Universiteit Leiden

Master Computer Science

A Markov Reward Process-Based Approach to Spatial Interpolation

Name:	Laurens Arp
Student ID:	2369036
Date:	29/10/2020
Specialisation:	Advanced Data Analytics – Artificial Intelligence Track
1st supervisor:	Dr. Mitra Baratchi
2nd supervisor:	Prof.dr. Holger Hoos

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The interpolation of spatial data can be of tremendous value in various applications, such as forecasting weather from only a few measurements of meteorological or remote sensing data. Existing methods for spatial interpolation, such as variants of kriging and spatial autoregressive models, tend to suffer from at least one of the following limitations: (a) the assumption of stationarity, (b) the assumption of isotropy, and (c) the trade-off between modelling local or global spatial interaction. Addressing these issues in this work, we propose the use of Markov reward processes (MRPs) as a spatial interpolation method, and we introduce three variants thereof: (i) a basic static discount MRP (SD-MRP), (ii) an accurate but mostly theoretical optimised MRP (O-MRP), and (iii) a transferable weight prediction MRP (WP-MRP). All variants of MRP interpolation operate locally, while also implicitly accounting for global spatial relationships in the entire system through recursion. Additionally, O-MRP and WP-MRP no longer assume stationarity and are robust to anisotropy. We evaluated our proposed methods by comparing the mean absolute errors of their interpolated grid cells to those of 7 common baselines, selected from models based on spatial autocorrelation, (spatial) regression, and deep learning.

We performed detailed evaluations on two publicly available datasets (local GDP values, and COVID-19 patient trajectory data). The results from these experiments clearly show the competitive advantage of MRP interpolation, which achieved significantly lower errors than the existing methods in 23 out of 40 experimental conditions, or 35 out of 40 when including O-MRP.

Contents

Chapter 1

Introduction

Research or industry, data science or application, to many of us missing data remains a fact of life. The problem of "filling in" missing data between known observations is known as interpolation. This problem can become especially difficult to solve in spatial settings, as observations are made in two- or three-dimensional space, and one cannot necessarily assume *stationarity* – that is, the distribution of values may change depending on the location in question. Moreover, some spatial effects may have different properties depending on the direction of the interaction; this is referred to as *anisotropy* (as opposed to isotropy).

Nonetheless, spatial interpolation is typically done by exploiting the property of *spatial autocorrelation*: the values of a variable tend to be more similar the closer their spatial proximity is to one another. While this concept has been useful in designing spatial models, pure spatial autocorrelation would only consider either local or distance-based pair-wise (global) interactions, and not the geographical properties of any intermediate locations. This could be problematic; for example, two locations separated by a mountain range or a body of water may be less similar than two locations that are part of the same city, even if the distances between both pairs of locations were the same. The problem is further exacerbated by the two-dimensionality of the problem, as there are many different paths from one point on a 2D grid to another.

To date, a number of spatial interpolation methods have been proposed. Some of these, such as kriging [1] and inverse distance weighting [2], rely solely on the spatial autocorrelation of the target variable to predict unknown values in a grid. Others, such as spatial autoregressive- [3] and moving average [4] models, optionally extend this with the use of explanatory variables. All these methods tend to suffer from at least one of the problems mentioned above, with individual methods offering different trade-offs in assumptions and limitations.

In this thesis, we propose a Markov reward process (MRP)-based approach to spatial interpolation aimed at overcoming these limitations. This approach, inspired by the application of MRPs in reinforcement learning [5], will treat the target variable prediction at a certain location as the "expected reward" of a

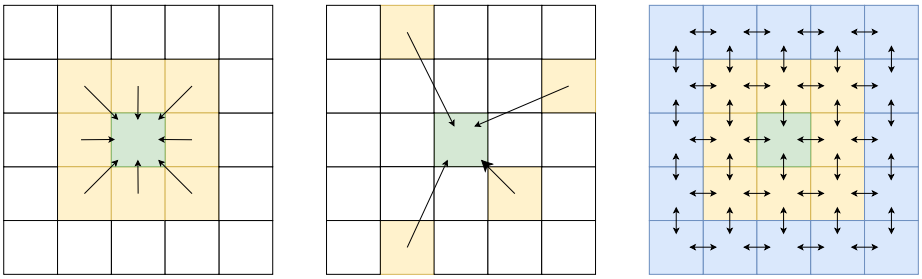


Figure 1.1: Illustration of local (left), distance-based (middle) and system-oriented (right) perspectives. In local and distance-based perspectives, the predicted value of the green cell is determined by the yellow cells (equal weights if local, unequal weights if distance-based). In the system-oriented perspective, which is the perspective used by our proposed method, the prediction for the green cell is made using the yellow neighbours, which were in turn affected by their own neighbours (blue, yellow and green cells).

state. While this interpolation method is local in principle, the use of a recursive definition using predicted neighbouring values allows MRPs to be implicitly global. This enables us to consider the entire region as a global system of locations, each with particular local properties, indirectly mutually interacting with all other locations via multiple paths, as illustrated in Figure 1.1.
The main challenges in adapting MRPs to spatial interpolation are twofold. First, the necessary equivalence needs to be formalised between reinforcement learning concepts and their respective interpolation counterparts. Second, as MRPs are based on the concept of *discounts* that can be exploited to represent spatial autocorrelation, determining the best discount setting poses a challenge. Simple approaches might use a single discount parameter, automatically tuned using a training set and applied to a test set, though this would assume both stationarity and isotropy. A more ambitious approach might be to construct a similarly data-driven manner of assigning weights, but specific to every individual pair of neighbours.

Our main contributions in this work are as follows:

- We propose a novel, MRP-based spatial interpolation method, combining the strengths of local spatial autorcorrelation and implicit global spatial interactions.
- We introduce three MRP interpolation models, two of which are robust to anisotropy and non-stationarity.
- We evaluate our methods in terms of mean absolute error on spatial data from four cities from two different countries. As target variables, we used a global dataset on GDP by World Bank [6], where we also tested for international transferability, and a South Korean dataset of COVID-19 patient trajectories [7].

- We compare our methods to various existing models for spatial interpolation, including kriging, spatial autoregressive models and convolutional neural networks. In this comparison, a practically applicable MRP outperformed all baselines in 23 out of 40 conditions. All methods were configured using automated algorithm configuration for (close to) optimal performance and to ensure a fair comparison.

In the next chapter (Chapter 2), we will formalise the problem of spatial interpolation, and introduce the notation we will use throughout the thesis. Chapter 3 will cover existing work and methods for spatial interpolation, and Chapter 4 will explain our own proposed methods in detail. As for the evaluation of those methods, Chapter 5 will cover the details of our experimental setup, and Chapter 6 will analyse the results of those experiments. We conclude the thesis with the conclusion in Chapter 7, and also add an ethical statement to give some insight into the ethical implications of our proposed methods, as well as the responsible use of our experimental datasets.

Chapter 2

Problem statement

We formalise the problem of spatial interpolation as follows. First, we define a two-dimensional matrix $\mathbf{Y}$ of size $H \times W$ of target variable values $y_{h,w}$, where h represents the row index and w represents the column index of a cell in $\mathbf{Y}$. Moreover, for convenient indexing we define $\mathbf{c} = [c_1, c_2, ..., c_{H \times W}]$, where every $c \in \mathbf{c}$ represents an index pair (h, w). For a target variable y of interest, every cell in the matrix indexed by c is associated with a true value y_c^*, which may be either known in $\mathbf{Y}$ or not. If it is known, we set the target value $y_c = y_c^*$, and if it is not known, we set $y_c = \varnothing$ (null value, not to be confused with an empty set). Second, we define the spatial feature vectors $\mathbf{x}_c = [x_c^1, x_c^2, ..., x_c^D]$ (where D is the number of spatial features) for all $c \in \mathbf{c}$. These feature vectors contain explanatory variables derived from the geographical and spatial properties of the location at c, such as the number of houses, hospitals and bus stops.
The problem of spatial interpolation can now be formulated as creating a model $\mathcal{M}(\mathbf{Y}, \mathbf{x})$ using known true values from $\mathbf{Y}$ and/or explanatory variable vectors $\mathbf{x}_c$ to predict an estimated target value $\hat{y}_c = \mathcal{M}(\mathbf{Y}, \mathbf{x}_c)$ for all c where $y_c = \varnothing$. Our objective is to minimise the mean difference between $\hat{y}_c$ and the unknown true values y_c^*. Concretely, if we define $\mathbf{Y}_\varnothing$ as $\{c : y_c = \varnothing\}$, we wish to find an optimal model $\mathcal{M}^*$:

$$\mathcal{M}^* \in \underset{\mathcal{M}}{\operatorname{argmin}} \frac{1}{|\mathbf{Y}_\varnothing|} \sum_{c \in \mathbf{Y}_\varnothing} (\mathcal{M}(\mathbf{Y}, \mathbf{x}_c) - y_c^*) \tag{2.1}$$

Chapter 3

Related work

To date, various spatial interpolation methods have been proposed. For convenience, we will categorise methods into (i) *purely spatial methods*, and (ii) *regression with spatial and explanatory variables*. Purely spatial methods consist of those models relying solely on known true values and their spatial locations to fill in the grid: $\hat{y}_c = \mathcal{M}(\mathbf{Y})$. Conversely, regression with spatial and explanatory variables predict $\hat{y}_c$ using spatial autocorrelation, while also supporting the use of explanatory variables: $\hat{y}_c = \mathcal{M}(\mathbf{Y}, \mathbf{x}_c)$.

3.1 Purely spatial methods

Given the widespread use of kriging-based interpolation, we will sub-categorise the purely spatial methods further into kriging and non-kriging methods.

3.1.1 Kriging

Kriging [1, 8], more generally known as Gaussian processes outside of geostatistics, is a set of methods centred on the concept of the covariance of values and distance. That is, the relationship between the value of the target variable in two locations and the distance between these locations (spatial autocorrelation) is being modeled. This model, called the *variogram* (also known as the *kernel* in Gaussian processes) can take various forms, such as linear, exponential or Gaussian functions. Contemporary contributions to the kriging field include a scalable gradient-based surrogate function method [9] and a neural network-based method to overcome kriging's limitation of disregarding the characteristics of intermediate locations in paths between pairs of locations [10].

Let us define a cell of interest as c_i, and a neighbouring cell of c_i as c_j. The basic model for kriging interpolation can now be formulated as follows:

$$y_{c_i} = \sum_{c_j \in N(c_i)} (\gamma_{c_j, c_i} \cdot y_{c_j}) + \epsilon_{c_i} \tag{3.1}$$

Here $N(c_i)$ is a sample of locations with known measurements neighbouring c_i, γ_{c_j,c_i} is a weight scalar associated with neighbour c_j, and ϵ_{c_i} is the error or residual of the model. Essentially, Equation 3.1 is taking a weighted sum of a sample of neighbouring values. The challenge in kriging is to determine the various weights γ_{c_j,c_i}, which is what the variogram is used for. The variability of measurements $var(d)$ is modeled as a function of the distance d. In its basic form, this results in the following formula:

$$var(d) = \frac{1}{2 \cdot c(d)} \cdot \sum_{(c_i,c_j) \in D_d} (y_{c_i} - y_{c_j})^2 \tag{3.2}$$

Here $c(d)$ is the number of times a distance d appeared in the dataset and $\frac{1}{2 \cdot c(d)}$ is a normalisation term. Furthermore, we define D_d as the set of location pairs (c_i, c_j) where $d(c_i, c_j) = d$; that is, the set of all location pairs sharing the same distance between the elements of the pairs. From this, a vector γ of weights γ_{c_j,c_i} can be computed by solving the matrix equation

$$\gamma = A^{-1} \cdot b \tag{3.3}$$

Here A is matrix of size $(|N(c_i)| \times |N(c_i)|)$ containing $var(d)$ for the distances $d(c_j, c_{j^*})$ for all combinations of c_j and c_{j^*} in $N(c_i)$, and b is a vector containing $var(d)$ for the distances $d(c_i, c_j)$ for all c_j in $N(c_i)$. Solving Equation 3.3 gives us the weight vector γ needed to perform interpolation using Equation 3.1.
It is worth mentioning that the kriging model suffers from a number of assumptions, most notably stationarity and anisotropy. Moreover, the covariance of the target variable over distance (variogram) is assumed to be constant for the entire dataset. Specific variants of kriging offer different trade-offs in assumptions. We we will now describe a selection of these specific kriging variants, although the list will by no means be exhaustive.

First, let us consider **ordinary kriging** (OK) [11]. In this variant, the assumption is one of strict stationarity: the (unknown) mean of the target variable is assumed to be constant throughout the entire dataset. To alleviate the problems of strict stationarity, **universal kriging** (UK) [12] allows for a trend in the mean to exist within the dataset, though it still assumes weak stationarity, as it only allows for a trend in the mean modelled on the coordinates of the location in question. Moreover, Journel and Rossi found that modelling trends in kriging only matters in extrapolation, rather than interpolation [13]. In **simple kriging** [14] the mean is known and assumed to be constant, which is then used in its predictions. A final variant worth special attention is **regression kriging** [15], also known as kriging after detrending, which bears some resemblance to our later approach as it combines elements of kriging and regression. In this variant, the mean is assumed to have a drift as in universal kriging, but unlike universal kriging, this drift is predicted using external explanatory variables rather than coordinates in a separate model. Strongly related would be **kriging with external drift** [16], which combines the drift prediction and kriging into a single form.

Regardless of the degree to which stationarity and isotropy is assumed by any variant, all kriging-based methods are limited by their reliance on pair-wise distance-based covariance models. Intermediate locations are not considered, and explanatory features are typically only used to compensate for trends in the data. However, depending on the region and dataset in question, kriging can perform very well, and can be applied fairly easily to any grid.

3.1.2 Non-kriging methods

Although the various forms of kriging tend to be the more popular option for spatial interpolation not using external explanatory variables, other methods do exist and are used in popular geographic information system (GIS) software such as ArcGIS [17]. This section will briefly cover a selection of the more popular methods available. Unlike kriging or the later regression-based methods, this set of methods typically does not require model parameters to be trained by data. Instead, neighbouring observations are either used directly (in the case of methods like nearest neighbour), or estimated using a manually defined distance parameter (in inverse distance weighting).

