks 0 to 250. The non-summarised full time series data was completely excluded. Additionally, the summarised full time series data had to be run with 100 cross-validation components instead of 144.

In all cases, 0% of the input parameters are predicted correctly.

In all cases, the non-summarised data outperforms the summarised data in terms of root mean squared error between the predicted and true input parameters. Within the non-summarised data, the differences between the datasets with different numbers of time steps are small. The data without added noise consistently outperforms the data with added noise, although the differences are in most cases very small. Within the summarised data, the data with 10 and 20 time steps are clearly outperformed by the data with endpoints only, 50 time steps, and the full time series. The data with both output and temporal noise in some cases outperforms the clean data. Overall, the data with only endpoints performs best, followed closely by the full time series data.

Where the point prediction performance for the summarised data in almost all cases indicates mis- or unidentified parameters, the full data fares noticeably better. There is even a score of 1, indicating that the initial number of sheep was perfectly identified using the data with 10 time steps. In the non-summarised data we also see a clear trend where mis-identification increases with increasing noise levels. Except for the initial number of sheep input parameter, there is no clear distinction between the number of time points in the data.

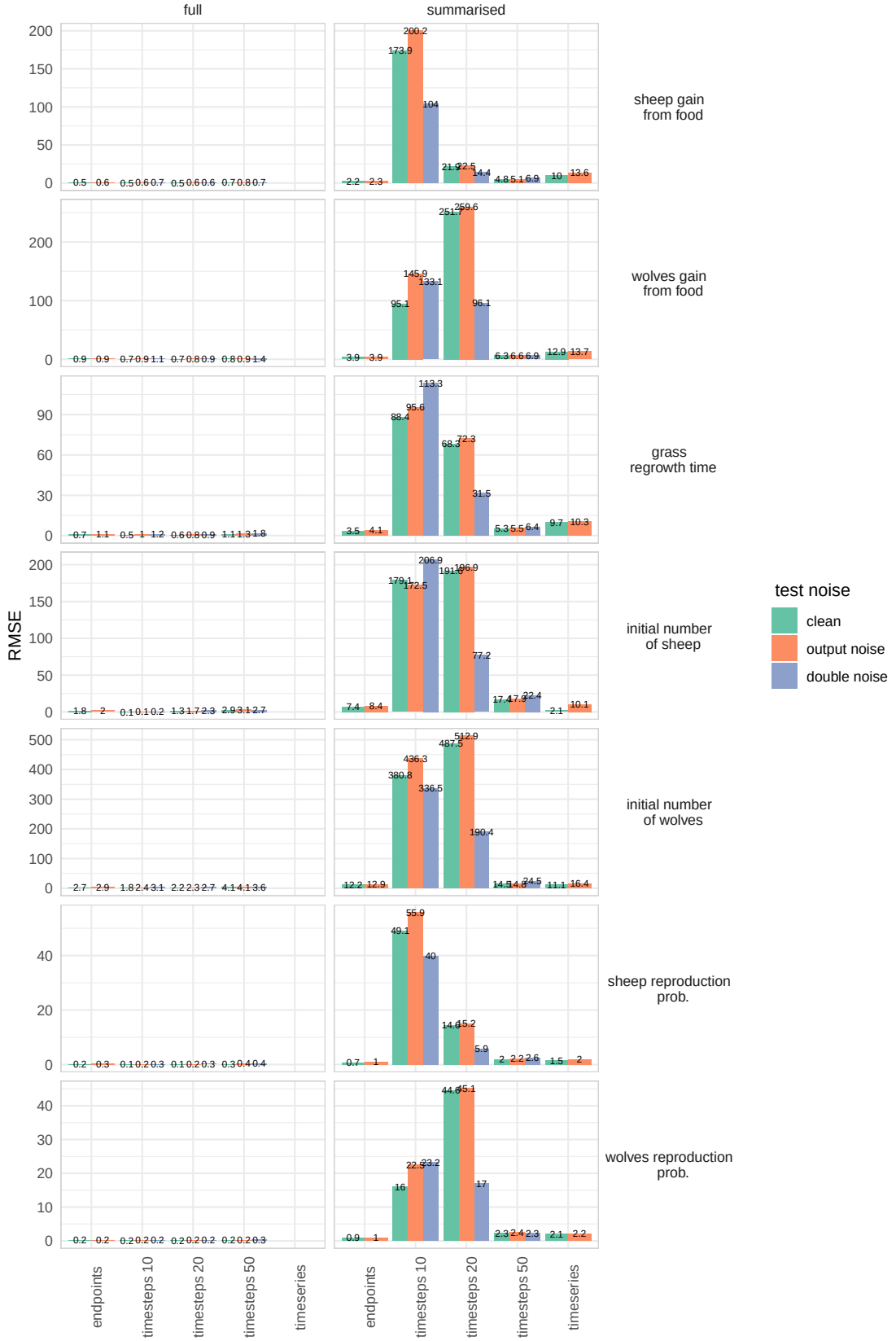


Figure B.18: Root mean squared error of the predicted vs. the true input parameters for the Wolf Sheep Predation ABM, using PLS.

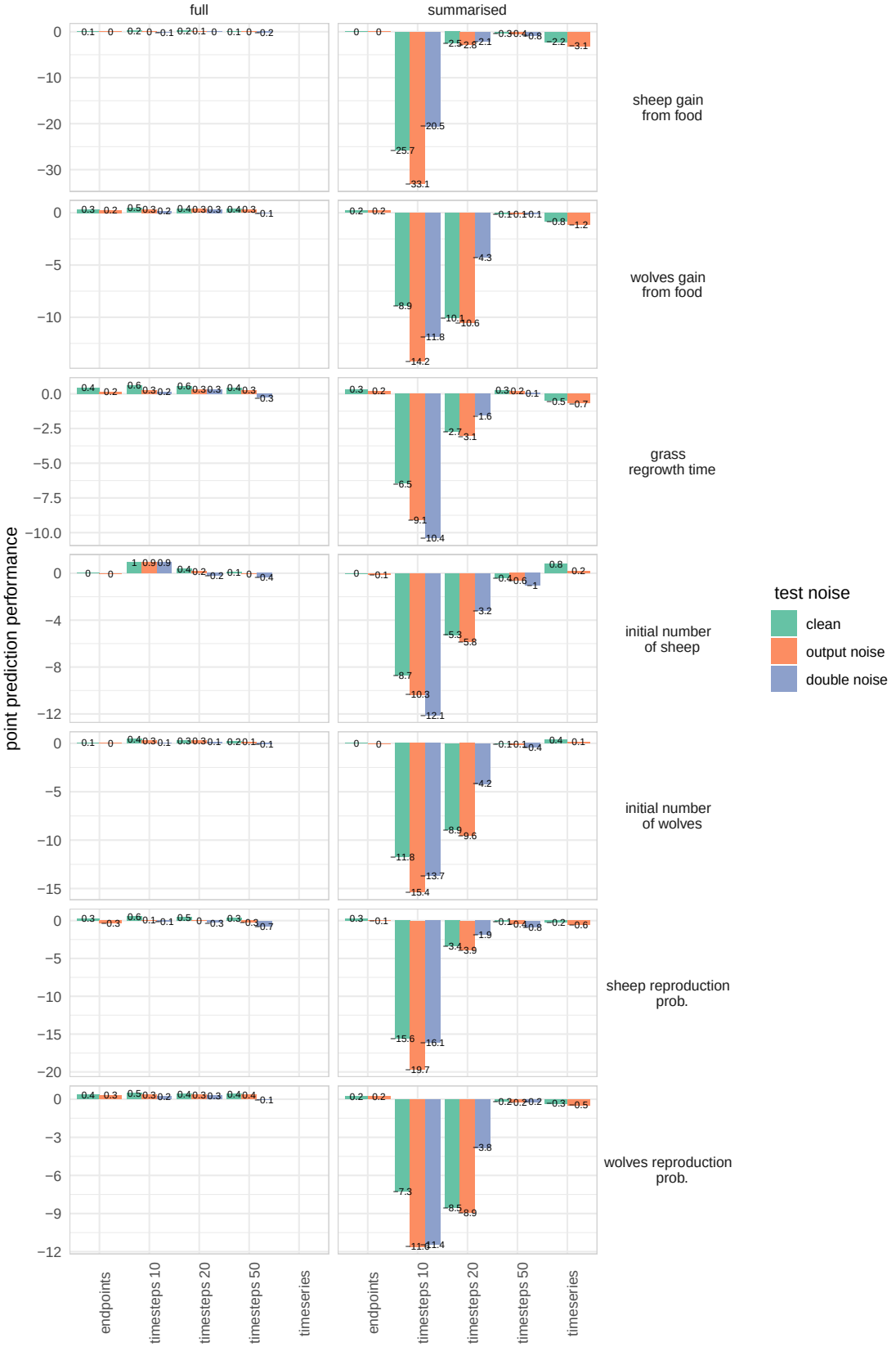


Figure B.19: Point prediction performance of the predicted input parameters for the Wolf Sheep Predation ABM, using PLS.

B.4 Random Forest

The summarised data does not predict any parameters correctly (see Figure B.20). The full data, however, does predict a very small percentage of some of the parameters correctly. There are no instances where all parameters are predicted correctly (see top left panel of Figure B.20). Note also that any instances where parameters are predicted correctly, are those without noise added to the data. For the wolf and sheep reproduction and the grass regrowth parameters, all numbers of time points predict some parameters correctly.