Inverse distance weighting [2]. Much like kriging, Inverse Distance Weighting (IDW) relies on predicting $\hat{y}_{c_i}$ using a weighted sum of a sample of neighbours. Canonically all weights sum to 1. The IDW interpolation formula is thus fairly straightforward:

$$\hat{y}_{c_i} = \sum_{y^*_{c_j} \in \mathbf{Y} \cap \mathbf{Y}_{\varnothing}} \gamma_{c_j} \cdot y_{c_j} \tag{3.4}$$

As in the case of kriging in Equation 3.1, the core IDW formula is based on a weighted sum, with the main challenge being to find the appropriate weights. Unlike the variogram used in kriging, IDW assumes a linear discount over distance (which would also be a model option for the variogram), and thus defines its weights using inverse weighting $\frac{1}{d(c_j,c_i)}$. The final weight γ_{c_j} assigned to a location is given by dividing the inverse distance by the sum of all weights, thus guaranteeing that weights will sum to 1 and bypassing the problem of distance scaling. This leads to the following formula:

$$\gamma_{c_j} = \frac{\frac{1}{d(c_j,c_i)}}{\sum_{k \in \mathbf{Y} \cap \mathbf{Y}_{\varnothing}} \frac{1}{d(c_k,c_i)}} \tag{3.5}$$

Beside this basic form, more sophisticated versions of IDW do exist, such as allowing the user greater control through the use of a *power* parameter determining how much relative emphasis is placed on nearby measurements. Recent contributions include the new probabilistic weighting scheme proposed by Łukaszyk [18] and the approach proposed by Lu and Wong allowing the weighting parameter to vary, resulting in an adaptive version of IDW [19].

Nearest neighbour. A staple of many data-driven applications, nearest neighbour (NN) methods can also be used for spatial interpolation [20, 21].

In its simplest form, $\hat{y}_{c_i}$ would simply be equal to y_{c_j} of the nearest j where $y_{c_j} \neq \varnothing$. In the case of ties, one can take the average value of neighbouring locations $\frac{\sum_{c_j \in N(c_i)} y_{c_j}}{|N(c_i)|}$ as an estimate for $\hat{y}_{c_i}$, or select one y_{c_j} at random to represent $\hat{y}_{c_i}$. The method could be extended to k-NN allowing for the selection of k nearest neighbours, of which the average or most frequent option could be used.

Natural neighbour. Also known as Sibson interpolation (after its author) [22] or area-stealing interpolation, natural neighbour interpolation is yet another method based on weighted sums; in fact, its basic interpolation formula is identical to Equation 3.1. As with most weighted sum-based methods, natural neighbour interpolation is distinguished from other methods by how it computes the weights γ_{c_j}. The main idea of its approach is the use of Voronoi polygons [23], also known as Dirichlet regions or Thiessen polytopes [24]. Although the exact construction of these polygons is outside the scope of this project, the basic concept is fairly straightforward. For every known observation $c_k \in \mathbf{Y} : y_{c_k} \neq \varnothing$ a polygon P_{c_k} is created consisting of all points for which c_k is the nearest known observation. Next, when predicting the value of an unknown cell c_i, another Voronoi polygon P_{c_i} is created for this point, overlapping with the polygons of its neighbours $c_{j_1}, ..., c_{j_n}$. The weight γ_{c_j} of a neighbouring value y_{c_j} is then computed as the proportion of overlap between P_{c_i} and P_{c_j}. This can be conveniently formulated using set notation:

$$\gamma_{c_j} = \frac{|\{P_{c_i} \cup P_{c_j}\}|}{\sum_{c_k \in N(c_i)} |\{P_{c_i} \cup P_{c_k}\}|} \tag{3.6}$$

Recent work on natural neighbour interpolation has often focused on improvements to the efficiency of the algorithm, as the computation of Voronoi polygons can be an expensive operation. Examples of this type of work include the algorithm proposed by Ledoux and Gold [25] and the algorithm proposed by Park et al [26].

Splines. In spline interpolation, the goal is to interpolate the grid in such a manner as to make the resulting landscape (if plotted) as smooth as possible [27]. The analogy often used to explain the concept of this method is to imagine a sheet of stretchy rubber, anchored at known observations at the height of their values. The sheet would then form a smooth surface between the known observation anchor points, and the heights of the sheets at locations with unknown values would form the estimated value. Moving beyond the analogy, the basic interpolation formula for splines is:

$$\hat{y}_{c_i} = T(c_i) + \sum_{c_j \in N(c_i)} \gamma_{c_j} \cdot R(c_i, c_j) \tag{3.7}$$

While the exact mathematical details of $T(c_i)$ and $R(c_i, c_j)$ are beyond the scope of this work, we refer to the original publication of splines applied to interpolation by Harder and Desmarais [27] for an in-depth overview. The purpose of $T(c_i)$ is to define a "trend" for c_i, which allows the function to have

some momentum carried over beyond a point. Similarly, $R(c_i, c_j)$ controls the smoothness of the interpolated surface; in the analogy, one can imagine the contrast between a large rubber sheet with some room for smooth curvature of the surface, and a small rubber sheet pulled taut on all the anchor points, resulting in a very straight surface with sharp edges at anchor points. Finally, γ_{c_j} is found by solving a system of linear equations, similar to approaches like kriging in Equation 3.3.
Much like natural neighbour interpolation, much of the more recent work focuses on finding efficient solutions for large-scale datasets, such as the work by Hancock and Hutchinson [28]. However, as illustrated by the work of Sharifi et al [29], splines interpolation is also seeing applications in downscaling remote sensing data.

As in the case of kriging, methods like IDW and spline interpolation rely heavily on distance-based interactions between points, and do not account for the characteristics of intermediate locations. Conversely, nearest neighbour and natural neighbour interpolation only consider locations within a pre-defined neighbourhood, and thus do not allow for spatial interaction beyond a certain threshold.

3.2 Regression with spatial and explanatory variables

Let us now turn to common approaches for spatial regression. These methods, unlike purely spatial methods, can deploy external features or explanatory variables in order to predict a target variable, and tend to originate in the field of (geo-)statistics. Note that the set of methods covered in this section is by no means exhaustive, nor intended to be. For a more comprehensive overview of (geo-)statistical methods for spatial prediction tasks, including various regression and interpolation methods, we refer to the 2017 survey by Ziang [30].

Basic regression. When we mention basic or simple regression, we refer to a class of methods used for typical, non-spatial regression. In its simplest from, this can consist of linear regression; that is,

$$y_{c_i} = \mathbf{x}_{c_i} \cdot \theta + \epsilon_{c_i} \tag{3.8}$$

, where θ is a weight vector set by a training algorithm, $\mathbf{x}_{c_i}$ is the vector of features for location c_i, and ϵ_{c_i} represents the residuals or error of the model. For convenient notation, we add a bias feature of 1 to $\mathbf{x}_{c_i}$, allowing the corresponding weight from θ to represent the intercept of the linear model. Generalising to an entire dataset, the model can be compactly formulated using a vectorised notation:

$$\mathbf{y} = \mathbf{X} \cdot \theta + \epsilon \tag{3.9}$$

Basic regression is not limited to linear regression, however; any model or algorithm mapping a feature vector to a predicted target variable would fall under

this category. A large number of these models, and their training algorithms, are implemented in the popular Python library scikit-learn [31]. A selection of popular regression models implemented in scikit-learn include linear regression using Ordinary Least Squares (OLS) training [32], linear regression using Stochastic Gradient Descent (SGD) training [33] and Support Vector Machines (SVMs) [34].

Spatial Autoregressive Models [3]. Unlike simple regression, Spatial Autoregressive (SAR) models explicitly model spatial relationships. They do so by adding an extra term to a linear regression model representing the spatial autoregression, or autocorrelation over distance, of the target variable. Thus, the general (vectorised) form of SAR models is:

$$\mathbf{y} = \phi \cdot \mathbf{M} \cdot \mathbf{y} + \mathbf{X} \cdot \theta + \epsilon \tag{3.10}$$

In Equation 3.10, note that the right-hand side of the summation is simply equal to simple linear regression, multiplying features with their associated weights. The left-hand side, however, introduces new concepts. The first is ϕ, a scalar weight parameter similar to θ's elements scaling the vector produced by $\mathbf{M} \cdot \mathbf{y}$. Here $\mathbf{M}$ is a $H \times W$ matrix, modelling the spatial relationship between instances, which is an important design choice in SAR. In its simplest form, $\mathbf{M}$ would be a sparse matrix with a weight of 1 for instances that are direct neighbours, and 0 for instances that are not. In this case, common neighbourhood definitions include Rook (left, right, up, down) and Queen (left, right, up, down, diagonals) [35]. Another approach would be to assign a distance-based weight between all instances, rather than using a binary neighbourhood definition. In either case, $\mathbf{M}$ is multiplied by the vector of known values $\mathbf{y}$ to get a vector of weighted sums of neighbouring values. Of course, in practice not all true values will be known, in which case one might turn to inputation techniques such as mean substitution or hot-deck imputation [36]. Adding the left-hand (spatial) vector to the right-hand (regression) vector results in a final prediction vector. It may be of interest to note that the SAR model is also commonly referred to as the simultaneous autoregressive model (still SAR) or the spatial lag model.

The basic SAR model has seen various extensions in recent work. For example, Yang et al. proposed an extension of the basic SAR model with lagged explanatory variables and cross-variable lags [37]. Another, very recent extension to SAR would be the work by Fix et al aimed at being resilient to extremes in areal data [38].

Moving average models. Moving average (MA) models, more often used in time-series contexts [39], can also be used for spatial regression problems [4]. The key idea of this approach is to, rather than using the lagged target variable y_{c_j} from a neighbour c_j like in SAR models, use the lagged error ϵ_{c_j} instead. The intuition behind this approach is that the linear regression term models the general relationship between explanatory variables and the target variable, while the errors shift the location of predictions based on the geographical location. For example, one can consider that the errors of predicting house prices based

on the size of the house would differ per neighbourhood, even if the sizes are identical. If the errors of neighbours are high (predicted lower than the true value), we would expect the value at c_i to also be higher than the model would indicate.

We can now define a spatial variant of an MA model as:

$$\mathbf{y} = \phi \cdot \mathbf{M} \cdot \epsilon^{(1)} + \mathbf{X} \cdot \theta + \epsilon^{(2)} \tag{3.11}$$

, where $\epsilon^{(1)}$ consists of the errors used by the MA model to compute a new prediction, and $\epsilon^{(2)}$ represents the residuals of the MA model itself. As Equation 3.11 shows, the general form of SAR and MA models are very similar; in fact, the only difference is that, rather than using the vector Y of labels for all instances c_i, we multiply $\mathbf{M}$ by the vector of prediction errors $\epsilon^{(1)}$ as our spatial term. A drawback of this approach is that, much like SAR required imputation methods for handling unavailable y_{c_j} for neighbours c_j, so too do MA models suffer from the same limitation (as y_{c_j} is needed to compute ϵ_{c_j}). Furthermore, the definition of the spatial term in Equation 3.11 is recursive in nature: in order to ascertain the error of a neighbour c_j, we'd need to know the errors of c_j's neighbours in turn, which could propagate indefinitely. A possible strategy to deal with this in time-series applications would be to use the average value of the entire time-series as a prediction for the starting observation, thus enabling an initial error measurement. In spatial contexts, which are typically two-dimensional and multi-directional, a common approach is to use the "MA by AR" approach, as introduced by Durbin [39] and expanded to 2-D contexts by Francos and Friedlander [40]. In this approach the errors are computed by using a regression model for initial predictions, the errors of which are then used to train the "real" MA model. In the original work, the authors used a spatial autoregressive regression model for this purpose.

Autoregressive moving average models. SAR and MA models can be used in conjunction, which results in autoregressive moving average (ARMA) models [41] [42]. These models have been used in various time-series applications, but spatial variants of ARMA models have also been created [4]. In a spatial context, this would result in the following equation:

$$\mathbf{y} = \phi_1 \cdot \mathbf{M} \cdot \mathbf{y} + \phi_2 \cdot \mathbf{M} \cdot \epsilon^{(1)} + \mathbf{X} \cdot \theta + \epsilon^{(2)} \tag{3.12}$$

As shown in Equation 3.12, a combined SAR and MA model would simply add the spatial terms of both constituent models to the linear regression term. An extra advantage of combining these two models would be that, if accounting for missing initial errors by using those generated by the linear regression term, it allows the MA term to use a more sophisticated (spatial) method to determine $\epsilon^{(1)}_{c_i}$, as the prediction for any instance can be computed using Equation 3.10.
A recent early-access paper of particular relevance to the current COVID-19 pandemic is the paper by Qiu et al using ARMA to model a transmission network of infectious disease, specifically influenza [43].

Geographically weighted regression. Geographically weighted regression (GWR) is founded on the assumption of spatial non-stationarity [44]. As

such, the relationship between features and the target variable can be different for different subsets of the dataset. GWR is based on simple linear regression, but instead of learning one weight vector θ for the entire dataset, the weights instead form a surface of potential values. The values of the weights in θ are determined by the geographical location of the instance in question. Thus, the equation for this model is:

$$y_{c_i} = \theta_i^0 + \sum_{k=1}^{|\theta|} \theta_{c_i}^k \cdot \mathbf{x}_{c_i}^k + \epsilon_{c_i} \tag{3.13}$$

Equation 3.13 requires some explanation. The first term, $\theta_{c_i}^0$, is simply a bias term; here c_i represents the coordinates of the instance, and 0 is the index of the weight in θ. In the second term, $\sum_{k=1}^{|\theta|} \theta_{c_i}^k \cdot \mathbf{x}_{c_i}^k$, we take the weighted sum of all elements of feature vector $\mathbf{x}$, where the corresponding weight from θ is specific to location c_i. If we add a bias feature of 1 to $\mathbf{x}$, we can vectorise Equation 3.13 to:

$$\mathbf{y} = \theta_{c_i} \cdot \mathbf{x}_{c_i} + \epsilon \tag{3.14}$$

In Equation 3.14, we can see that the only difference with Equation 3.9 is that it uses a location-specific weight vector θ_{c_i}. The actual values of θ are computed using a modified version of OLS, in which a location-specific weight matrix M_{c_i} is used to multiply the features with. Thus, the location-specific weight vector θ_{c_i} is computed using:

$$\theta_{c_i} = (\mathbf{X}^T \cdot \mathbf{M}_{c_i} \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{M}_{c_i} \cdot \mathbf{y} \tag{3.15}$$

Equation 3.15 involves many matrix multiplications, and is in fact simply OLS weight fitting ($(\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$) with multiplications by $\mathbf{M}_{c_i}$ added in. The manner of determining the weight matrix $\mathbf{M}_{c_i}$ is referred to as a *kernel*, and can take various forms such as uniform, Gaussian, or exponential functions.

Finally, deep learning and **convolutional neural networks** (CNN) in particular have been used to great effect in many computer vision applications, such as interpolation-based image and video resolution upscaling [45, 46]. Such computer vision-based interpolation CNNs could also be applied to general spatial interpolation by replacing the vector of 3 RGB values of an image by the spatial feature vector $\mathbf{x}$. Moreover, in their 2020 publication, Hashimoto and Suto formulated a CNN architecture for the specific purpose of spatial interpolation [47].

As in the case of nearest neighbour and natural neighbour interpolation, this category of methods is limited by their use of a pre-defined local neighbourhood, dismissing information outside of the neighbourhood radius.

3.3 Summary and evaluation of related work

Though not without merits, every method listed in this chapter suffers from its own set of limitations. We will start with the purely spatial methods.

In the case of kriging, the pair-wise distance-based weight computation is limited, as the spatial characteristics of intermediate locations could be highly relevant. Kriging also typically assumes isotropy and stationarity – although some variants of kriging allow for the existence of trends in the data, weak stationarity is still assumed. Inverse distance weighting, though useful for a quick solution to spatial interpolation, suffers from all shortcomings of kriging, while being less flexible due to its use of purely distance-based weights rather than a variogram. Nearest neighbour interpolation is likewise simple to use, and therefore useful when quick rough solutions are needed, but suffers from its static, predefined neighbourhood definition, and will be very sensitive to extreme values in these neighbourhoods. Natural neighbour, while elegant in its approach of assigning weights to neighbours, is limited to purely local spatial effects of direct neighbours. Spline interpolation can be somewhat complex to apply and requires many parameters to be tuned effectively; moreover, its interpolation is rather distance-based and does not consider intermediate locations.

Moving on to regression methods capable of incorporating explanatory variables and spatial effects, we will start with basic regression. Given that basic regression uses purely local explanatory features to predict target variables directly, the lack of any spatial effects and the resulting waste of information from nearby known measurements would be its greatest weakness. SAR, MA and ARMA, while all subtly different, rely on rigidly defined neighbourhoods through the weight matrix $\mathbf{M}$. Although the weight matrix does allow for neighbourhoods to be flexibly assigned, they are rather rigid once set. In the case of MA and ARMA, determining the prediction errors $\epsilon^{(1)}$ is not trivial since it requires the prediction model to have already made predictions (in order to compute errors) prior to training the model. Though there are strategies to address this problem, all of these may introduce additional uncertainty to the model. Much like basic regression, geographically weighted regression disregards potentially valuable information from the known values of neighbouring cells. Although it does account for spatial effects by changing the regression weights based on the geographical location in question, its spatial effects are limited. Finally, convolutional neural networks will typically contain a large amount of trainable parameters, especially deep CNNs, which means training the models may require more time, as well as more data, than other types of models; this latter requirement may be particularly problematic in our case, where we typically only have a few hundred training examples available in a training set. Moreover, since the performance of neural networks can be heavily dependent on the architecture of the network, CNNs may require further resource-intense neural architecture search (NAS) to perform well. Moreover, all CNNs will rely on strictly defined neighbourhoods in their convolutional kernels, thus dismissing potentially relevant information from outside these neighbourhoods.

In the next chapter, we will introduce our proposed methods aimed at overcoming the limitations of the existing work.

Chapter 4

Proposed methods

In this section, we will first explain the reinforcement learning background of our proposed method, after which we will explain the general idea behind the use of these techniques for spatial interpolation. Next, we propose three variants of MRP interpolation, and explain the assumptions and properties of each variant. The creation of every variant was motivated by the need for variety in terms of applicability, as well as stationarity and isotropy assumptions.

4.1 Markov reward processeses

Markov reward processes (MRPs) are important to the field of reinforcement learning (RL), as they form the basis for the widely used Markov *decision* processes [48] (MDPs). The main idea of an MRP is that it models a *state space* S an *agent* (such as a robot or a video game character) can find itself in. States $s \in S$ could be described using various different characteristics, but a simple case conveniently close to our problem would be states described solely by a location. In every state s, the agent has a number of actions $a(s)$ available to it, and every action $a \in a(s)$ would transition the agent from state s to a successor state s', resulting in an immediate reward R_a for the agent for taking action a. Using this information, a *state value* $v(s)$ for state s could be computed for every state, by assigning the average reward of all actions $a \in a(s)$ as a state value $v(s)$. However, because it can be useful to look further into the future than only considering the immediate reward R_a, the estimated state value of the successor state s' should also be taken into account: an action a with a high R_a might take the agent to a successor state s' from which no further rewards can be acquired, which may not be desirable. Thus $v(s)$ is computed using the average *action value* $v(a) = R_a + v(s')$ instead, which adds the estimated value of s' as an expected future reward to R_a. At the same time, in many cases the relative certainty of receiving a reward early on may be more important than a less certain reward in the future. To model this, MRPs incorporate a *discount* parameter γ, which controls the degree to which an agent is near-

sighted (low γ, future rewards have a low weight) or far-sighted (high γ, future rewards have a high weight). Thus the action value $v(a)$ can be computed as $v(a) = R_a + \gamma \cdot v(s')$, leading to the following recursive definition of state value $v(s)$:

$$v(s) = \frac{\sum_{a \in a(s)} R_a + \gamma \cdot v(s')}{|a(s)|} \tag{4.1}$$

Of course, since *estimated* successor state values are used in this recursive formula rather than known values, Equation 4.1 must be iterated until an equilibrium of stable values is reached. Once the state values are known, an MDP would extend this by defining an optimal *policy* (actions to take given a state) based on the computed state values; however, we are only interested in the state values themselves, which is why we only use MRPs and not MDPs in our method.

4.2 Markov reward processes for spatial interpolation

In our problem definition, we are no longer dealing with an agent, but interpret a state s as a location c, an action a as a spatial interaction between locations, and the state value $v(s)$ as a predicted target value $\hat{y}$, which is the value we are ultimately interested in. In MRPs, the main value being computed is the state value $v(s)$, here $\hat{y}_c$, based on the expected future reward of a state (cell) c. As before, let us define a cell of interest $c_i = (h, w)$ and an arbitrary neighbouring cell $c_j = (h^*, w^*)$. For all $c_i \in \mathbf{Y}$, the estimated state value is computed using the summed (or average in canonical RL) estimated neighbouring values $\hat{y}_{c_j}$, weighted by a *discount* parameter γ, for all $c_j \in N(c_i)$ (note that, in the RL analogy, we are only using $v(s')$ and not R_a). Here $N(c_i)$ is the set of direct neighbours to c_1, where neighbourhoods can be defined flexibly, but a simple strategy would be to use shared borders (rook neighbourhoods). This leads to a recursive definition of $\hat{y}_{c_i}$:

$$\hat{y}_{c_i} = \sum_{c_j \in N(c_i)} \gamma \cdot \hat{y}_{c_j} \tag{4.2}$$

As a result of this recursive definition, all estimations depend on other estimations in turn, which means a single computation of Equation 4.2 per cell will not be sufficient to estimate y_{c_i}. Therefore, Equation 4.2 is iterated until an equilibrium is reached.

Intuitively, in spatial interpolation the weighted sum of neighbouring predictions represents a local "spatial lag", similar to SAR and many other spatial autocorrelation-based methods. However, the iterative use of recursive computations of $\hat{y}_{c_j}$, rather than known values y_{c_j}, sets MRPs apart from other methods by modelling a local computation of a recursive spatial lag. This recursivity allows every location c_i to implicitly influence every other location c_j, through all possible paths between c_i and c_j.

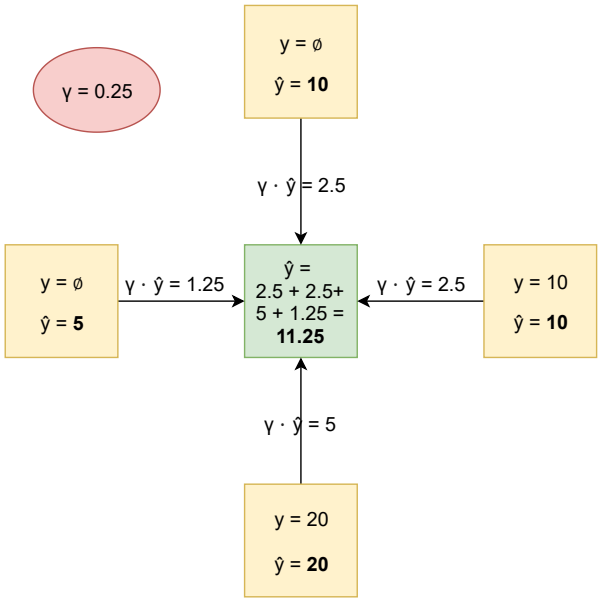


Figure 4.1: Simple example of an $\hat{y}$ update for a single cell (green). The estimated values of its neighbours (yellow) are discounted by $\gamma = 0.25$ and summed (optionally averaged) to determine a new $\hat{y}$. This process is repeated for every cell in the MRP in one iteration.

Figure 4.1 shows the basic idea of MRP interpolation. In the next sections, we will explain the three MRP variants in detail. The differences between these three variants are determined by how they compute the discount weights between c_j and c_i.

4.2.1 Static discount MRP.

The static discount MRP (SD-MRP) is the MRP interpolation variant closest to a canonical MRP as used in RL. If the true target value y_{c_i} at location c_i is already known, we use this as a static prediction for $\hat{y}_{c_i}$; if it is not known, we compute an estimated $\hat{y}_{c_i}$ using Equation 4.2. By iterating Equation 4.2 until equilibrium, we obtain a matrix with interpolated values of all cells, anchored by information from the cells c where the true values y_c are known.
An advantage of using this basic version of MRP interpolation would be that it requires no explanatory variables, which makes it very straightforward to apply in any scenario. Moreover, its non-reliance on additional data means it does not need a training phase, although it is advisable to tune γ either through the use of a separate training set or subsampled true target values of the dataset itself. The main disadvantage would be that it uses a static, global γ discount parameter, which means it assumes stationarity and isotropy.

4.2.2 Optimised MRP.

Given the assumptions of stationarity and isotropy of SD-MRP, we were interested in seeing whether we could create MRP interpolation methods robust to non-stationarity and anisotropy. To this end, we would like to use location-specific weights γ_{c_i,c_j} instead of a static weight γ. The challenge, then, would be to find a method to reliably find a weight vector Γ, consisting of individual weights γ_{c_i,c_j} for all neighbouring cells in $\mathbf{Y}$, such that the total interpolation error is minimal. If y_c^* is known for all $c \in \mathbf{Y}$, we are free to use black-box optimisation to directly find the set of optimal weights $\Gamma^* = \{\gamma^*_{c_j,c_i} : (c_i, c_j \in \mathbf{Y}) \wedge (c_j \in N(c_i))\}$ using interpolation loss on a selection of artificially hidden nodes as a loss function $\mathcal{L}$:

$$\Gamma^* \in \underset{\Gamma \in \mathbb{R}^{|\Gamma|}}{\operatorname{argmin}} \mathcal{L}(\mathbf{Y}, \Gamma) \tag{4.3}$$

However, if not all y_c^* are known, the weights to and from these cells cannot be optimised with any certainty. Moreover, these highly customised optimal weights will not be transferable to other regions, as they reflect the landscape of the region they were optimised for. As a result, optimised MRP (O-MRP) is not practically applicable in most situations: fully observable regions need no interpolation, and the weights cannot be re-used except in very specific circumstances (e.g., recurrent measurements that may be not be complete immediately, or the inference of the true values of noisy measurements).

4.2.3 Weight prediction MRP.

In an ideal case, we would use a method allowing us to exploit the benefits of optimal location-specific weights, regardless of where the MRP is used. This has led to the final version of MRP interpolation, in which we use the spatial feature vectors $\mathbf{x}_{c_i}$ and $\mathbf{x}_{c_j}$ as inputs to a weight prediction model to predict the optimal weight $\gamma^*_{c_i,c_j}$ from spatial data describing the locations (such as houses, shops and land use), rather than optimisation. If successful, this weight prediction MRP (WP-MRP) would combine the greater accuracy and resillience to non-stationarity and anisotropy from O-MRPs, and the broad applicability from SD-MRPs (provided explanatory variable data is available).

To train the weight prediction model, we run O-MRP on a training region to acquire an optimal weight configuration. Next, we create a feature vector $\mathbf{x}_{c_i,c_j}$ by combining $\mathbf{x}_{c_i}$ and $\mathbf{x}_{c_j}$ for every (c_i, c_j) pair in $\mathbf{Y}$. For every location pair, $\mathbf{x}_{c_i,c_j}$ is matched with the optimised weight $\gamma^*_{c_i,c_j}$, resulting in a tabular training set $\mathbf{X}$ with optimal weights Γ^* as a ground truth. We can thus define a regression model $\mathcal{M}_w(\mathbf{x})$, such that, if $\mathbf{Y}_N = \{(c_i, c_j) : (c_i, c_j \in \mathbf{Y}) \wedge (c_j \in N(c_i))\}$:

$$\mathcal{M}_w^* \in \underset{\mathcal{M}_w}{\operatorname{argmin}} \frac{1}{|\Gamma^*|} \sum_{(c_i,c_j) \in \mathbf{Y}_N} (\mathcal{M}_w(\mathbf{x}_{c_i,c_j}) - \gamma^*_{(c_i,c_j)}) \tag{4.4}$$

Here we train $\mathcal{M}_w(\mathbf{x})$ on Γ^* using any regression (machine learning) algorithm. The full pipeline of WP-MRP is outlined in Algorithm 1 (which assumes

Algorithm 1: Full pipeline for WP-MRP

Input: Training set matrices $\mathbf{X}^{train}$ and $\mathbf{Y}^{train}$, test set matrix $\mathbf{X}^{test}$ and $\mathbf{Y}^{test}$, maximum MRP iterations max_iter

Result: Interpolated matrix $\hat{\mathbf{Y}}$

1 $\Gamma^* = optimise\_weights(\mathbf{Y}^{train})$
2 $\mathcal{M}_w := fit\_model(X^{train}, \Gamma^*)$
3 $iter := 0$
4 **while** $iter < max\_iter$ **do**
5 **forall** $c_i \in \mathbf{Y}^{test}$ **do**
6 **if** $y_{c_i} \neq \varnothing$ **then**
7 $\hat{y}_{c_i} := 0$
8 **forall** $c_j \in N(c_i)$ **do**
9 $\hat{y}_{c_i} := \hat{y}_{c_i} + \mathcal{M}_w(\mathbf{x}^{test}_{c_j,c_i}) \cdot \hat{y}_{c_j}$
10 **end**
11 **else**
12 $\hat{y}_{c_i} := y_{c_i}$
13 **end**
14 **end**
15 $iter := iter + 1$
16 **end**
17 $\hat{\mathbf{Y}} := \hat{y}_c$ for $c \in \mathbf{Y}^{test}$
18 **return** $\hat{\mathbf{Y}}$

available functions for black-box optimisation and model fitting; any algorithm of choice for either task can be used). Line 1 runs O-MRP on a training set, and line 2 fits a weight prediction model to the optimal weights found by O-MRP. Lines 3-16 show the iterative updates of cells in $\mathbf{Y}$, and lines 17-18 create and return the predictions in the form of a grid $\hat{\mathbf{Y}}$.