In terms of RMSE, the full data clearly outperforms the summarised data. Additionally, within the full data the performance consistently decreases with increasing noise levels. This trend is also present within the summarised data, but less consistently so. We can, however, see that the clean data always outperforms the data with added noise. Within the summarised data, the full time series has a clear advantage over the data with fewer time points for the initial number of sheep and initial number of wolves parameters. This trend is less obvious for the other parameters (see Figure B.21)

The full data without noise predicts parameter values accurately, according to the point prediction performance (see Figure B.22). The performance decreases with increased noise levels and when switching to summarised data, but rarely indicates mis-identification.

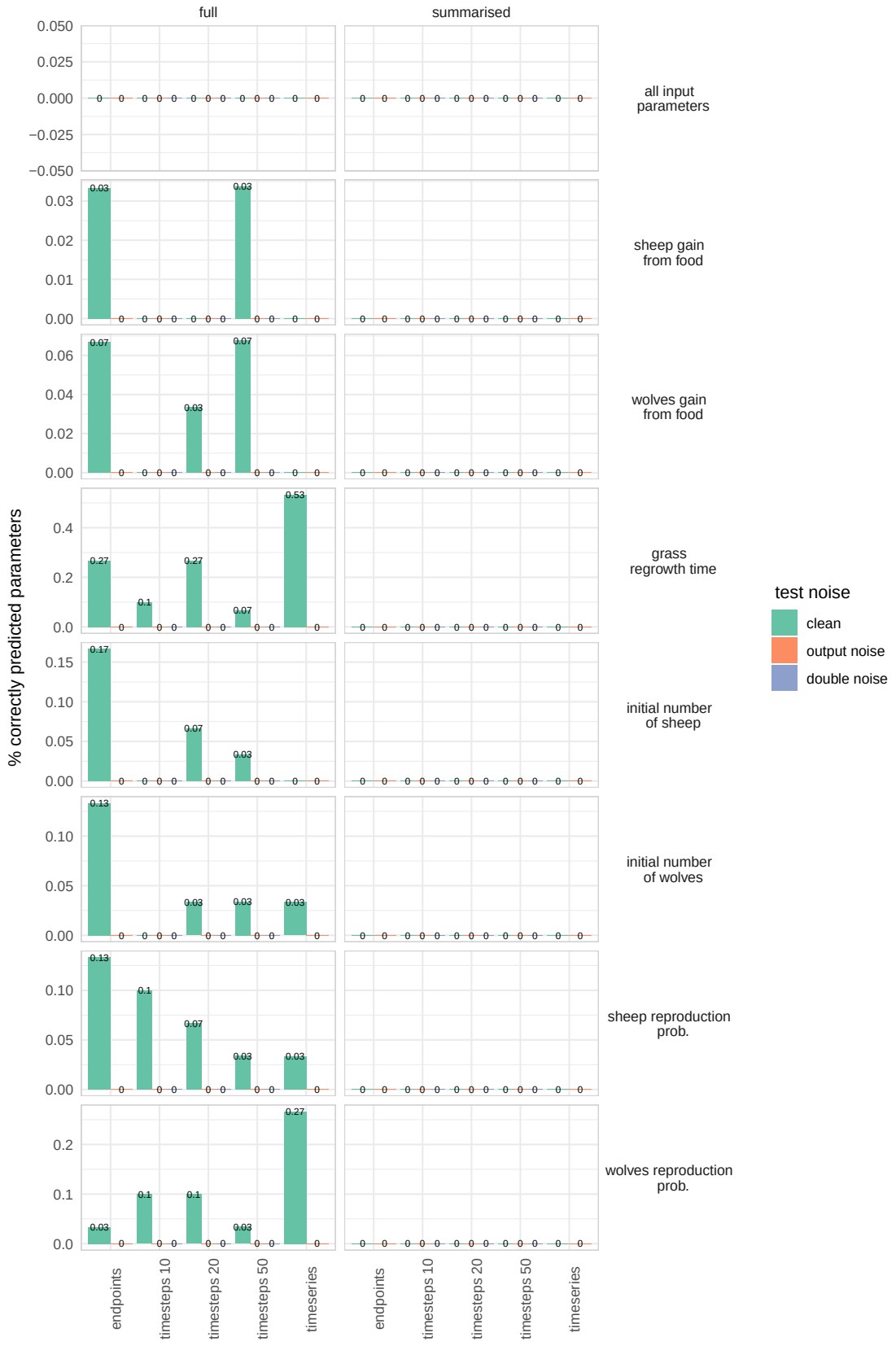


Figure B.20: Percentages of correctly estimated input parameters for the Wolf Sheep Predation ABM, using RF.

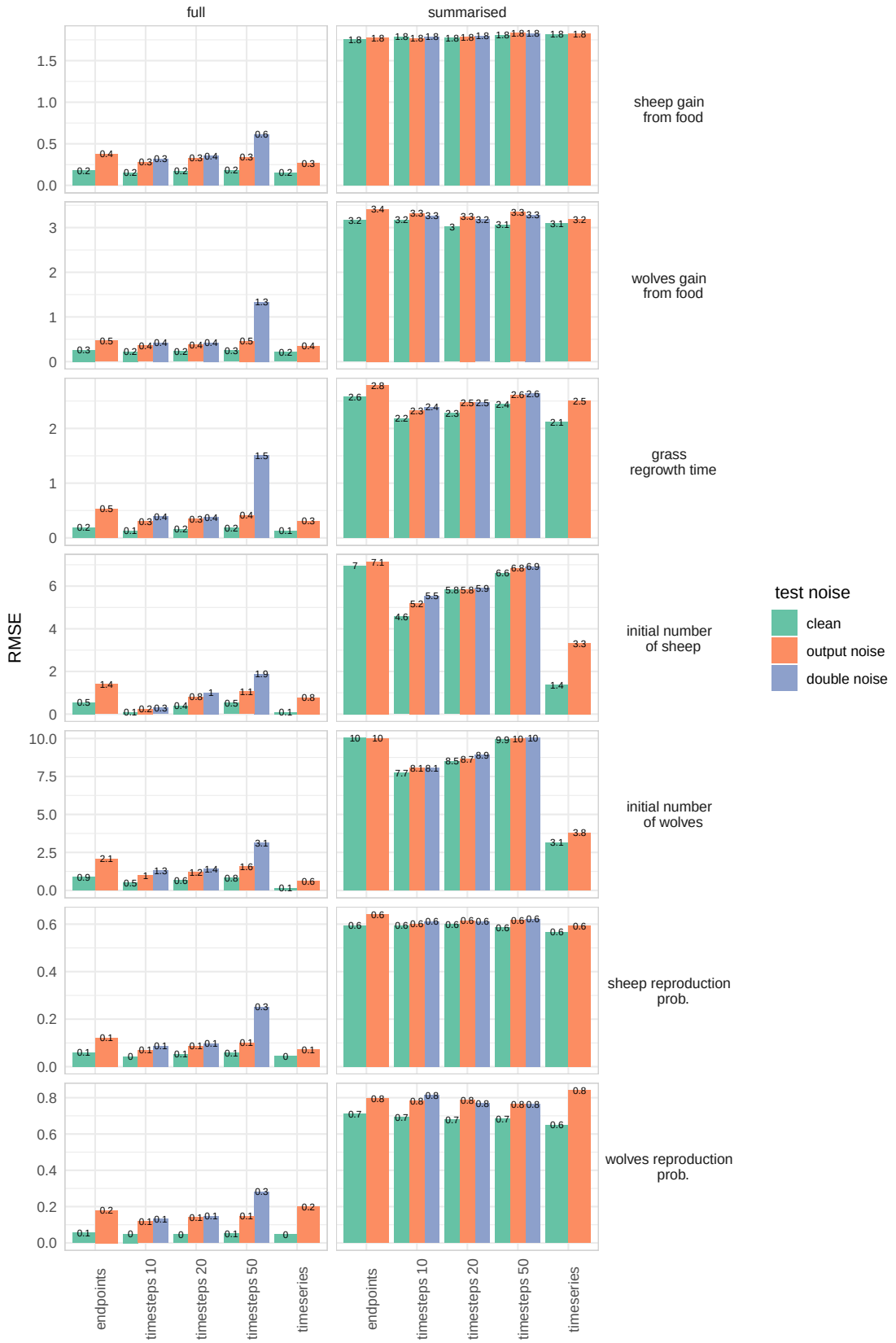


Figure B.21: Root mean squared error of the predicted vs. the true input parameters for the Wolf Sheep Predation ABM, using RF.

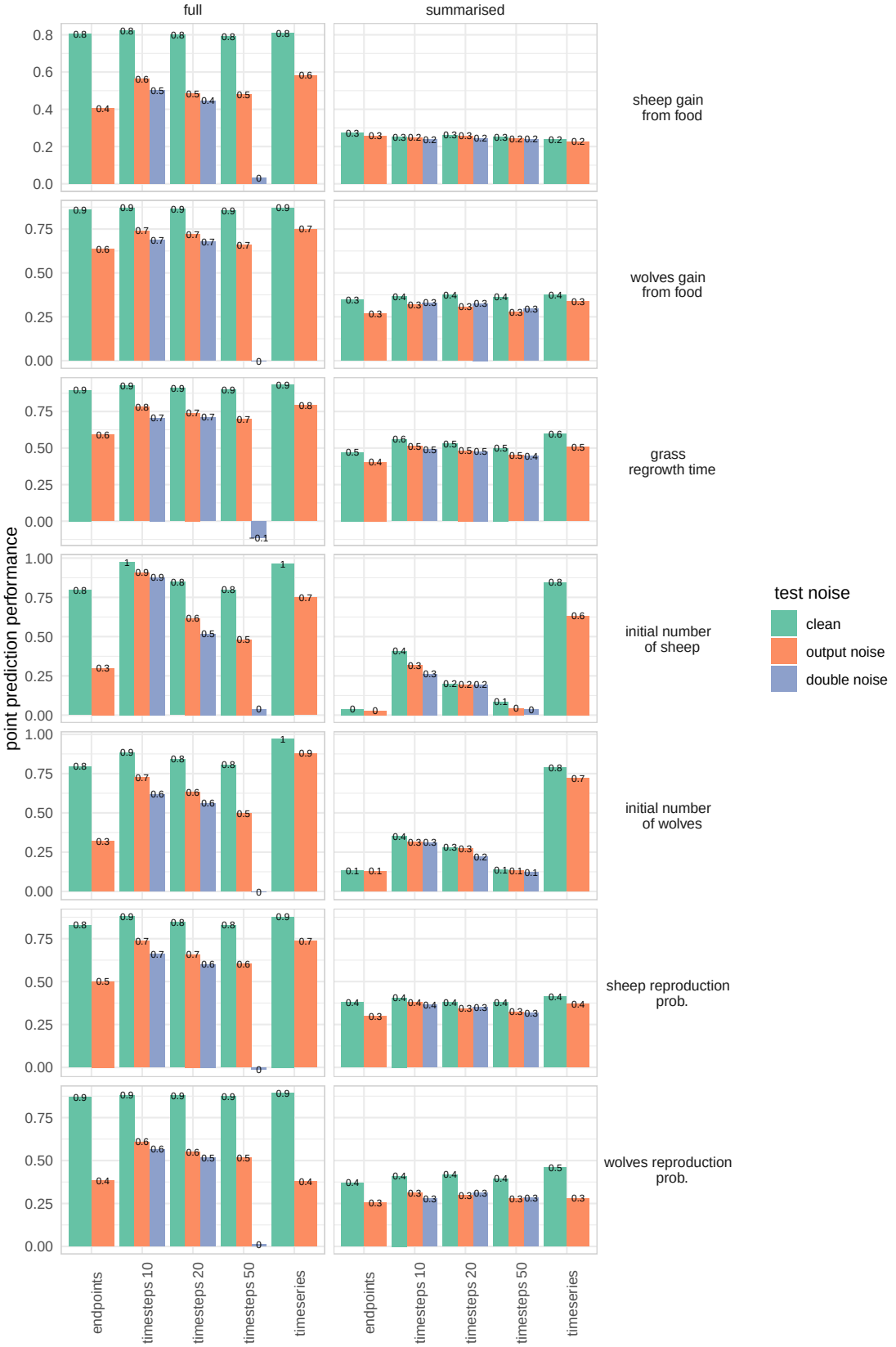


Figure B.22: Point prediction performance of the predicted input parameters for the Wolf Sheep Predation ABM, using RF.

B.5 Multivariate Random Forest

The multivariate random forest is by far the slowest approach and as such it was not possible to run the exact same analyses for this algorithm. Instead of trying to fit all 7 parameters, only the initial number of sheep and sheep reproduction input parameters were fitted. Additionally, only 100 randomly chosen parameter combinations out of the sample of 200 were used.

The MRF is unable to accurately predict either parameter when supplied with the summarised data (see Figure B.23). Although the performance is poor in all cases, the data without noise does clearly outperform the data with added noise. There is no clear difference due to the number of time steps included.

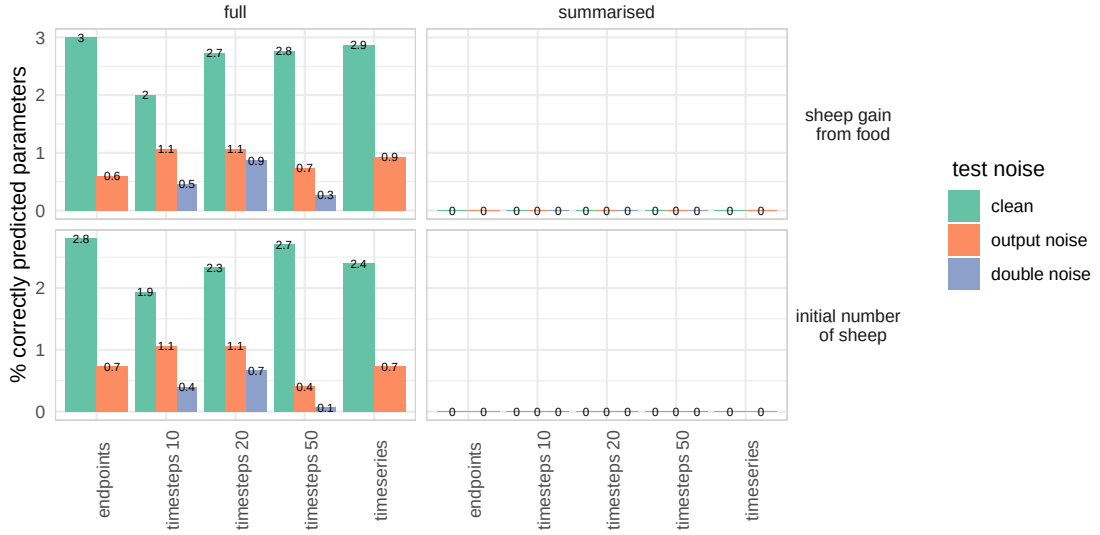


Figure B.23: Percentages of correctly estimated input parameters for the Wolf Sheep Predation ABM, using MRF.

The non-summarised data yields better performances than the summarised data in terms of the RMSE (see Figure B.24). The differences between the various amounts of time steps are small, although in the summarised data the data with 50 time steps performs noticeably better than those with 10, 20, or the full time series. The noise levels make little difference, although in some cases the double noise performs a bit worse than the clean and output noise data.

The point prediction performance indicates mis-identification of the sheep gain from food parameter in only one case (see Figure B.25). The clean non-summarised data performs better than any of the other datasets, regardless of the number of time steps included.

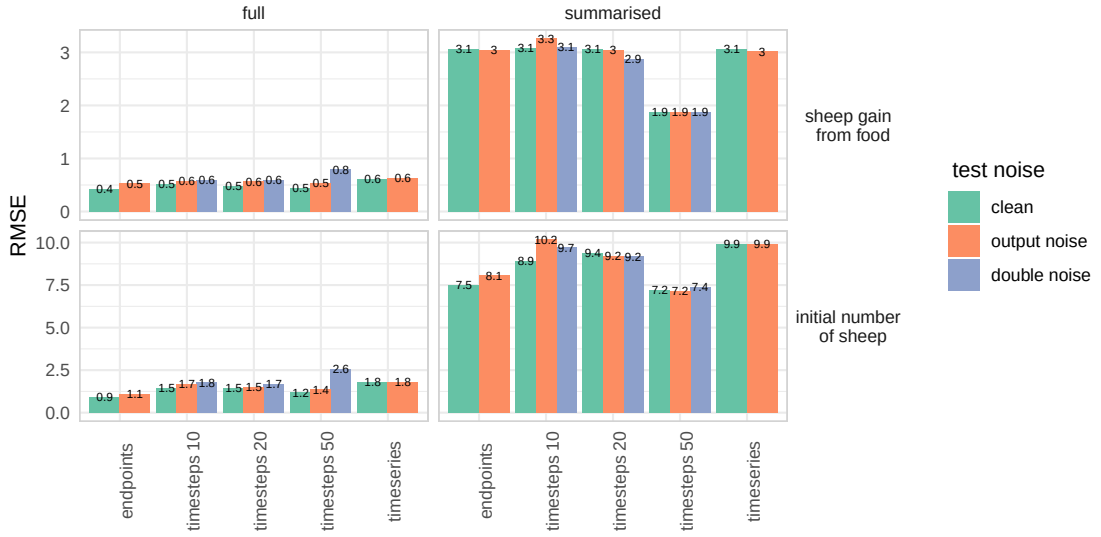


Figure B.24: Root mean squared error of the predicted vs. the true input parameters for the Wolf Sheep Predation ABM, using MRF.

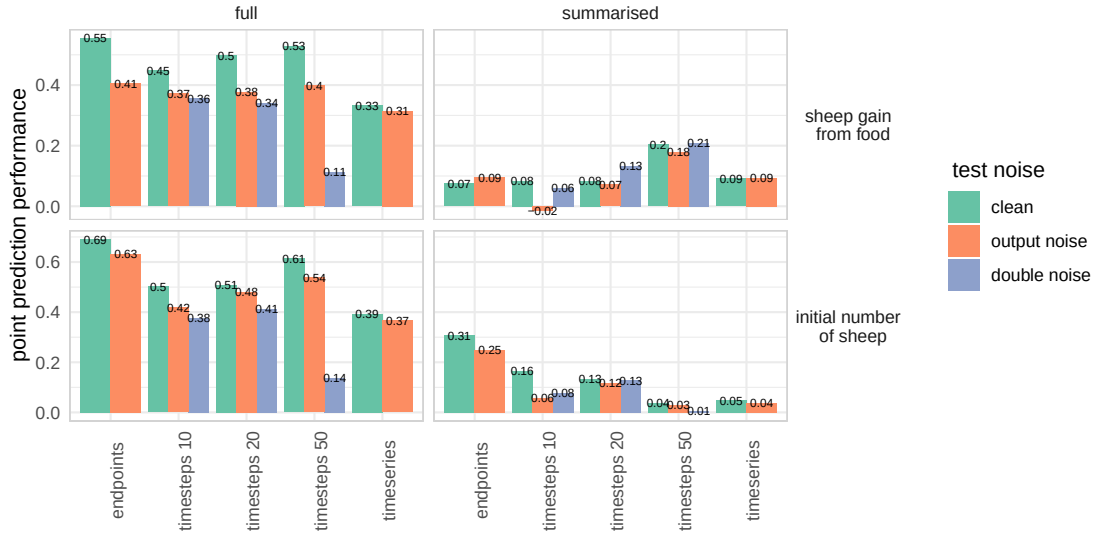


Figure B.25: Point prediction performance of the predicted input parameters for the Wolf Sheep Predation ABM, using MRF.