Chapter 5

Experiments

We were interested in answering two main questions with our experiments:

- Can MRP interpolation achieve lower mean absolute prediction errors for various proportions of missing values compared to baseline methods?
- To what extent could these models be generalised and transferred across national and international regions?

In the case of the first question, while we were primarily interested in finding out which method would have the best performance, we wished to assure that any performance differences were not caused by a susceptibility to the proportion of known y^* values. This also led to the question of transferability: if data is scarce for one region, but plentiful for another, it could be useful to be able to take advantage of the large amounts of available data in the latter region to aid predictions in the former.

5.1 Baselines

Our selection of baselines was aimed at including competitive interpolation and regression methods used for spatial and geo-spatial modelling in practice. The selection we made consisted of:

- **Ordinary kriging** (OK) and **universal kriging** (UK), as implemented by the Python library PyKrige [49]. The general prediction model for kriging derived from Equation 3.1, with weight parameters solved using Equations 3.2 and 3.3, can be formulated as:

$$\hat{y}_{c_i} = \sum_{c_j \in N(c_i)} (\gamma_{c_j,c_i} \cdot y_{c_j}) \tag{5.1}$$

- **Non-spatial (basic) regression**, as described by Equations 3.8 and 3.9, but using auto-sklearn [50] to automatically select the best performing

conventional regression model. Since we are now explicitly estimating $\hat{y}$, our prediction model based on Equation 3.9 becomes:

$$\hat{\mathbf{y}} = \mathbf{X} \cdot \theta \tag{5.2}$$

- **Spatial autoregressive-** (SAR), **moving average-** (MA) and **autoregressive moving average** (ARMA) models, with prediction models based on Equations 3.10, 3.11 and 3.12, and using the "MA by AR" approach [4] for MA and ARMA. In the interest of fairness, we implemented these models manually, since this allowed us to use auto-sklearn as in the case of basic regression. The terms resulting from $\mathbf{m}_c \cdot \mathbf{y}$ and $\mathbf{m}_c \cdot \epsilon$ thus became additional features supplementing $\mathbf{x}$, and ϕ_1 and ϕ_2 became additional weights to train. The prediction models for estimating $\hat{y}$ for SAR (Equation 5.3), MA (Equation 5.4) and ARMA (Equation 5.5 are thus formulated as:

$$\hat{\mathbf{y}} = \phi \cdot \mathbf{M} \cdot \mathbf{y} + \mathbf{X} \cdot \theta \tag{5.3}$$

$$\hat{\mathbf{y}} = \phi \cdot \mathbf{M} \cdot \epsilon + \mathbf{X} \cdot \theta \tag{5.4}$$

$$\hat{\mathbf{y}} = \phi_1 \cdot \mathbf{M} \cdot \mathbf{y} + \phi_2 \cdot \mathbf{M} \cdot \epsilon + \mathbf{X} \cdot \theta \tag{5.5}$$

- **Convolutional neural networks** (CNN), using automated neural architecture search (NAS) implemented by auto-keras [51] for all training sets (50 trials, 1000 epochs).

5.2 Data

Our interpolation setup requires data of two types: a matrix $\mathbf{Y}$ of ground truth target values, and (depending on the method used) spatial features $\mathbf{x}_c$ for all $c \in \mathbf{Y}$. Consequently, our data collection process for evaluating this approach was also split into these two categories.

5.2.1 Ground truth values

For our ground truth data, we chiefly used a gridded dataset containing worldwide GDP estimates sourced from World Bank [6]. Although this dataset was gridded already, for greater flexibility we defined our own matrix with a fixed cell width and height of $0.02°$ (approximately 1 kilometer in the experimental regions we used), and averaged the values from the dataset to the appropriate cell in $\mathbf{Y}$. As the GDP dataset is fully available at every location in the world, it was very convenient for the evaluation of our method. However, as an indication of the generalisibility of our results, as well as an interesting test case for a potential application of our method, we also used a dataset on COVID-19 patient trajectories in South Korea [7]. We aggregated the locations in the trajectories as the total number of visits by infected patients per grid cell, a meaningful variable to be able to predict for virus containment policies. To evaluate our methods, we artificially hid a number of true values in $\mathbf{Y}$, allowing us to have

(a) True values (b) Non-hidden values (c) Interpolated grid

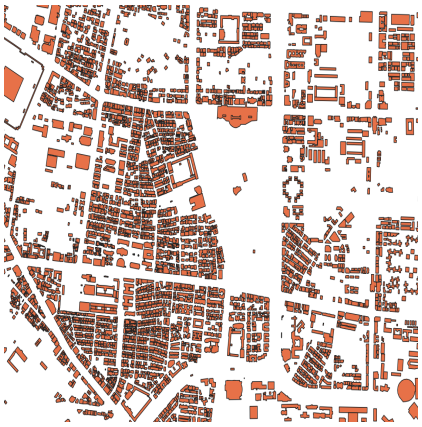
Figure 5.1: Taipei GDP grid, consisting of (a) ground truth values, (b) ground truth values where half were artificially hidden, and (c) an interpolated grid combining ground truth values for non-hidden cells with SD-MRP-predicted hidden values.

access to both y_c^* and any number of locations where $y_c = \varnothing$ (missing values). This process was controlled by the parameter p, representing the proportion of values in a dataset that is artificially hidden. An example of the ground truth preprocessing pipeline can be found in Figure 5.1. In Figure 5.1a, the original ground truth data is displayed, whereas Figure 5.1b shows the remaining true values after artificially hiding values (at $p = 0.5$). Finally, Figure 5.1c shows the resulting reconstructed original values by running SD-MRP on the values from Figure 5.1b.

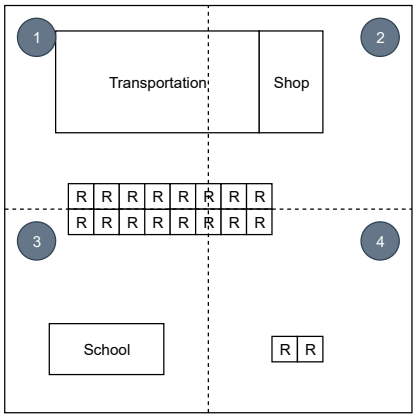
5.2.2 Spatial features

As spatial features for $\mathbf{x}$, we aggregated a selection of vector and point map data sourced from OpenStreetMap [52]. For all $c \in \mathbf{Y}$, every element in $\mathbf{x}_c$ represented the count of all objects in the map data corresponding to a certain *type*, such as apartments, houses and shops. An example of the map data can be found in Figure 5.2a, and the process of turning this map data into feature vectors for individual locations is illustrated in Figures 5.2b and 5.2c. There are various design choices available for preprocessing and feature selection on this type of data; a summary of our preprocessing options can be found in Table 5.1.

In accordance with the design philosophy of programming by optimisation (PbO) [53], we did not commit to any of these choices, and instead used automated algorithm configuration to select the best possible feature construction pipeline per method. To this end, we ran the Baysian optimisation-based algorithm configurator SMAC [54] for 36 hours for all methods on a machine with an Intel Xeon E5-2683 v4 CPU running at 2.10GHz. This ensured that we were evaluating different methods using (close to) optimal configurations of preprocessing options (as well as algorithm hyperparameters, if applicable) specific to those methods.



(a) Example map data for part of the Taipei city centre [52]



(b) Illustration of splitting up map data into four locations

Location	Transportation	Residential	Shop	School
1	0.8	5.5	0	0
2	0.2	2.5	1	0
3	0	5.5	0	1
4	0	4.5	0	0

(c) Features per location from Figure 5.2b from type counts

Figure 5.2: Preprocessing pipeline for spatial features.

5.2.3 Regions

As test regions we used the city Daegu in South Korea and Taipei in Taiwan. For both cities, we ran experiments for three *transferability conditions*: the available data in the city itself (*same-city*), transferred from another city in the same country (*same-country*), or transferred from a city in the other city's country (*different-country*). Seoul was used as an extra training city in South Korea, and Taichung was used for Taiwan.

5.3 Experimental setup

The basic form of our experiment consisted of executing 30 runs of every method, at proportion of hidden values p settings of 0.1, 0.3, 0.5, 0.7 and 0.9, for Daegu and Taipei. A single run of training and testing a model for a particular condition, when involving optimisation, took about a half hour to complete, whereas methods such as kriging and SD-MRP took less than a second or a few seconds to run, respectively. For relevant baselines as well as WP-MRP, we ran auto-sklearn [50] with a budget of 150 seconds to automatically select the best performing machine learning model and hyperparameters. For O-MRP and

Problem	Design option	Design option explanation
Missing values	drop	Drop all objects from a spatial dataset if they have no identifiable object type.
	replace	Add objects from a spatial dataset without an identifiable type as objects of the type "generic".
Feature selection	top n	Only use the n most frequently occurring object types to construct x.
	top n percent	Only use the top n percent of most frequent object types to construct x.
	top n variable	Only use the top n object types based on the variance of their frequency (computed beforehand) to construct x.
	frequency threshold	Only use object types of which their relative frequency exceeds a threshold to construct x.
	taxonomy	Map low-level object types to manually specified higher-level categories to construct x.
Feature normalisation	mean normalisation	Scale feature f using the mean, max and min value of f in the dataset: $f := \frac{f-mean(f)}{max(f)-min(f)}$
	unit length scaling	Scale feature vector x to unit length: $x := \frac{x}{\|x\|}$
	z-score normalisation	Scale feature f using the mean and standard deviation of f in the dataset: $f := \frac{f-mean(f)}{\sigma}$

Table 5.1: Overview of design options for the preprocessing pipeline of spatial features. Objects are individual shapes seen in Figure 5.2b(a), object types are the different functions objects may have, as illustrated in 5.2b(b) and (c), and a feature f is a single element of feature vector x. Missing values refers to the manner in which geographical objects in a spatial dataset with no identifiable type attribute are handled, feature selection refers to strategies to keep or discard features based on various selection criteria, and feature normalisation refers to various methods to normalise the locations and ranges of features.

WP-MRP training, we used the covariance matrix adaptation evolution strategy (CMA-ES) algorithm as implemented in the Python CMA package [55], using an iteration budget of 100 (early stopping allowed), to perform black-box optimisation.

We first ran this experimental setup for the same-city condition. Next, to test for transferability, we re-ran this experiment with same-country and different-country transferability conditions. In the case of the South Korean COVID-19 dataset, we were only able to test for national transferability. In the case of kriging, training regions were only used to select the best performing variogram model, and in the case of SD-MRP, they were only used to tune the global discount paramter γ. Finally, we should note that O-MRP can only be applied directly to the test region and therefore has no need of a training set; moreover, as explained in Chapter 4 O-MRPs are mainly used as a theoretical lower bound for errors that could be achieved with MRP methods.

For our performance comparisons we performed normality tests on the 30-run samples for all conditions. If normally distributed, we ran one-tailed t-tests to establish statistical significance for performance improvements between methods. If one of the samples was not normally distributed, we ran a Wilcoxon signed-rank test instead, as it can function as an alternative to the t-test when

comparing the mean of two samples for which a normal distribution cannot be assumed [56]. For all tests, we set $\alpha = 0.05$. Methods were ranked based on the number of other methods they significantly outperformed, with a special emphasis placed on the top-3 methods and the worst-performing method.

Chapter 6

Results

All results of our experiments can be found in the thesis Appendix. In this chapter, we will highlight a number of specific results in Tables 6.1, 6.2 and 6.3. We will first directly compare the performance of all different methods, particularly which methods perform best under which conditions. Next, we will interpret the results on the transferability potential of different methods.

Our experimental results have generally been favourable to our proposed methods. In 23 out of 40 conditions SD-MRP outperformed (ties allowed) all baselines. This number was 15 for WP-MRP and 35 for O-MRP. In total, MRPs accounted for 80 out of 136 top-3 performances. In a number of conditions, such as GDP interpolation for Daegu (same-city training), SD-MRP or WP-MRP achieved lower mean errors than a higher-ranked baseline. We believe this is due to either the higher standard deviation, or the lack of a normal distribution, rendering the lower mean error not statistically significant. However, this does imply that training an MRP multiple times and selecting the best performing model may give better results than the baselines.

In general, in conditions where the baselines outperformed MRP methods, the MRP methods typically still achieved fairly good results. In fact, apart from 3 conditions for WP-MRP and 1 condition for SD-MRP, MRP methods were never the worst performing method. This was a positive outcome for MRPs compared to the baselines, which were not consistently competitive. The two kriging methods (with negligible performance differences between them) were the worst-performing methods most often, at 15 out of 40 conditions (particularly in the COVID-19 dataset), but were also the best in 5 conditions and among the top 3 in 19 conditions. When tested on Taipei and trained on Taichung or Taipei itself, kriging performed particularly well, at times surpassing even O-MRP. We speculate that the relatively good performance of kriging for Taichung-trained conditions may partially be caused by the non-representativeness of Taichung as a training region, as – unlike the other cities which grew naturally over time – Taichung was developed as a highly planned city. Since kriging was applied directly to the test region, with only the type of variogram model being tuned by the training set, it may not have suffered from this effect.