B.6 Compare algorithms within Wolf Sheep Predation Agent-Based Model

B.6.1 Algorithms

The RF and MRF outperform the linear regression and partial least squares algorithms in terms of the initial number of sheep parameter. The difference between RF and MRF is small, but the MRF has higher scores in terms of the number of times the parameter was guessed correctly (see Figure B.26).

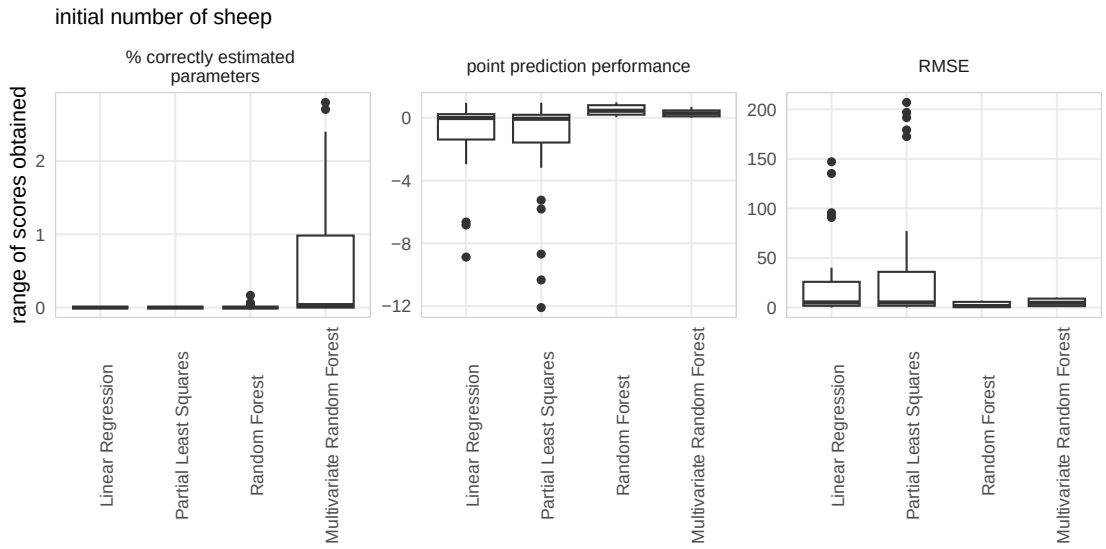


Figure B.26: The percentage correctly predicted initial number of sheep parameters, point prediction performance, and the RMSE between the true and predicted initial number of sheep parameters for the Wolf Sheep Predation ABM, per algorithm. Boxplots represent the range of results obtained with different numbers of time points, noise levels, and with summarised vs. non-summarised data.

The RF outperforms the linear regression and partial least squares algorithms in terms of the initial number of wolves parameter. There is very little discernible difference between the linear regression and partial least squares algorithms (see Figure B.27).

The RF and MRF outperform the linear regression and partial least squares algorithms in terms of the sheep gain from food parameter. The difference between RF and MRF is small, but the MRF has higher scores in terms of the number of times the parameter was guessed correctly (see Figure B.28).

The RF outperforms the linear regression and partial least squares algorithms in terms of the wolves gain from food parameter. There is very little discernible difference between the linear regression and partial least squares algorithms (see Figure B.29).

The RF outperforms the linear regression and partial least squares algorithms in terms of the sheep reproductive probability parameter. The linear regression performs slightly less well than the partial least squares algorithm (see Figure B.30).

The RF outperforms the linear regression and partial least squares algorithms in terms of the wolves reproductive probability parameter. There is very little discernible difference between the

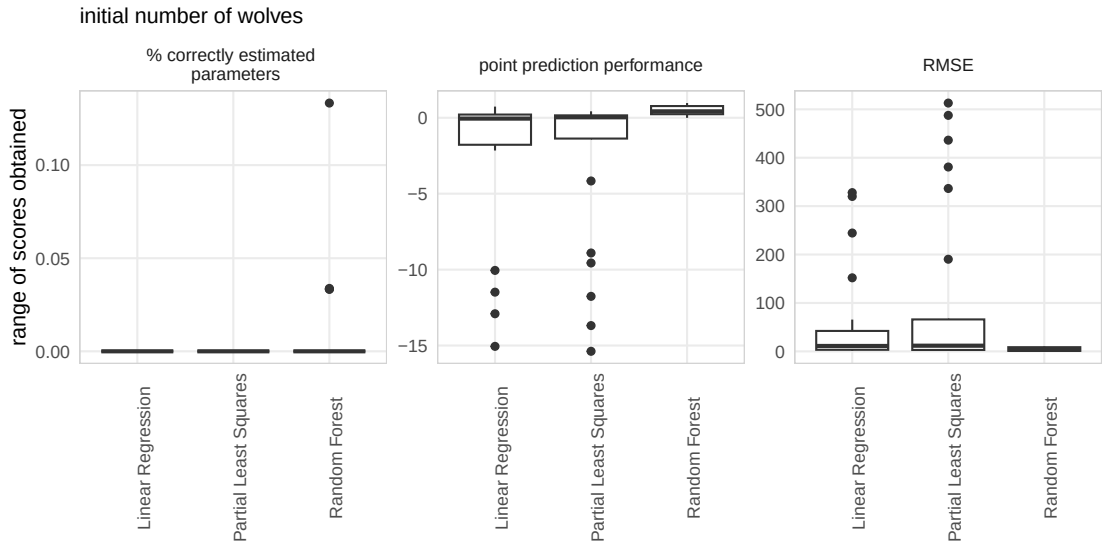


Figure B.27: The percentage correctly predicted initial number of wolves parameters, point prediction performance, and the RMSE between the true and predicted initial number of wolves parameters for the Wolf Sheep Predation ABM, per algorithm. Boxplots represent the range of results obtained with different numbers of time points, noise levels, and with summarised vs. non-summarised data.

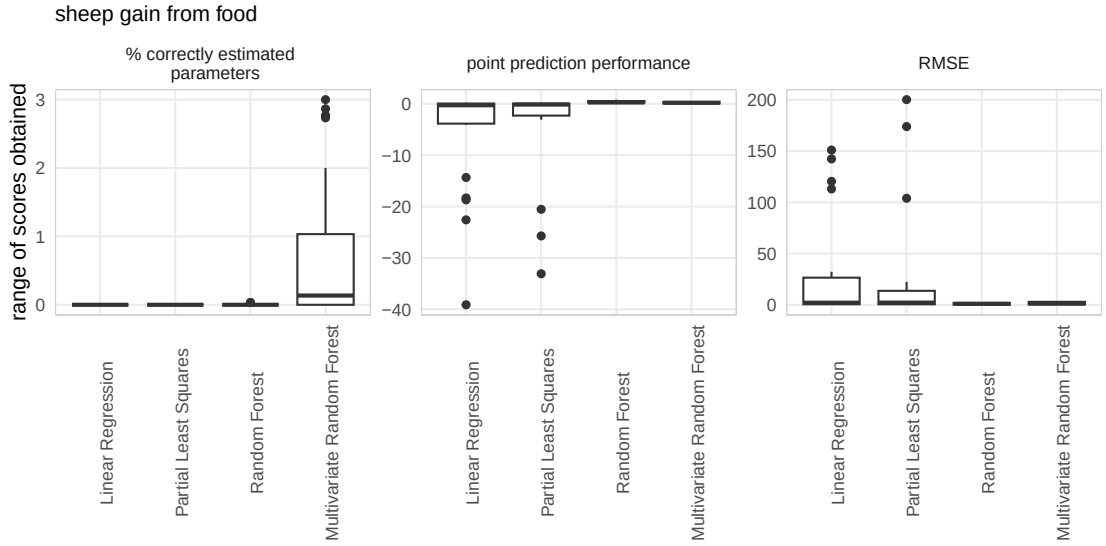


Figure B.28: The percentage correctly predicted sheep gain from food parameters, point prediction performance, and the RMSE between the true and predicted sheep gain from food parameters for the Wolf Sheep Predation ABM, per algorithm. Boxplots represent the range of results obtained with different numbers of time points, noise levels, and with summarised vs. non-summarised data.