Method	Test data: Daegu GDP									
	Training region: Seoul									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$10.129^{(\otimes)}$	0.767	$10.03^{(\otimes)}$	0.664	$10.27^{(\otimes)}$	0.68	$9.808^{(\otimes)}$	0.595	$10.19^{(\otimes)}$	0.749
UK	$10.166^{(\otimes)}$	0.615	$10.3^{(\otimes)}$	0.618	$9.971^{(\otimes)}$	0.647	$10.05^{(\otimes)}$	0.701	$10.21^{(\otimes)}$	0.947
Basic	3.7723	1.457	3.757	0.553	3.833	0.413	3.842	0.247	3.867	0.111
SAR	4.7495	1.606	3.805	1.068	4.203	1.422	4.812	1.577	4.331	1.369
MA	5.1548	2.078	6.567	4.967	6.085	2.404	4.847	1.836	5.334	3.829
ARMA	4.5039	1.181	4.675	1.319	5.062	1.326	5.412	1.763	5.164	1.559
CNN	3.0891	0.165	$3.091^{(3)}$	0.165	$3.089^{(3)}$	0.165	3.089	0.165	$3.086^{(3)}$	0.164
SD-MRP	$2.1037^{(2)}$	1.202	$2.177^{(2)}$	0.577	$2.085^{(2)}$	0.347	2.616(2)	0.307	$2.911^{(2)}$	0.338
WP-MRP	$2.7458^{(3)}$	1.534	$3.02^{(3)}$	0.855	$3.162^{(3)}$	0.651	$2.901^{(3)}$	0.376	3.017	0.228
O-MRP	$\mathbf{0.5288}^{(1)}$	0.327	$\mathbf{0.982}^{(1)}$	0.227	$\mathbf{1.444}^{(1)}$	0.252	$\mathbf{2.055}^{(1)}$	0.296	$\mathbf{2.346}^{(1)}$	0.352

Table 6.1: Results per setting of p per method for GDP interpolation, trained on Seoul and tested on Daegu. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Taipei GDP									
	Training region: Taichung									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$2.6273^{(2,3)}$	0.276	$2.585^{(1,2,3)}$	0.269	$\mathbf{2.521}^{(1)}$	0.233	$\mathbf{2.702}^{(1,2)}$	0.234	$2.684^{(1,2)}$	0.252
UK	$2.6603^{(2,3)}$	0.24	$2.628^{(1,2,3)}$	0.267	$2.698^{(2)}$	0.382	$2.707^{(1,2)}$	0.284	$\mathbf{2.62}^{(1,2)}$	0.223
Basic	5.1524	1.705	5.173	0.895	5.245	0.753	5.148	0.381	5.08	0.24
SAR	6.2848	2.396	6.917	4.324	$9.434^{(\otimes)}$	13.95	$6.023^{(\otimes)}$	0.381	$6.065^{(\otimes)}$	0.852
MA	$7.5805^{(\otimes)}$	5.457	$9.742^{(\otimes)}$	11.33	$7.996^{(\otimes)}$	4.825	$9.533^{(\otimes)}$	15.58	13.05	17.24
ARMA	5.7533	0.884	5.822	0.898	5.852	1.067	$6.041^{(\otimes)}$	0.843	6.008	1.072
CNN	6.1006	2.251	5.914	1.26	5.824	1.454	5.683	1.359	6.137	2.114
SD-MRP	3.9926	1.451	4.441	0.784	4.375	0.409	4.516	0.471	$4.913^{(3)}$	0.442
WP-MRP	5.4675	2.538	5.934	1.924	5.284	0.962	5.343	0.718	5.183	0.319
O-MRP	$\mathbf{1.356}^{(1)}$	0.49	$\mathbf{2.472}^{(1,2,3)}$	0.573	$3.513^{(3)}$	0.531	$4.264^{(3)}$	0.477	$4.909^{(3)}$	0.485

Table 6.2: Results per setting of p per method for GDP interpolation, trained on Taichung and tested on Taipei, using the same notation as in Table 6.1.

Method	Test data: Daegu COVID-19									
	Training region: Daegu									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$12.365^{(\otimes)}$	1.342	$13.64^{(\otimes)}$	1.475	$13.04^{(\otimes)}$	1.513	$13.17^{(\otimes)}$	1.341	$13.4^{(\otimes)}$	1.202
UK	$12.546^{(\otimes)}$	1.511	$13.09^{(\otimes)}$	1.415	$13.18^{(\otimes)}$	1.313	$13.13^{(\otimes)}$	1.55	$13.48^{(\otimes)}$	1.444
Basic	2.5102	0.234	2.56	0.461	2.435	0.606	2.492	1.006	2.279	1.625
SAR	$1.1801^{(2)}$	0.836	$1.638^{(2,3)}$	0.679	$1.63^{(3)}$	0.629	2.437	2.029	2.402	1.632
MA	1.8745	0.395	2.239	1.085	3.912	7.824	3.807	3.683	$26.69^{(\otimes)}$	83
ARMA	2.3224	1.26	2.355	0.901	2.31	0.675	2.682	1.478	3.453	2.227
CNN	$1.6711^{(3)}$	0.247	$1.794^{(2,3)}$	0.376	1.9	0.662	1.879	1.083	$1.819^{(2)}$	1.97
SD-MRP	2.1936	2.399	1.925	1.014	$1.61^{(2)}$	0.699	$1.708^{(3)}$	0.397	$1.652^{(3)}$	0.248
WP-MRP	2.0035	2.427	$1.948^{(2,3)}$	2.024	$1.632^{(3)}$	0.679	$1.585^{(2)}$	0.452	1.67	0.245
O-MRP	$\mathbf{0.2647}^{(1)}$	0.311	$\mathbf{0.611}^{(1)}$	0.311	$\mathbf{0.757}^{(1)}$	0.366	$\mathbf{1.008}^{(1)}$	0.313	$\mathbf{1.197}^{(1)}$	0.253

Table 6.3: Results per setting of p per method for COVID-19 interpolation, trained on Daegu and tested on Daegu, using the same notation as in Table 6.1.

Although WP-MRP did not compare as well to the baselines as SD-MRP, its performance was generally comparable to, if not slightly better than CNN's performance, despite being a much lighter model. It does appear, however, that WP-MRP rarely suffers from invalid results (outliers on an order between 10^{10} and even 10^{100}) in its results; closer inspection showed that in these cases, the optimisation algorithm failed to converge for training on Taichung, and we ran our analyses excluding these runs. Given the results, though it outperformed the best-performing baseline in 15 out of 40 runs, we believe that further work is needed to realise the full potential of WP-MRP.

When it comes to transferability, only a very mild reduction of accuracy can be observed for more distant training regions for WP-MRP. However, MRP interpolation generally appears not to be heavily affected by the distance between training and testing regions, which we consider a positive result. MRPs do appear to be more sensitive to the setting of p than most baselines, as can be seen in Figures 6.1 and 6.2.

MRP methods were more prominently effective for the COVID-19 dataset than the GDP dataset, implying there may be some variance in performance based on the dataset. We speculate that the irregularity of the COVID-19 dataset, containing localised clusters that do not work well with distance-based methods such as kriging, was the cause for this. Apart from this, the performance rankings between the GDP and COVID-19 datasets were quite similar. This is a good outcome, not only because it is an encouraging sign that our results may generalise to other datasets as well, but also because the effective interpolation of virus exposure risk might be a very useful tool in virus containment efforts.

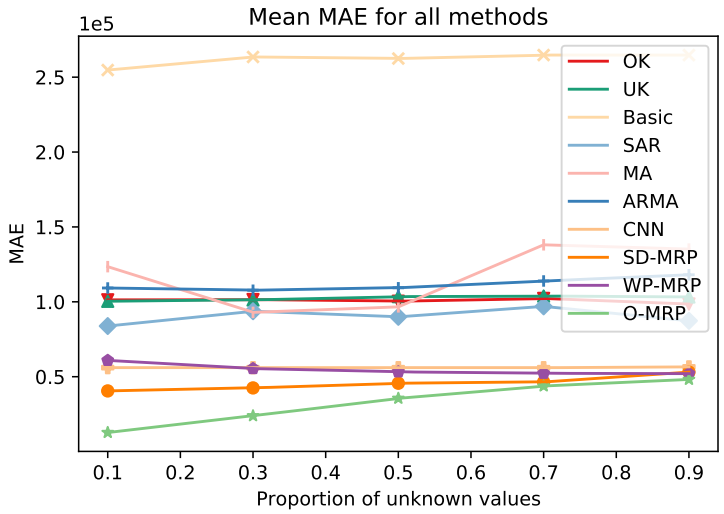


Figure 6.1: Performance of all methods as a function of p for GDP interpolation trained on Seoul and tested on Daegu.

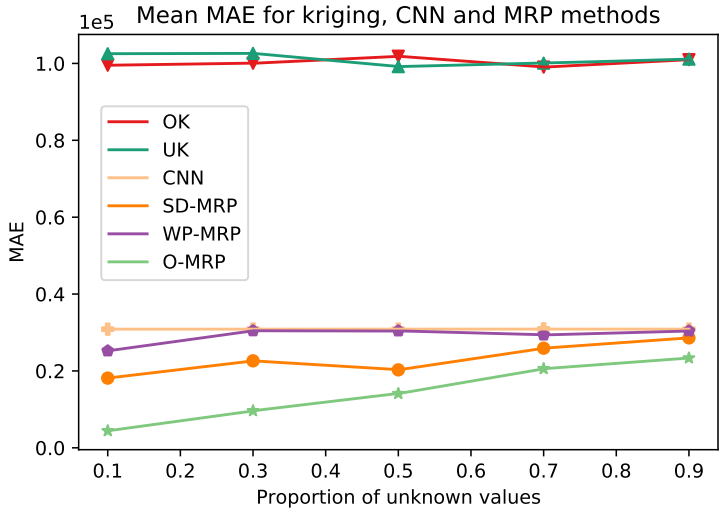


Figure 6.2: Performance MRP methods compared to the most competitive baselines (kriging and CNN) for GDP interpolation trained on Seoul and tested on Daegu.

Chapter 7

Conclusion and future work

In this thesis, we have explored the use of Markov reward processes (MRPs) to solve spatial interpolation problems. These MRPs were primarily aimed at addressing the limitation of existing methods, which suffer from having to trade-off between local and global spatial effects. We introduced three different variants of MRP interpolation (SD-MRP, O-MRP and WP-MRP), offering different combinations of practical applicability and robustness to non-stationarity and anisotropy. We tested these methods on a GDP dataset for two cities in South Korea and Taiwan, with same-city, same-country and different-country transferability conditions. In our experiments, SD-MRP outperformed all baselines in 23 out of 40 conditions, and O-MRP did in 35 out of 40 conditions. The WP-MRP was unable to predict optimised weights with sufficient accuracy to outperform SD-MRP, but still remained competitive in most cases and outperformed all baselines in 15 conditions.

In future work, it would be interesting to further study WP-MRP; perhaps different models or different explanatory variables could be more effective at predicting optimal weights. Of course, other variants of MRP interpolation, beyond the three proposed in this thesis, also offer a promising avenue for future work.

Overall, our results have clearly shown the potential of MRP-based spatial interpolation, and we thus hope to see it studied further and to see it broadly applied in practice.

Ethical statement

The COVID-19 dataset provided by DACON is quite rare, as it contains individual trajectories of infected COVID patients prior to their diagnosis. Although such a dataset could clearly be of great use to the South Korean pandemic response, as well as that of other countries looking to learn from this data, there are obvious privacy concerns to raise for this dataset. We believe our use of this dataset to have been justified by the strict standards of responsible AI one should expect in the AI research community, particularly as we transformed the dataset from individual trajectories (which could lead to personally identifiable information) to location-based aggregated counts of visits without considering any temporal aspects. While this format does not completely eliminate individual privacy concerns, we believe it is sufficiently addressed in this manner for the use of this dataset to be admissible in a thesis. One should also be wary of the use of spatial interpolation on this dataset, as well as others, for the purposes of local and regional discrimination. While data-driven decision making certainly has its merits, one should not forget that models are only models making predictions, and real-world policy being made on the basis of these predictions, particularly policy negatively affecting individuals or groups of individuals, should always be viewed with great caution and restraint.
Finally, while a discussion could (and should) certainly be had on the merits of using data that is available anyway for the improvement of the pandemic response around the world, weighed against the wariness against the use of data that one may be principally opposed to being collected in the first place, one should not forget the debt of gratitude owed to the South Koreans providing their data if it is indeed used for the improvement of the global pandemic response.

Bibliography

[1] Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

[2] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, 1968.

[3] Luc Anselin. Spatial econometrics: methods and models (vol. 4). *Studies in Operational Regional Science. Dordrecht: Springer Netherlands*, 1988.

[4] RP Haining. The moving average model for spatial interaction. *Transactions of the Institute of British Geographers*, pages 202–225, 1978.

[5] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[6] World Bank DECRG. GIS processing, 2010.

[7] DACON. Corona Data Visualization AI Contest, 2020.

[8] Daniel G Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[9] Mohamed A Bouhlel and Joaquim RRA Martins. Gradient-enhanced kriging for high-dimensional problems. *Engineering with Computers*, 35(1):157–173, 2019.

[10] Koya Sato, Kei Inage, and Takeo Fujii. On the performance of neural network residual kriging in radio environment mapping. *IEEE Access*, 7:94557–94568, 2019.

[11] Noel Cressie. Spatial prediction and ordinary kriging. *Mathematical geology*, 20(4):405–421, 1988.

[12] Georges Matheron. Universal kriging. In *Matheron's Theory of Regionalised Variables*. Oxford University Press, 1969.

[13] Andre G Journel and ME Rossi. When do we need a trend model in kriging? *Mathematical Geology*, 21(7):715–739, 1989.

[14] Richard Webster and Alexander B McBratney. Mapping soil fertility at broom's barn by simple kriging. *Journal of the Science of Food and Agriculture*, 38(2):97–115, 1987.

[15] Georges Matheron. *Le krigeage universel*, volume 1. École nationale supérieure des mines de Paris Paris, 1969.

[16] Alain Marechal. Kriging seismic data in presence of faults. In *Geostatistics for natural resources characterization*, pages 271–294. Springer, 1984.

[17] Esri. ArcGIS. `https://www.arcgis.com/index.html`, 2020. Accessed: 29-08-2020.

[18] Szymon Łukaszyk. A new concept of probability metric and its applications in approximation of scattered data sets. *Computational Mechanics*, 33(4):299–304, 2004.

[19] George Y Lu and David W Wong. An adaptive inverse-distance weighting spatial interpolation technique. *Computers & geosciences*, 34(9):1044–1055, 2008.

[20] Rukundo Olivier and Cao Hanqiang. Nearest neighbor value interpolation. *Int. J. Adv. Comput. Sci. Appl*, 3(4):25–30, 2012.

[21] Yunsheng Song, Jiye Liang, Jing Lu, and Xingwang Zhao. An efficient instance selection algorithm for k nearest neighbor regression. *Neurocomputing*, 251:26–34, 2017.

[22] Robin Sibson. A brief description of natural neighbour interpolation. *Interpreting multivariate data*, 1981.

[23] Georges Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1908(134):198–287, 1908.

[24] Peter A Burrough, Rachael McDonnell, Rachael A McDonnell, and Christopher D Lloyd. *Principles of geographical information systems*. Oxford university press, 2015.

[25] Hugo Ledoux and Christopher Gold. An efficient natural neighbour interpolation algorithm for geoscientific modelling. In *Developments in spatial data handling*, pages 97–108. Springer, 2005.

[26] Sung W Park, Lars Linsen, Oliver Kreylos, John D Owens, and Bernd Hamann. Discrete sibson interpolation. *IEEE Transactions on visualization and computer graphics*, 12(2):243–253, 2006.

[27] Robert L Harder and Robert N Desmarais. Interpolation using surface splines. *Journal of aircraft*, 9(2):189–191, 1972.

[28] Penelope A Hancock and MF Hutchinson. Spatial interpolation of large climate data sets using bivariate thin plate smoothing splines. *Environmental Modelling & Software*, 21(12):1684–1694, 2006.

[29] Ehsan Sharifi, B Saghafian, and R Steinacker. Downscaling satellite precipitation estimates with multiple linear regression, artificial neural networks, and spline interpolation techniques. *Journal of Geophysical Research: Atmospheres*, 124(2):789–805, 2019.

[30] Zhe Jiang. A survey on spatial prediction methods. *IEEE Transactions on Knowledge and Data Engineering*, 31(9):1645–1664, 2018.

[31] scikit learn. Scikit-learn. `https://scikit-learn.org/stable/`, 2020. Accessed: 29-08-2020.

[32] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*, volume 821. John Wiley & Sons, 2012.

[33] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[34] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[35] W Daniel Kissling and Gudrun Carl. Spatial autocorrelation and the selection of simultaneous autoregressive models. *Global Ecology and Biogeography*, 17(1):59–71, 2008.

[36] Rebecca R Andridge and Roderick JA Little. A review of hot deck imputation for survey non-response. *International statistical review*, 78(1):40–64, 2010.

[37] Kai Yang and Lung-fei Lee. Identification and qml estimation of multivariate and simultaneous equations spatial autoregressive models. *Journal of Econometrics*, 196(1):196–214, 2017.

[38] Miranda J Fix, Daniel S Cooley, and Emeric Thibaud. Simultaneous autoregressive models for spatial extremes. *Environmetrics*, 2020.

[39] James Durbin. Efficient estimation of parameters in moving-average models. *Biometrika*, 46(3/4):306–316, 1959.

[40] Joseph M Francos and Benjamin Friedlander. Parameter estimation of two-dimensional moving average random fields. *IEEE transactions on signal processing*, 46(8):2157–2165, 1998.

[41] P Whittle. Hypothesis testing in time series analysis, 1951. *Almquist and Wiksell, Upssala*, 1951.

[42] Peter Whittle. Prediction and regulation by linear least-square methods. 1963.

[43] Jianqing Qiu, Huimin Wang, Tao Zhang, and Changhong Yang. Spatial transmission network construction of influenza-like illness using dynamic bayesian network and vector-autoregressive moving average model. 2020.

[44] A Stewart Fotheringham, Martin E Charlton, and Chris Brunsdon. Geographically weighted regression: a natural evolution of the expansion method for spatial data analysis. *Environment and planning A*, 30(11):1905–1927, 1998.

[45] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[46] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[47] Riku Hashimoto and Katsuya Suto. Sicnn: Spatial interpolation with convolutional neural networks for radio environment mapping. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 167–170. IEEE, 2020.

[48] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[49] Benjamin S. Murphy. pykrige. `https://pypi.org/project/PyKrige/`, 2020. Accessed: 31-09-2020.

[50] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*, pages 113–134. Springer, Cham, 2019.

[51] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956, 2019.

[52] OpenStreetMap. OpenStreetMap. `https://www.openstreetmap.org/`, 2019. Accessed: 27-12-2019.

[53] Holger H Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, 2012.

[54] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[55] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

[56] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

Chapter 8

Appendix

This appendix contains full results that would have taken up too much space to include in the main thesis. The full results given here consist of all conditions, of which 3 examples were given in the main thesis. As in the thesis, we performed significance tests to rank all methods per condition, and highlighted our results based on this.

8.1 Tables

Tables of our full results are given below in Tables 8.1-8.8 starting on the next page. Overall, these extended results show the same patterns as those discussed in the thesis.

Method	Test data: Taipei GDP									
	Training region: Taipei									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$2.6301^{(2,3)}$	0.233	$2.66^{(2,3)}$	0.272	$2.629^{(2,3)}$	0.244	$2.672^{(2,3)}$	0.328	$\mathbf{2.615}^{(1)}$	0.232
UK	$2.6704^{(2,3)}$	0.236	$2.698^{(2,3)}$	0.267	$2.593^{(2,3)}$	0.223	$2.726^{(2,3)}$	0.278	$2.727^{(2)}$	0.249
Basic	3.9025	0.108	3.898	0.213	3.819	0.355	3.801	0.521	3.662	0.857
SAR	3.5036	0.532	4.041	0.935	4.121	0.488	4.494	0.642	$6.041^{(\otimes)}$	1.659
MA	3.5539	0.556	3.625	0.551	3.691	0.505	3.876	0.722	4.671	1.827
ARMA	3.8732	0.569	3.973	0.669	4.089	0.728	4.252	1.002	4.391	0.689
CNN	$5.4808^{(\otimes)}$	0.378	$5.427^{(\otimes)}$	0.567	$5.461^{(\otimes)}$	0.811	$5.349^{(\otimes)}$	1.039	5.337	2.259
SD-MRP	4.2816	1.981	4.863	1.119	4.625	0.736	$4.962^{(\otimes)}$	0.623	5.188	0.329
WP-MRP	4.9688	2.243	$5.556^{(\otimes)}$	1.077	$5.382^{(\otimes)}$	0.723	$5.223^{(\otimes)}$	0.523	5.296	0.263
O-MRP	$\mathbf{0.9534}^{(1)}$	0.514	$\mathbf{1.46}^{(1)}$	0.334	$\mathbf{1.753}^{(1)}$	0.343	$\mathbf{2.205}^{(1)}$	0.311	$3.021^{(3)}$	0.746

Table 8.1: Results per setting of p per method for GDP interpolation, trained on Taipei and tested on Taipei. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Taipei GDP									
	Training region: Seoul									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	10.131	0.589	10.15	0.451	10.04	0.583	10.23	0.548	9.853	0.465
UK	10.038	0.835	10.13	0.591	10.35	0.659	10.39	0.531	10.33	0.74
Basic	$25.474^{(\otimes)}$	4.757	$26.35^{(\otimes)}$	1.213	$26.24^{(\otimes)}$	0.704	$26.48^{(\otimes)}$	0.602	$26.47^{(\otimes)}$	0.306
SAR	8.3827	1.719	9.37	4.045	8.996	1.585	9.695	4.467	8.731	1.298
MA	12.348	8.498	9.295	3.965	9.662	5.201	13.81	14.06	13.52	12.25
ARMA	10.922	5.503	10.77	6.012	10.94	7.303	11.39	7.272	11.8	4.77
CNN	$5.6052^{(3)}$	0.242	$5.601^{(3)}$	0.243	5.6	0.245	5.6	0.245	$5.655^{(3)}$	0.238
SD-MRP	$4.0505^{(2)}$	1.348	$4.251^{(2)}$	0.725	$4.564^{(2)}$	0.508	$4.646^{(1,2)}$	0.606	$5.308^{(2)}$	0.557
WP-MRP	$6.0888^{(3)}$	2.464	5.542	1.057	$5.322^{(3)}$	0.781	$5.226^{(3)}$	0.532	5.205	0.297
O-MRP	$1.2719^{(1)}$	0.708	$2.388^{(1)}$	0.437	$3.554^{(1)}$	0.604	$4.378^{(1,2)}$	0.557	$4.818^{(1)}$	0.472

Table 8.2: Results per setting of p per method for GDP interpolation, trained on Seoul and tested on Taipei. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Taipei GDP									
	Training region: Taichung									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$2.6273^{(2,3)}$	0.276	$2.585^{(1,2,3)}$	0.269	$\mathbf{2.521}^{(1)}$	0.233	$\mathbf{2.702}^{(1,2)}$	0.234	$2.684^{(1,2)}$	0.252
UK	$2.6603^{(2,3)}$	0.24	$2.628^{(1,2,3)}$	0.267	$2.698^{(2)}$	0.382	$2.707^{(1,2)}$	0.284	$\mathbf{2.62}^{(1,2)}$	0.223
Basic	5.1524	1.705	5.173	0.895	5.245	0.753	5.148	0.381	5.08	0.24
SAR	6.2848	2.396	6.917	4.324	$9.434^{(\otimes)}$	13.95	$6.023^{(\otimes)}$	0.381	$6.065^{(\otimes)}$	0.852
MA	$7.5805^{(\otimes)}$	5.457	$9.742^{(\otimes)}$	11.33	$7.996^{(\otimes)}$	4.825	$9.533^{(\otimes)}$	15.58	13.05	17.24
ARMA	5.7533	0.884	5.822	0.898	5.852	1.067	$6.041^{(\otimes)}$	0.843	6.008	1.072
CNN	6.1006	2.251	5.914	1.26	5.824	1.454	5.683	1.359	6.137	2.114
SD-MRP	3.9926	1.451	4.441	0.784	4.375	0.409	4.516	0.471	$4.913^{(3)}$	0.442
WP-MRP	5.4675	2.538	5.934	1.924	5.284	0.962	5.343	0.718	5.183	0.319
O-MRP	$\mathbf{1.356}^{(1)}$	0.49	$\mathbf{2.472}^{(1,2,3)}$	0.573	$3.513^{(3)}$	0.531	$4.264^{(3)}$	0.477	$4.909^{(3)}$	0.485

Table 8.3: Results per setting of p per method for GDP interpolation, trained on Taichung and tested on Taipei. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Daegu GDP									
	Training region: Daegu									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$2.5744^{(3)}$	0.277	2.676	0.332	$2.69^{(3)}$	0.33	$2.611^{(3)}$	0.255	$2.733^{(3)}$	0.264
UK	2.7399	0.253	2.629	0.29	$2.652^{(3)}$	0.269	2.726	0.261	$2.634^{(3)}$	0.242
Basic	$0.8961^{(2)}$	0.054	$0.905^{(2)}$	0.084	$0.889^{(2)}$	0.174	$\mathbf{0.901}^{(1,2)}$	0.269	$\mathbf{1.055}^{(1)}$	0.738
SAR	2.4277	0.97	2.962	1.213	2.894	0.561	4.674	9.476	3.87	1.635
MA	$3.5353^{(\otimes)}$	0.476	$3.584^{(\otimes)}$	0.738	$3.793^{(\otimes)}$	0.659	$4.127^{(\otimes)}$	1.524	$8.519^{(\otimes)}$	12.18
ARMA	$3.7675^{(\otimes)}$	0.985	$3.898^{(\otimes)}$	1.103	$4.125^{(\otimes)}$	1.065	$4.205^{(\otimes)}$	1.091	4.829	5.114
CNN	2.9861	0.201	2.967	0.336	3.186	1.789	3.841	2.779	3.605	1.206
SD-MRP	2.4405	1.324	$2.198^{(3)}$	0.793	$2.459^{(3)}$	0.658	2.59	0.433	$2.745^{(3)}$	0.346
WP-MRP	2.4715	1.359	3.005	0.951	3.03	0.62	2.861	0.297	2.939	0.218
O-MRP	$\mathbf{0.4538}^{(1)}$	0.257	$\mathbf{0.652}^{(1)}$	0.171	$\mathbf{0.78}^{(1)}$	0.137	$0.977^{(1,2)}$	0.19	$1.377^{(2)}$	0.45

Table 8.4: Results per setting of p per method for GDP interpolation, trained on Daegu and tested on Daegu. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Daegu GDP									
	Training region: Seoul									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$10.129^{(\otimes)}$	0.767	$10.03^{(\otimes)}$	0.664	$10.27^{(\otimes)}$	0.68	$9.808^{(\otimes)}$	0.595	$10.19^{(\otimes)}$	0.749
UK	$10.166^{(\otimes)}$	0.615	$10.3^{(\otimes)}$	0.618	$9.971^{(\otimes)}$	0.647	$10.05^{(\otimes)}$	0.701	$10.21^{(\otimes)}$	0.947
Basic	3.7723	1.457	3.757	0.553	3.833	0.413	3.842	0.247	3.867	0.111
SAR	4.7495	1.606	3.805	1.068	4.203	1.422	4.812	1.577	4.331	1.369
MA	5.1548	2.078	6.567	4.967	6.085	2.404	4.847	1.836	5.334	3.829
ARMA	4.5039	1.181	4.675	1.319	5.062	1.326	5.412	1.763	5.164	1.559
CNN	3.0891	0.165	$3.091^{(3)}$	0.165	$3.089^{(3)}$	0.165	3.089	0.165	$3.086^{(3)}$	0.164
SD-MRP	$2.1037^{(2)}$	1.202	$2.177^{(2)}$	0.577	$2.085^{(2)}$	0.347	2.616(2)	0.307	$2.911^{(2)}$	0.338
WP-MRP	$2.7458^{(3)}$	1.534	$3.02^{(3)}$	0.855	$3.162^{(3)}$	0.651	$2.901^{(3)}$	0.376	3.017	0.228
O-MRP	$\mathbf{0.5288}^{(1)}$	0.327	$\mathbf{0.982}^{(1)}$	0.227	$\mathbf{1.444}^{(1)}$	0.252	$\mathbf{2.055}^{(1)}$	0.296	$\mathbf{2.346}^{(1)}$	0.352

Table 8.5: Results per setting of p per method for GDP interpolation, trained on Seoul and tested on Daegu. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Daegu GDP									
	Training region: Taichung									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$2.5227^{(3)}$	0.268	$2.621^{(3)}$	0.273	$2.615^{(3)}$	0.217	$2.591^{(3)}$	0.224	$2.675^{(2,3)}$	0.234
UK	2.6946	0.262	$2.656^{(3)}$	0.311	2.676	0.308	$2.665^{(3)}$	0.22	$2.702^{(2,3)}$	0.278
Basic	3.3167	1.177	3.408	0.721	3.522	0.527	3.63	0.316	3.511	0.155
SAR	3.1231	0.342	3.179	0.37	3.502	1.871	3.428	1.05	3.377	1.29
MA	$4.3585^{(\otimes)}$	1.575	$9.844^{(\otimes)}$	12.13	$5.664^{(\otimes)}$	4.31	$8.666^{(\otimes)}$	16.13	$4.754^{(\otimes)}$	4.063
ARMA	3.8306	1.151	3.641	0.452	3.944	0.773	$4.249^{(\otimes)}$	2.128	$3.979^{(\otimes)}$	0.359
CNN	3.1007	0.17	3.127	0.22	3.182	0.405	3.158	0.524	3.09	0.152
SD-MRP	$2.0466^{(2)}$	1.031	$2.101^{(2)}$	0.602	$2.183^{(2)}$	0.453	$2.428^{(2)}$	0.359	$2.689^{(2,3)}$	0.275
WP-MRP	3.0431	1.467	3.077	0.847	2.879	0.593	3.099	0.331	3.059	0.176
O-MRP	$\mathbf{0.462}^{(1)}$	0.243	$\mathbf{0.938}^{(1)}$	0.259	$\mathbf{1.498}^{(1)}$	0.251	$\mathbf{2.022}^{(1)}$	0.267	$\mathbf{2.493}^{(1)}$	0.328