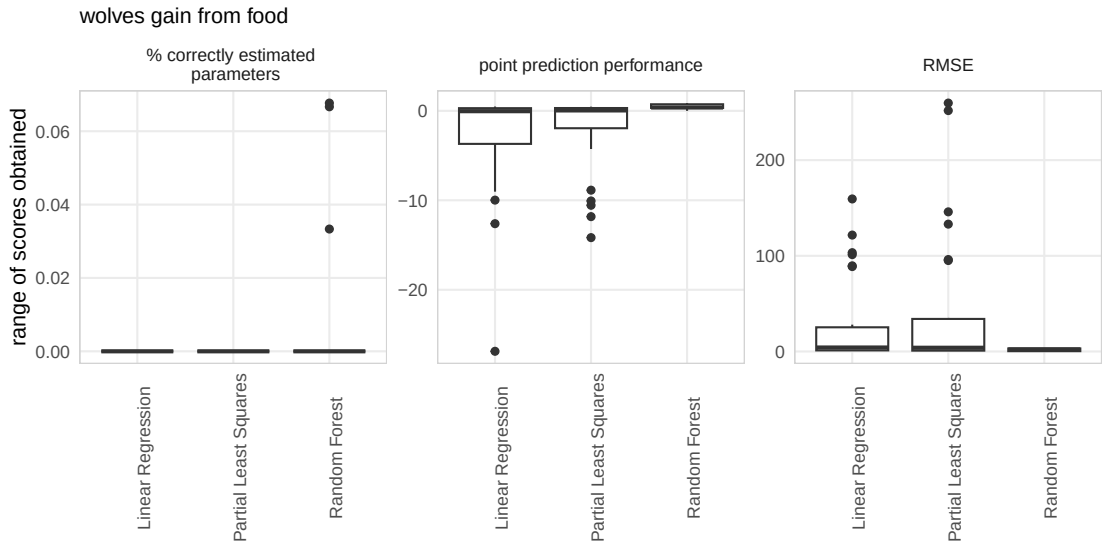


Figure B.29: The percentage correctly predicted wolves gain from food parameters, point prediction performance, and the RMSE between the true and predicted wolves gain from food parameters for the Wolf Sheep Predation ABM, per algorithm. Boxplots represent the range of results obtained with different numbers of time points, noise levels, and with summarised vs. non-summarised data.

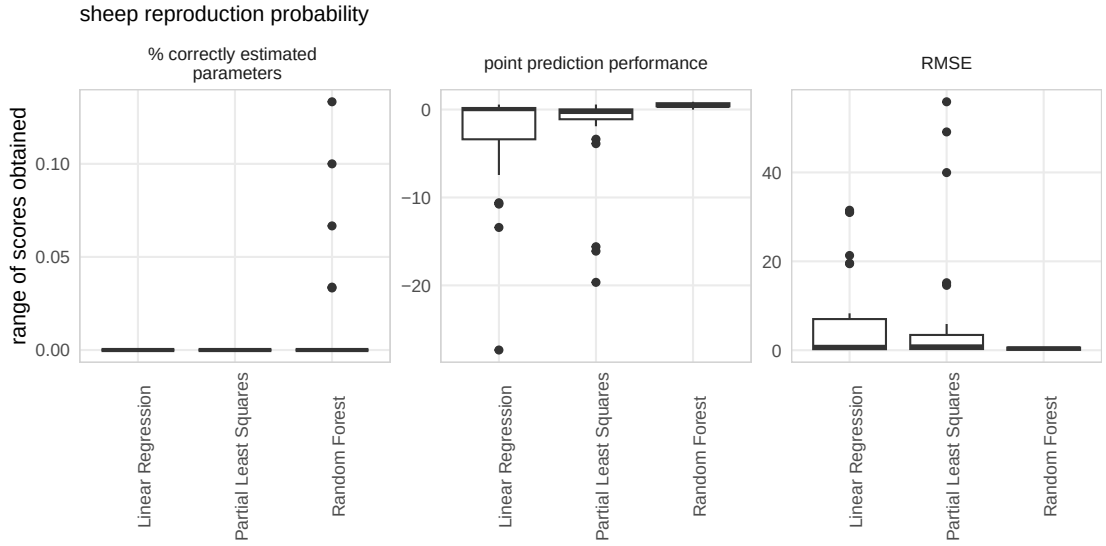


Figure B.30: The percentage correctly predicted sheep reproductive probability parameters, point prediction performance, and the RMSE between the true and predicted sheep reproductive probability parameters for the Wolf Sheep Predation ABM, per algorithm. Boxplots represent the range of results obtained with different numbers of time points, noise levels, and with summarised vs. non-summarised data.

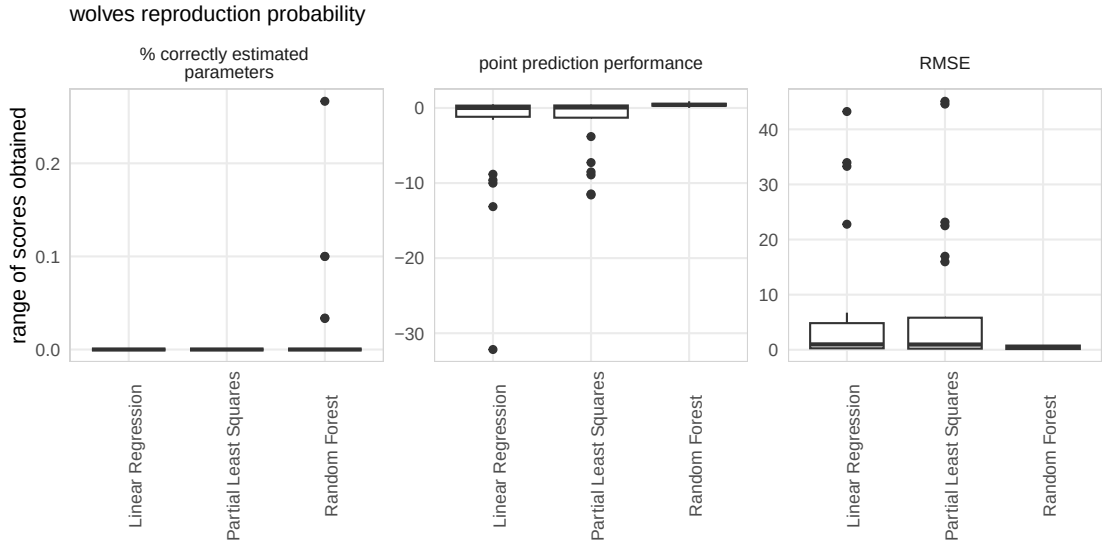


Figure B.31: The percentage correctly predicted wolves reproductive probability parameters, point prediction performance, and the RMSE between the true and predicted wolves reproductive probability parameters for the Wolf Sheep Predation ABM, per algorithm. Boxplots represent the range of results obtained with different numbers of time points, noise levels, and with summarised vs. non-summarised data.

linear regression and partial least squares algorithms (see Figure B.31).

The RF outperforms the linear regression and partial least squares algorithms in terms of the grass regrowth time parameter. There is very little discernible difference between the linear regression and partial least squares algorithms (see Figure B.32).

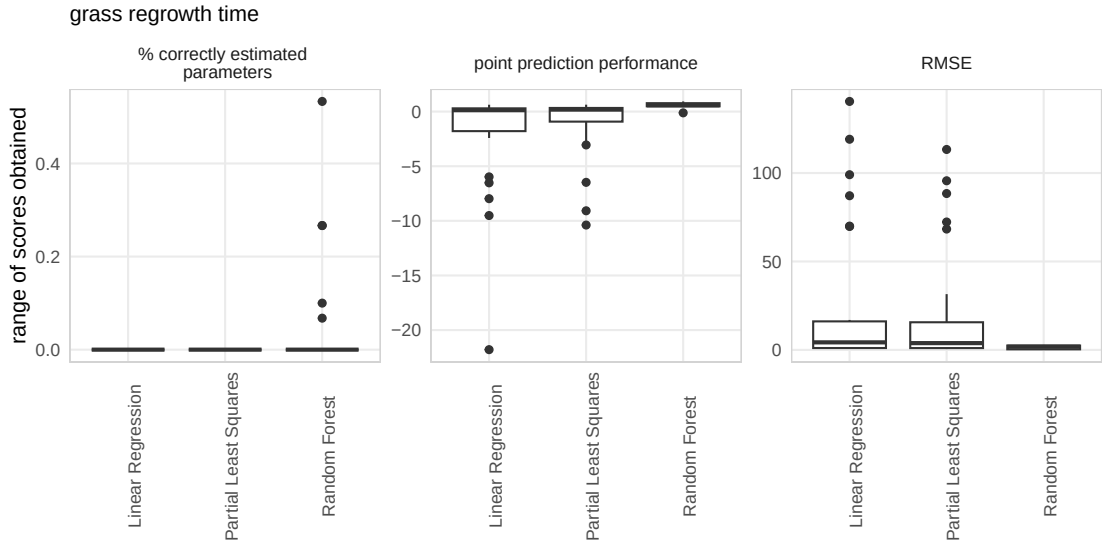


Figure B.32: The percentage correctly predicted grass regrowth time parameters, point prediction performance, and the RMSE between the true and predicted grass regrowth time parameters for the Wolf Sheep Predation ABM, per algorithm. Boxplots represent the range of results obtained with different numbers of time points, noise levels, and with summarised vs. non-summarised data.

B.6.2 Data

The trends for the effect of data are approximately the same across all seven parameters. The non-summarised data outperforms the summarised data, but there are no consistent patterns within the clean vs. noisy data and the number of time steps included (see Figures B.33, B.34, B.35, B.36, B.37, B.38, and B.39).