Table 8.6: Results per setting of p per method for GDP interpolation, trained on Taichung and tested on Daegu. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Daegu COVID-19									
	Training region: Daegu									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$12.365^{(\otimes)}$	1.342	$13.64^{(\otimes)}$	1.475	$13.04^{(\otimes)}$	1.513	$13.17^{(\otimes)}$	1.341	$13.4^{(\otimes)}$	1.202
UK	$12.546^{(\otimes)}$	1.511	$13.09^{(\otimes)}$	1.415	$13.18^{(\otimes)}$	1.313	$13.13^{(\otimes)}$	1.55	$13.48^{(\otimes)}$	1.444
Basic	2.5102	0.234	2.56	0.461	2.435	0.606	2.492	1.006	2.279	1.625
SAR	$1.1801^{(2)}$	0.836	$1.638^{(2,3)}$	0.679	$1.63^{(3)}$	0.629	2.437	2.029	2.402	1.632
MA	1.8745	0.395	2.239	1.085	3.912	7.824	3.807	3.683	$26.69^{(\otimes)}$	83
ARMA	2.3224	1.26	2.355	0.901	2.31	0.675	2.682	1.478	3.453	2.227
CNN	$1.6711^{(3)}$	0.247	$1.794^{(2,3)}$	0.376	1.9	0.662	1.879	1.083	$1.819^{(2)}$	1.97
SD-MRP	2.1936	2.399	1.925	1.014	$1.61^{(2)}$	0.699	$1.708^{(3)}$	0.397	$1.652^{(3)}$	0.248
WP-MRP	2.0035	2.427	$1.948^{(2,3)}$	2.024	$1.632^{(3)}$	0.679	$1.585^{(2)}$	0.452	1.67	0.245
O-MRP	$\mathbf{0.2647}^{(1)}$	0.311	$\mathbf{0.611}^{(1)}$	0.311	$\mathbf{0.757}^{(1)}$	0.366	$\mathbf{1.008}^{(1)}$	0.313	$\mathbf{1.197}^{(1)}$	0.253

Table 8.7: Results per setting of p per method for COVID-19 interpolation, trained on Daegu and tested on Daegu. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

Method	Test data: Daegu COVID-19									
	Training region: Seoul									
	p=0.1		p=0.3		p=0.5		p=0.7		p=0.9	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
OK	$13.167^{(\otimes)}$	1.303	12.85	1.759	$12.92^{(\otimes)}$	1.495	$13.02^{(\otimes)}$	1.518	$13.36^{(\otimes)}$	1.399
UK	$13.056^{(\otimes)}$	1.681	$13.4^{(\otimes)}$	1.487	$13.03^{(\otimes)}$	1.676	$13.53^{(\otimes)}$	1.51	$13.21^{(\otimes)}$	1.56
Basic	6.3494	1.914	7.157	1.274	7.051	0.911	7.168	0.681	7.022	0.767
SAR	5.4214	3.4	4.508	2.157	4.545	2.124	5.092	2.754	5.066	3.261
MA	4.9665	2.465	5.833	5.087	5.485	2.952	4.831	2.675	4.441	2.641
ARMA	4.4368	1.941	3.967	1.274	4.43	1.725	5.326	2.887	4.128	1.777
CNN	$1.8406^{(2,3)}$	0.05	$1.84^{(2,3)}$	0.052	$1.847^{(2,3)}$	0.049	$1.85^{(3)}$	0.054	2.09	0.076
SD-MRP	$1.7902^{(2,3)}$	1.745	$1.862^{(2,3)}$	0.82	$1.81^{(2,3)}$	0.467	1.968	0.23	$1.809^{(3)}$	0.125
WP-MRP	$2.0563^{(2,3)}$	2.958	$1.683^{(2,3)}$	0.912	$3.962^{(2,3)}$	15.4	$1.603^{(2)}$	0.479	$1.664^{(2)}$	0.221
O-MRP	$\mathbf{0.3441}^{(1)}$	0.402	$\mathbf{0.745}^{(1)}$	0.382	$\mathbf{1.219}^{(1)}$	0.448	$\mathbf{1.461}^{(1)}$	0.347	$\mathbf{1.613}^{(1)}$	0.152

Table 8.8: Results per setting of p per method for COVID-19 interpolation, trained on Seoul and tested on Daegu. Top-3 methods were marked using $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ (green), and the worst performing method was marked with $\mu^{(\otimes)}$ (red), determined using statistical significance testing at $\alpha = 0.05$. The lowest error per condition, regardless of statistical significance, was marked **bold**.

8.2 Graphs

We would also like to use this appendix to provide access to line graphs for all combinations of training and test sets, for both datasets. In the thesis, only two of these figured were included as examples. In this section of the appendix, every combination of (target variable dataset, training set, test set) has one graph for each of the following sets of methods:

- **All**: all 10 methods available
- **Purely spatial methods**: ordinary kriging, universal kriging, SD-MRP and O-MRP
- **(Spatial) regression methods**: basic regression, SAR, MA, ARMA, CNN and WP-MRP
- **MRP methods**: SD-MRP, WP-MRP and O-MRP
- **Most competitive**: ordinary kriging, universal kriging, SD-MRP and O-MRP

The figures are generally consistent with the example figure we included in the thesis. However, we can see the lower errors for kriging-based methods reflected in the results for GDP interpolation trained on Taichung and tested on Taipei in Figure 8.12, as well as some strange outliers for high p for MA on COVID-19 interpolation trained and tested on Daegu. It was also unexpected to see the errors of SD-MRP and WP-MRP decrease, rather than increase, in Figure 8.34.

8.2.1 GDP

This section will cover the interpolation of GDP tested on Taipei and Daegu. All figures will use scientific notations for the y-axis for easy readability.

Test region: Taipei

In general, the results for Taipei were not as favourable for MRPs as the results for Daegu. The most notable example of this is the performance of the two kriging methods, whose lines tend to be lower than all others except O-MRP. Moreover, because the kriging methods seem so unaffected by the setting of p, in several cases they even overtake O-MRP for higher p. Only when trained on Taipei did the kriging methods perform poorly.

Training region: Taichung

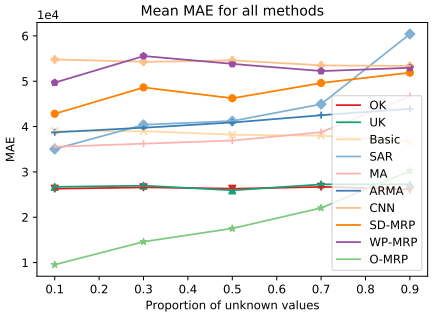


Figure 8.1: Mean MAE as a function of p for GDP interpolation trained on Taipei and tested on Taipei: all methods.

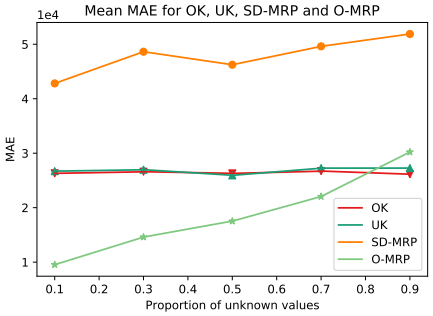


Figure 8.2: Mean MAE as a function of p for GDP interpolation trained on Taipei and tested on Taipei: purely spatial methods.

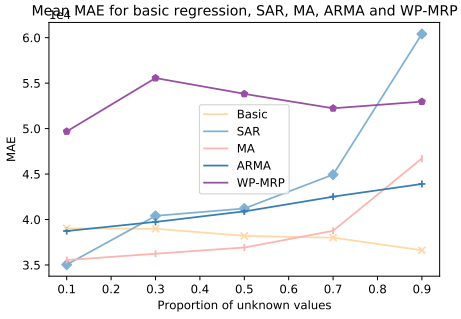


Figure 8.3: Mean MAE as a function of p for GDP interpolation trained on Taipei and tested on Taipei: (spatial) regression methods.

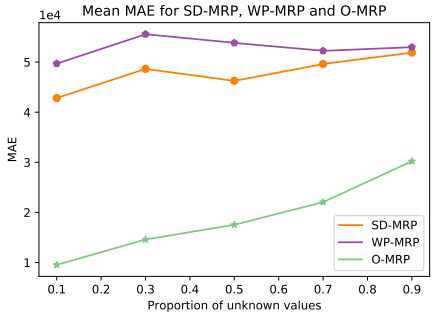


Figure 8.4: Mean MAE as a function of p for GDP interpolation trained on Taipei and tested on Taipei: MRP methods.

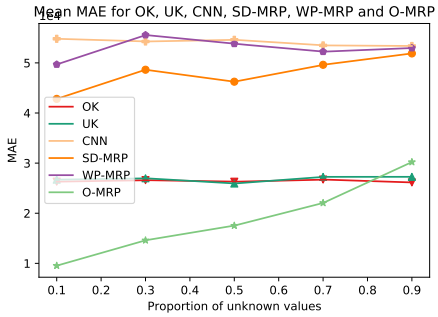


Figure 8.5: Mean MAE as a function of p for GDP interpolation trained on Taipei and tested on Taipei: most competitive methods.

Training region: Seoul

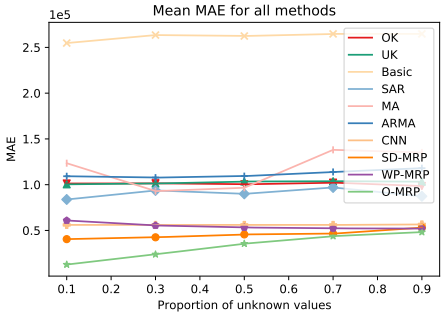


Figure 8.6: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Taipei: all methods.

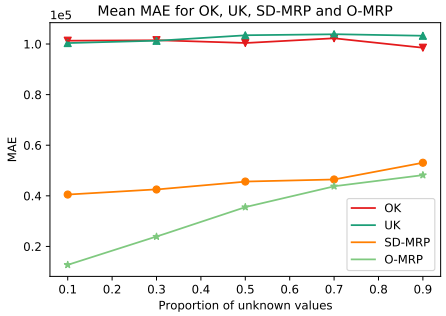


Figure 8.7: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Taipei: purely spatial methods.

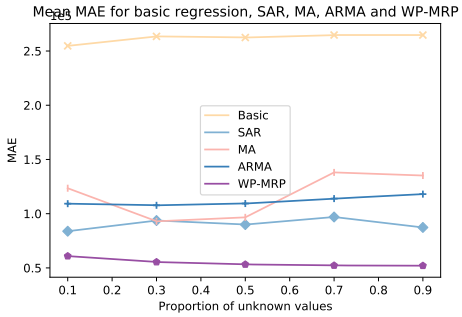


Figure 8.8: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Taipei: (spatial) regression methods.

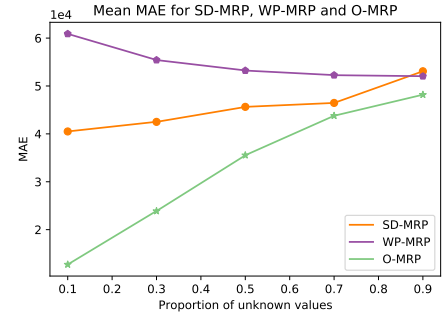


Figure 8.9: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Taipei: MRP methods.

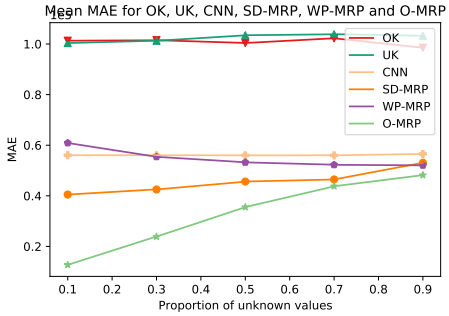


Figure 8.10: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Taipei: most competitive methods.

Training region: Taichung

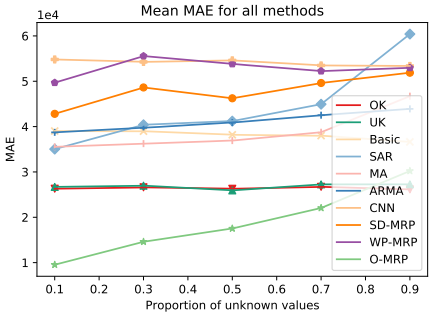


Figure 8.11: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Taipei: all methods.

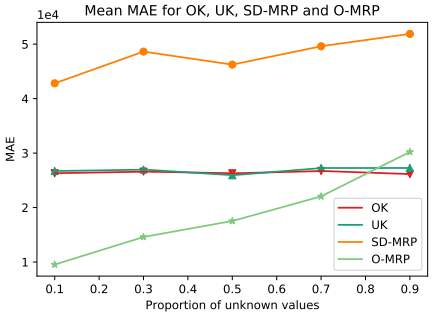


Figure 8.12: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Taipei: purely spatial methods.

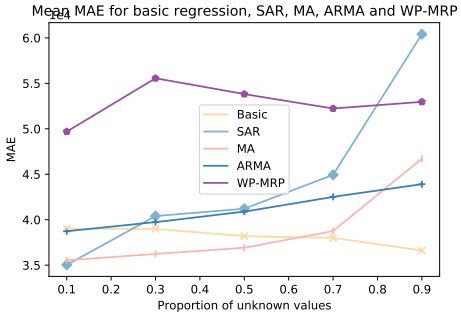


Figure 8.13: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Taipei: (spatial) regression methods.

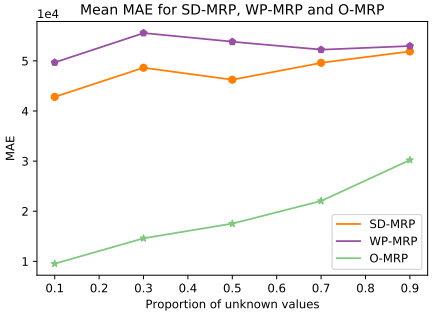


Figure 8.14: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Taipei: MRP methods.

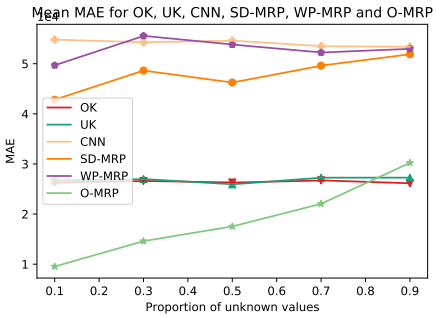


Figure 8.15: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Taipei: most competitive methods.

Test region: Daegu

The results for Daegu were more favourable to MRPs than the results for Taipei. In almost all graphs involving MRP methods, they can generally be found at the bottom of the figure with the lowest errors. Additionally, at times (such as in Figure 28) MA performed highly erratically when tested on Daegu.

Training region: Daegu

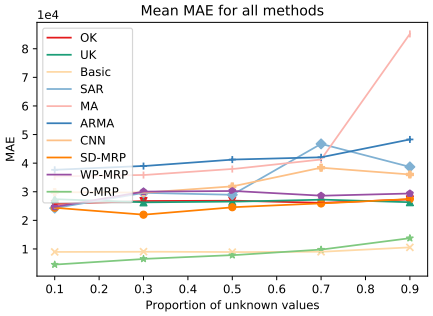


Figure 8.16: Mean MAE as a function of p for GDP interpolation trained on Daegu and tested on Daegu: all methods.