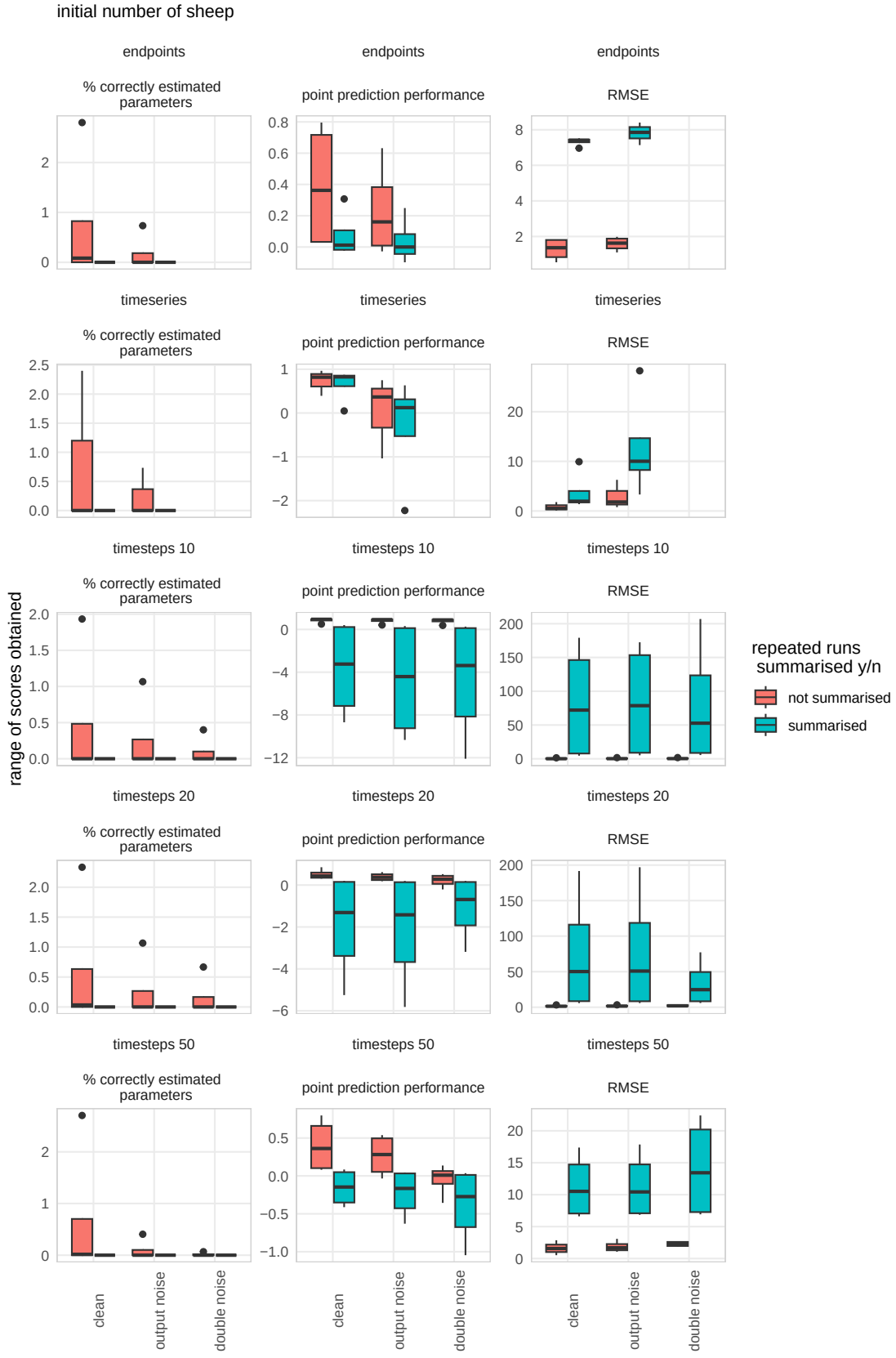


Figure B.33: The percentage correctly predicted initial number of sheep parameters, point prediction performance, and the RMSE between the true and predicted initial number of sheep parameters for the Wolf Sheep Predation ABM, per data type. Boxplots represent the range of results obtained with different algorithms.

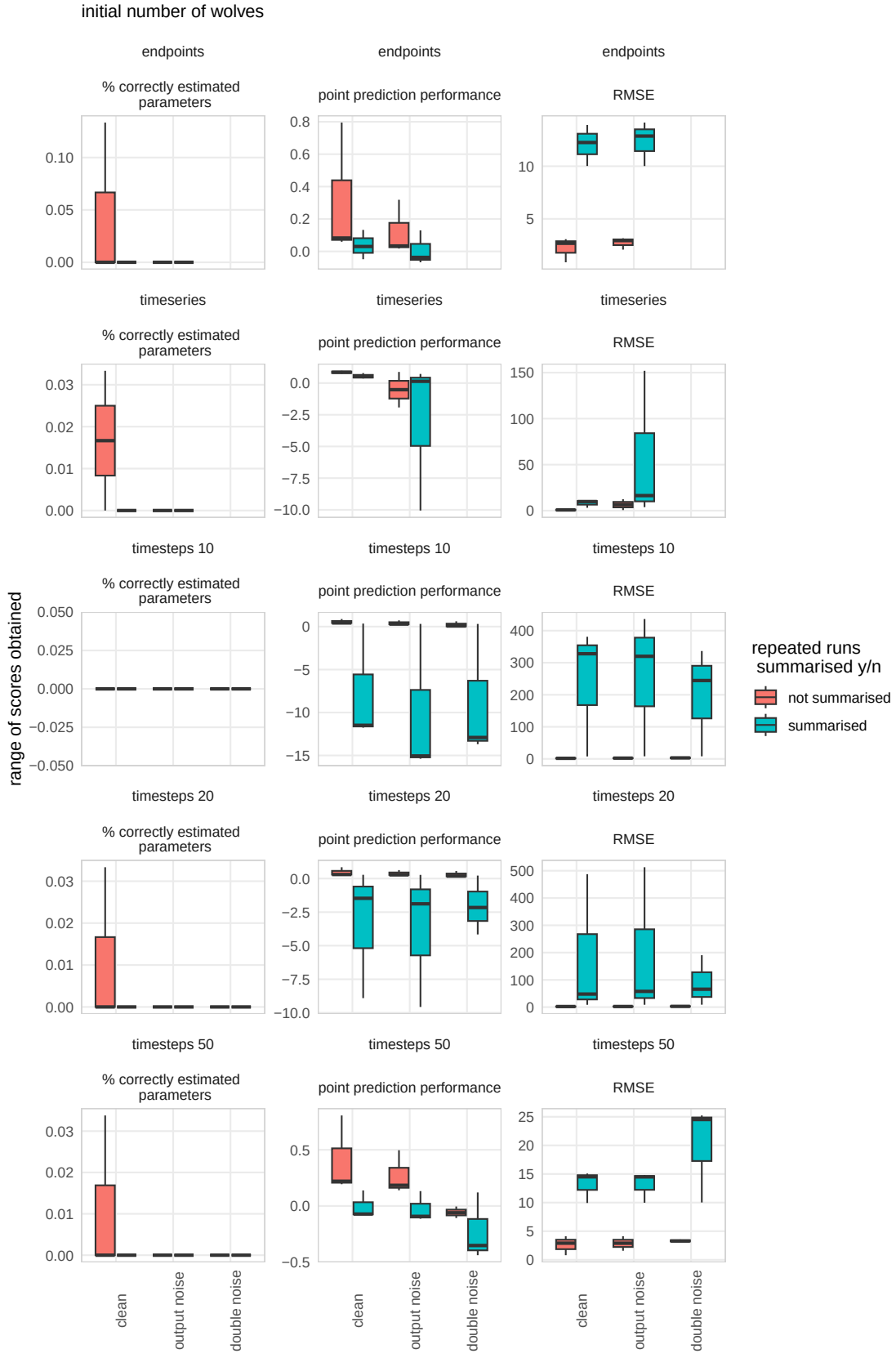


Figure B.34: The percentage correctly predicted initial number of wolves parameters, point prediction performance, and the RMSE between the true and predicted initial number of wolves parameters for the Wolf Sheep Predation ABM, per data type. Boxplots represent the range of results obtained with different algorithms.

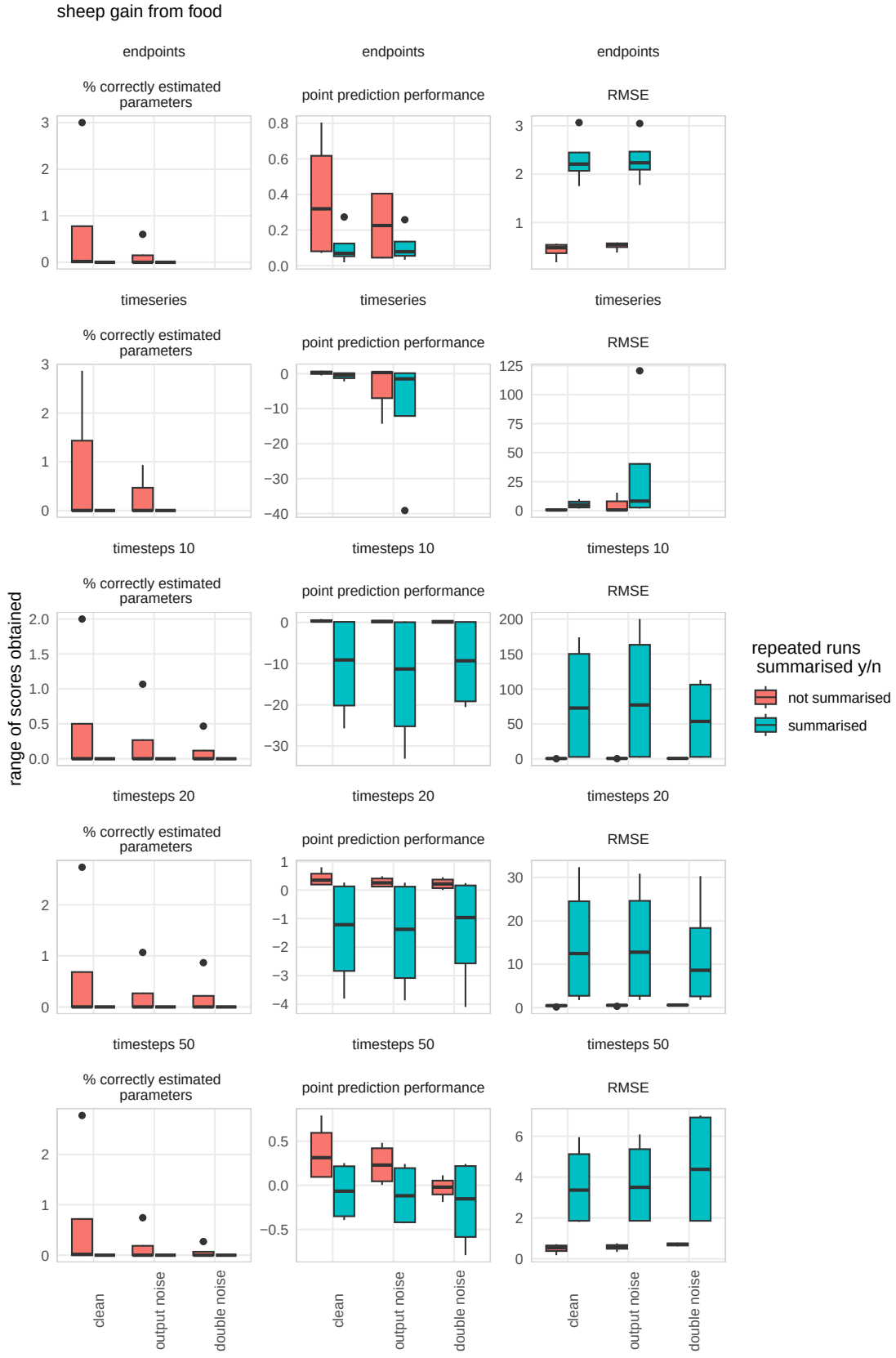


Figure B.35: The percentage correctly predicted sheep gain from food parameters, point prediction performance, and the RMSE between the true and predicted sheep gain from food parameters for the Wolf Sheep Predation ABM, per data type. Boxplots represent the range of results obtained with different algorithms.

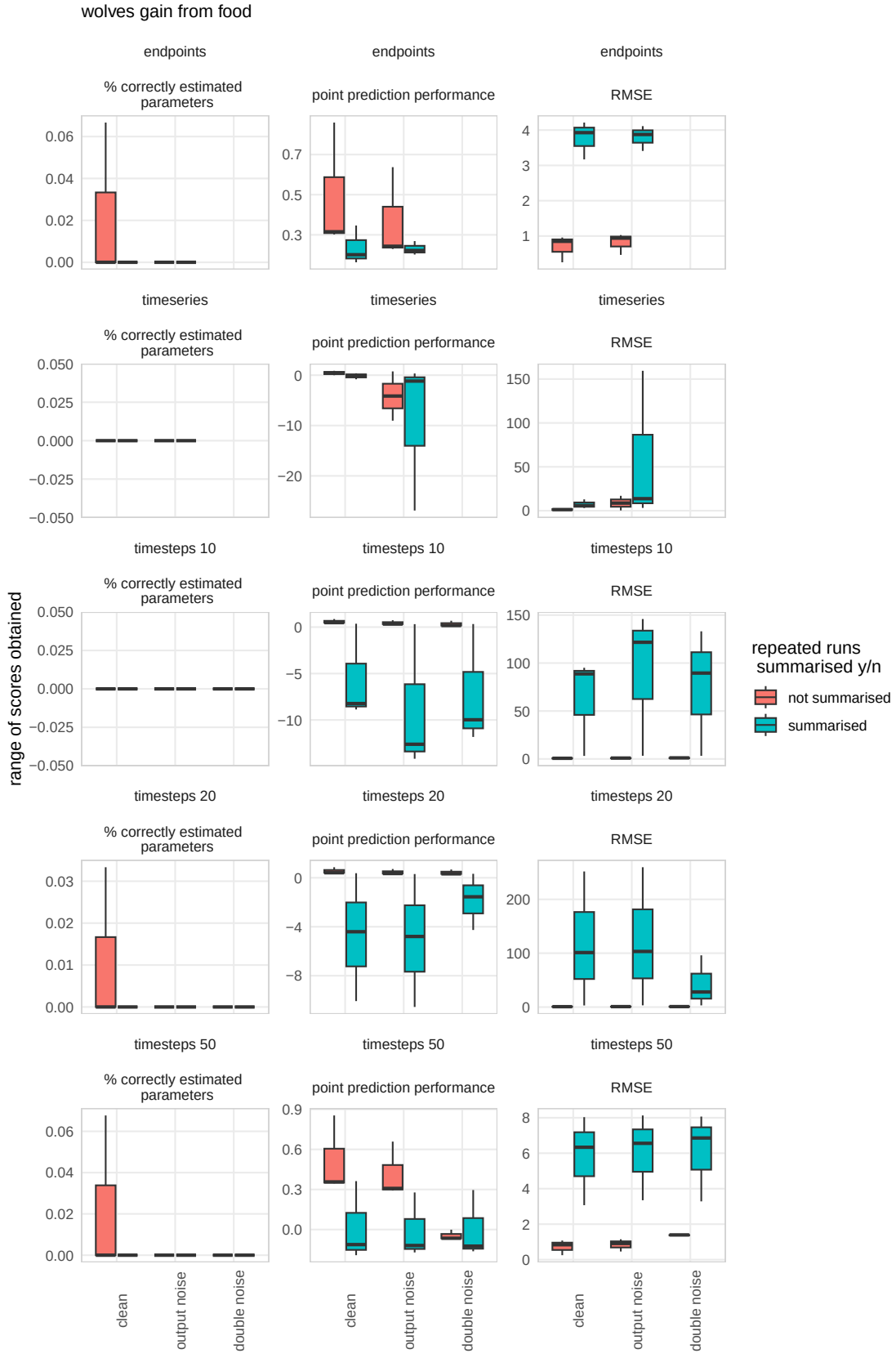


Figure B.36: The percentage correctly predicted wolves gain from food parameters, point prediction performance, and the RMSE between the true and predicted wolves gain from food parameters for the Wolf Sheep Predation ABM, per data type. Boxplots represent the range of results obtained with different algorithms.

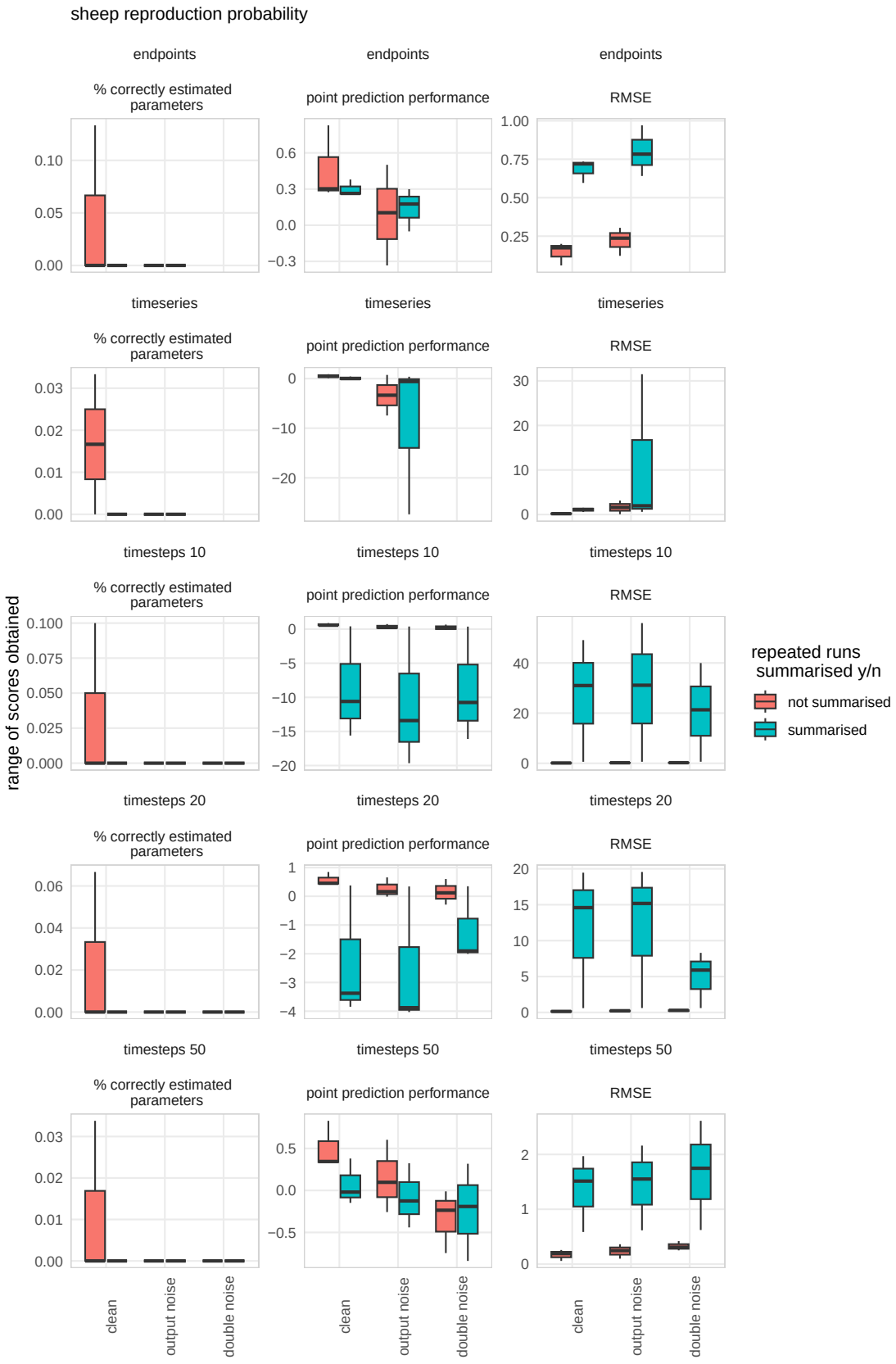


Figure B.37: The percentage correctly predicted sheep reproductive probability parameters, point prediction performance, and the RMSE between the true and predicted sheep reproductive probability parameters for the Wolf Sheep Predation ABM, per data type. Boxplots represent the range of results obtained with different algorithms.