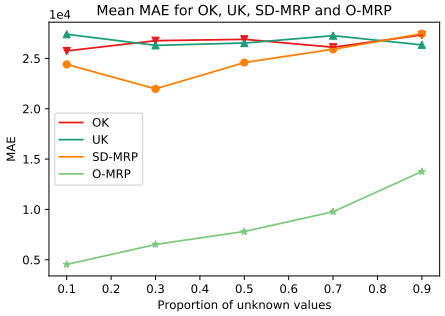


Figure 8.17: Mean MAE as a function of p for GDP interpolation trained on Daegu and tested on Daegu: purely spatial methods.

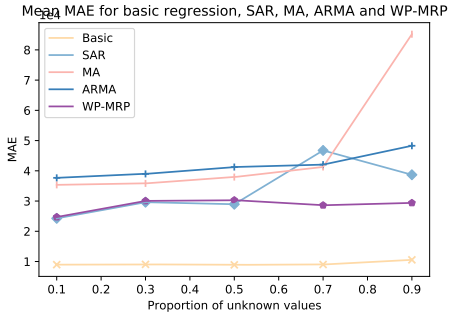


Figure 8.18: Mean MAE as a function of p for GDP interpolation trained on Daegu and tested on Daegu: (spatial) regression methods.

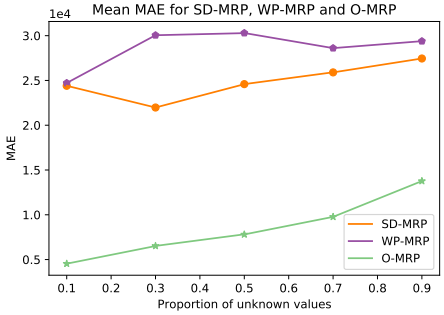


Figure 8.19: Mean MAE as a function of p for GDP interpolation trained on Daegu and tested on Daegu: MRP methods.

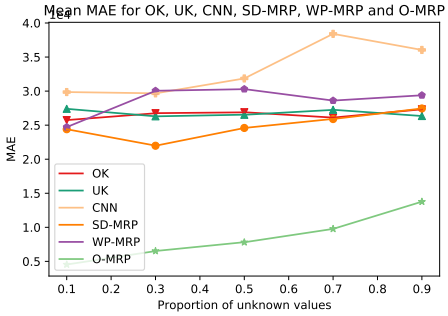


Figure 8.20: Mean MAE as a function of p for GDP interpolation trained on Daegu and tested on Daegu: most competitive methods.

Training region: Seoul

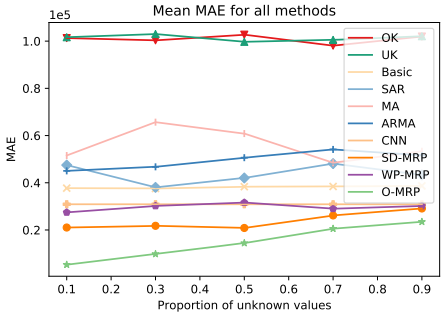


Figure 8.21: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Daegu: all methods.

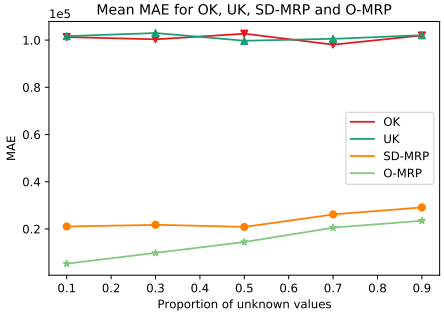


Figure 8.22: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Daegu: purely spatial methods.

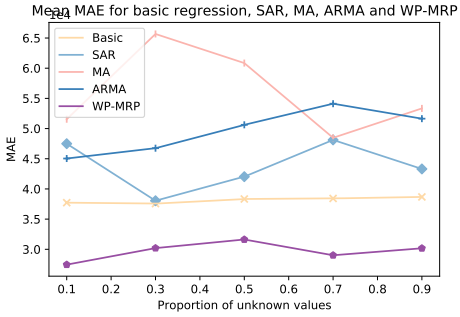


Figure 8.23: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Daegu: (spatial) regression methods.

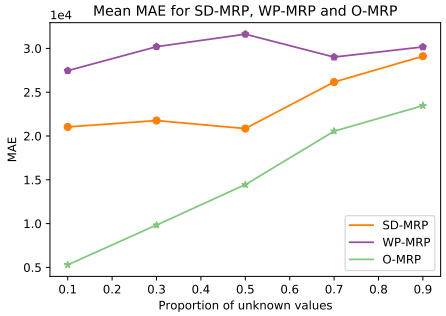


Figure 8.24: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Daegu: MRP methods.

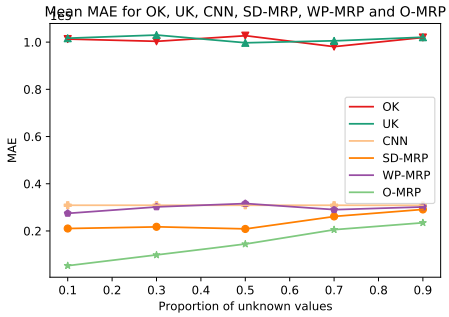


Figure 8.25: Mean MAE as a function of p for GDP interpolation trained on Seoul and tested on Daegu: most competitive methods.

Training region: Taichung

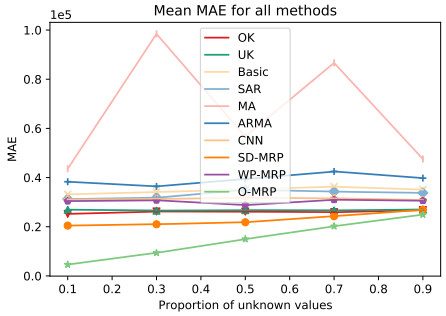


Figure 8.26: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Daegu: all methods.

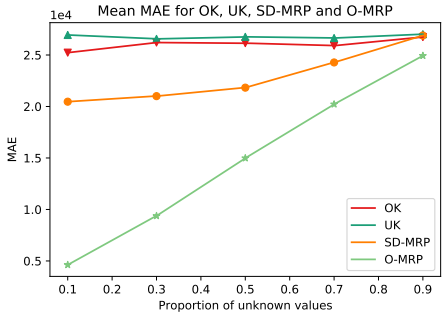


Figure 8.27: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Daegu: purely spatial methods.

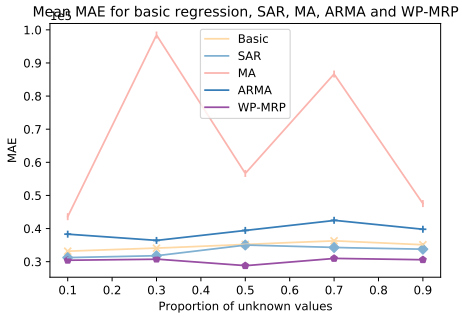


Figure 8.28: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Daegu: (spatial) regression methods.

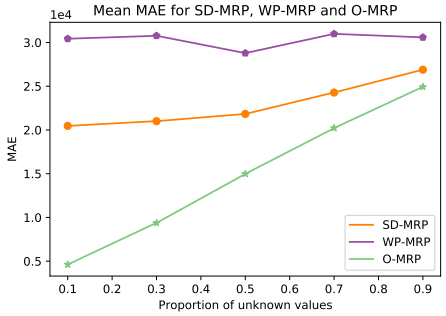


Figure 8.29: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Daegu: MRP methods.

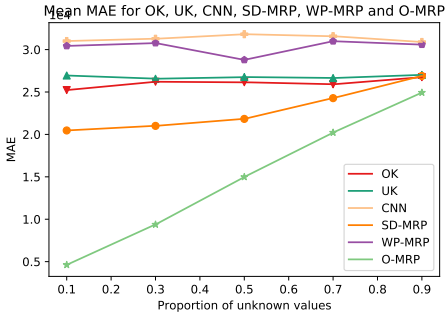


Figure 8.30: Mean MAE as a function of p for GDP interpolation trained on Taichung and tested on Daegu: most competitive methods.

8.2.2 COVID-19

The results for the COVID-19 dataset were by far the most favourable to MRPs. In nearly all cases the MRP methods all performed better than the other baselines. Interestingly, though, basic regression also performed well when trained on Daegu. When trained on Seoul, however, this was no longer the case. As in other cases, we see some erratic performance from MA such as in Figure 33, and strangely we also see the errors of SD-MRP and WP-MRP decrease with p in Figure 34. We are not sure what caused this, but it is certainly unusual.

Test region: Daegu

Training region: Daegu

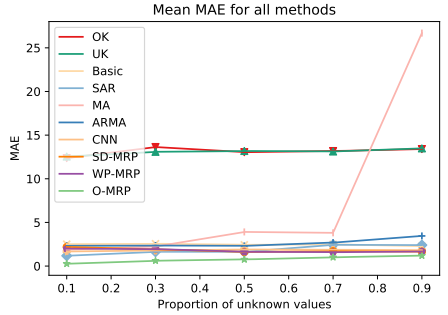


Figure 8.31: Mean MAE as a function of p for COVID-19 interpolation trained on Daegu and tested on Daegu: all methods.

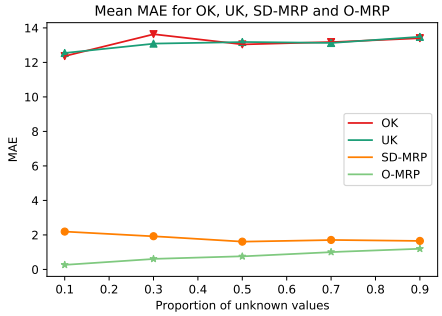


Figure 8.32: Mean MAE as a function of p for COVID-19 interpolation trained on Daegu and tested on Daegu: purely spatial methods.

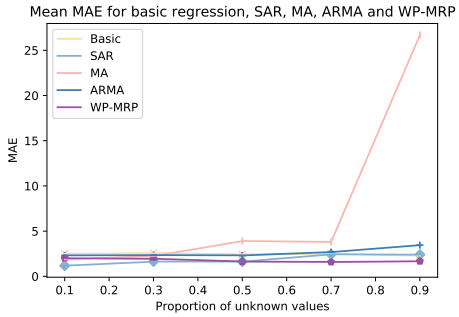


Figure 8.33: Mean MAE as a function of p for COVID-19 interpolation trained on Daegu and tested on Daegu: (spatial) regression methods.

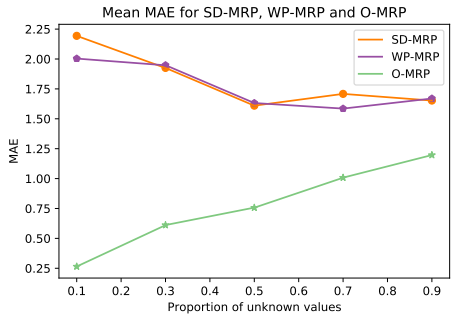


Figure 8.34: Mean MAE as a function of p for COVID-19 interpolation trained on Daegu and tested on Daegu: MRP methods.

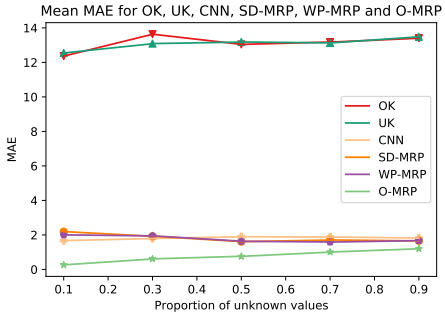


Figure 8.35: Mean MAE as a function of p for COVID-19 interpolation trained on Daegu and tested on Daegu: most competitive methods.

Training region: Seoul

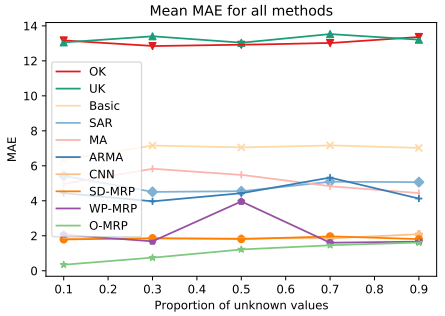


Figure 8.36: Mean MAE as a function of p for COVID-19 interpolation trained on Seoul and tested on Daegu: all methods.

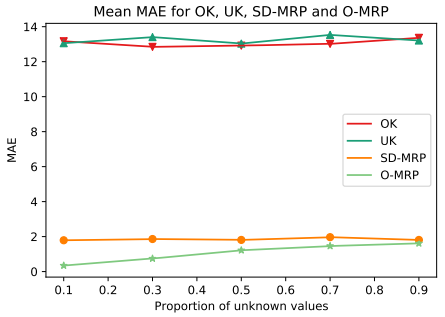


Figure 8.37: Mean MAE as a function of p for COVID-19 interpolation trained on Seoul and tested on Daegu: purely spatial methods.

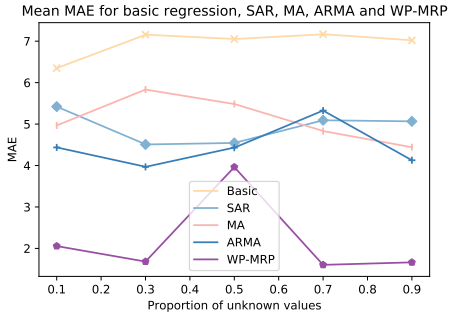


Figure 8.38: Mean MAE as a function of p for COVID-19 interpolation trained on Seoul and tested on Daegu: (spatial) regression methods.

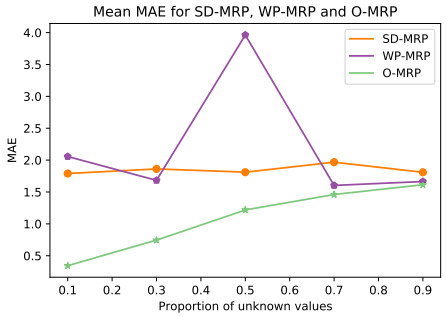


Figure 8.39: Mean MAE as a function of p for COVID-19 interpolation trained on Seoul and tested on Daegu: MRP methods.

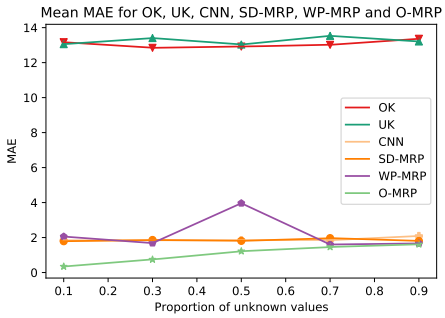


Figure 8.40: Mean MAE as a function of p for COVID-19 interpolation trained on Seoul and tested on Daegu: most competitive methods.