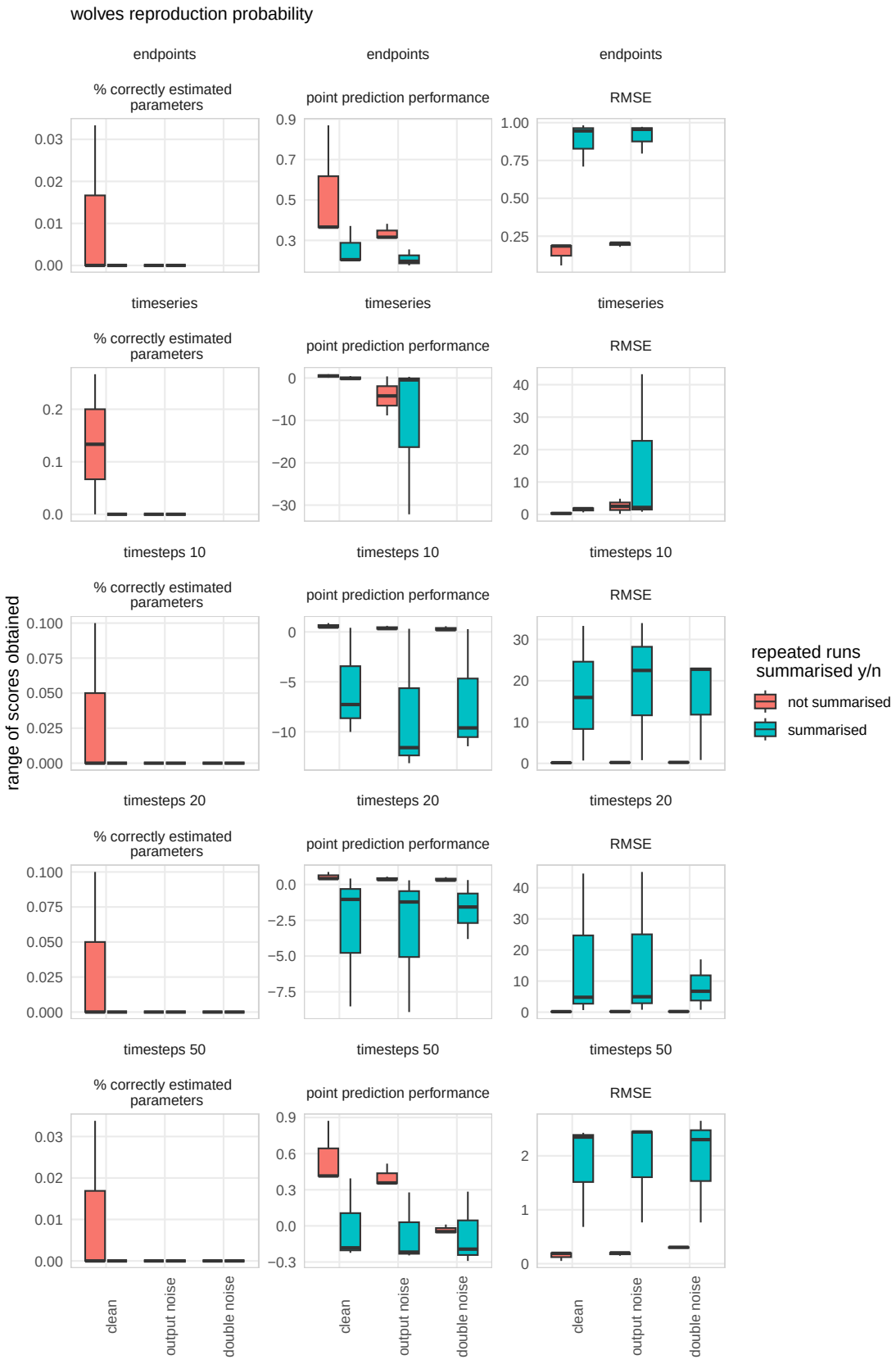


Figure B.38: The percentage correctly predicted wolves reproductive probability parameters, point prediction performance, and the RMSE between the true and predicted wolves reproductive probability parameters for the Wolf Sheep Predation ABM, per data type. Boxplots represent the range of results obtained with different algorithms.

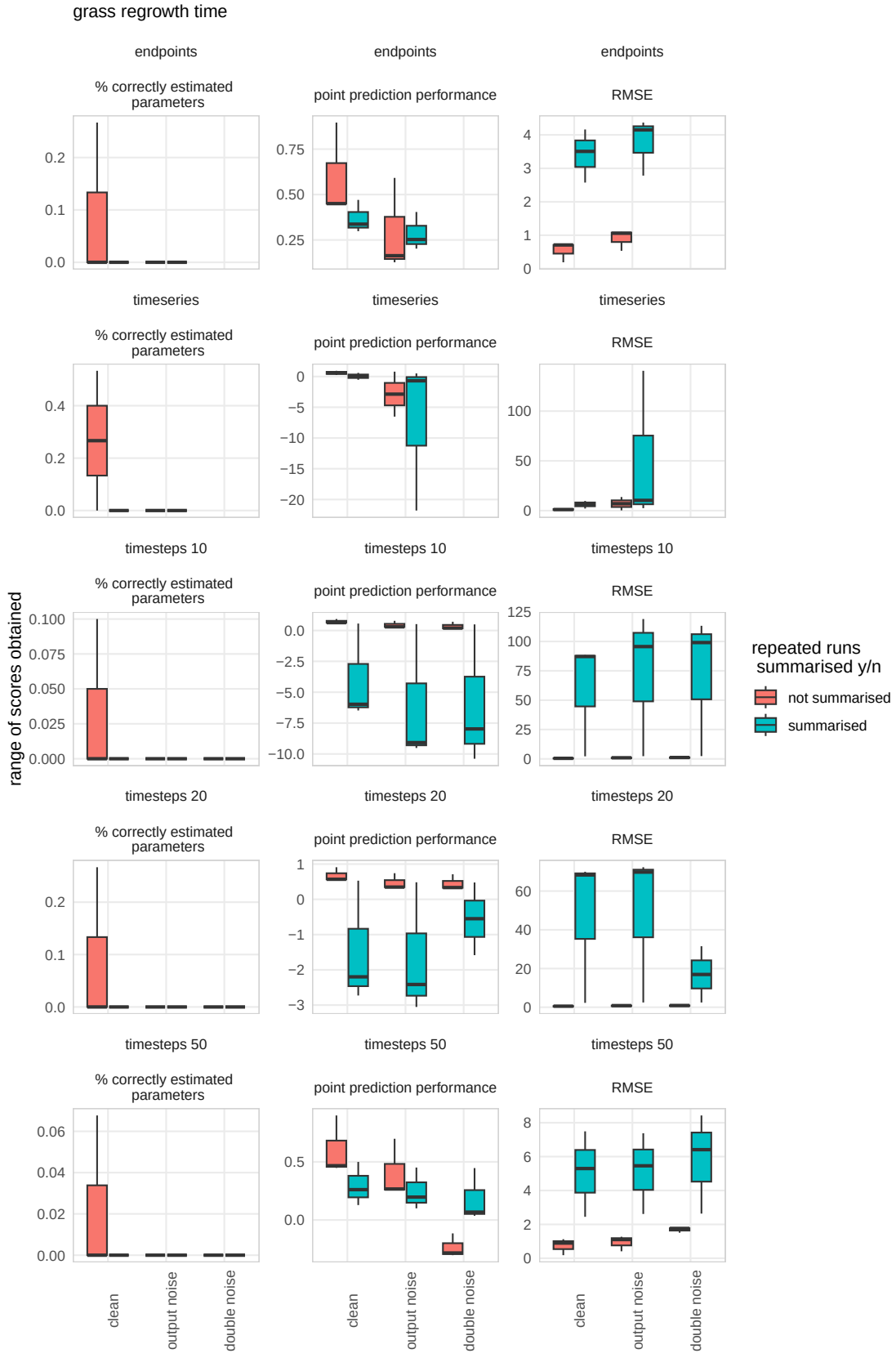


Figure B.39: The percentage correctly predicted grass regrowth time parameters, point prediction performance, and the RMSE between the true and predicted grass regrowth time parameters for the Wolf Sheep Predation ABM, per data type. Boxplots represent the range of results obtained with different algorithms.