Spring 5-11-2014

DECAF: A New Event Detection Logic For The Purpose Of Fusing Delineated-Continuous Spatial Information

Kerry Q. Hart
University of Nebraska-Lincoln, kerry.q.hart@live.com

DECAF: A NEW EVENT DETECTION LOGIC FOR THE PURPOSE OF FUSING DELINEATED-CONTINUOUS SPATIAL INFORMATION

by

Kerry Q. Hart II

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Ashok Samal

Lincoln, Nebraska

May, 2014

DECAF: A NEW EVENT DETECTION LOGIC FOR THE PURPOSE OF FUSING DELINEATED-CONTINUOUS SPATIAL INFORMATION

Kerry Q. Hart II, M.S.

University of Nebraska, 2014

Adviser: Ashok Samal

Geospatial information fusion is the process of synthesizing information from complementary data sources located at different points in space and time. Spatial phenomena are often measured at discrete locations by sensor networks, technicians, and volunteers; yet decisions often require information about locations where direct measurements do not exist. Traditional methods assume the spatial phenomena to be either discrete or continuous, an assumption that underlies and informs all subsequent analysis. Yet certain phenomena defy this dichotomy, alternating as they move across spatial and temporal scales. Precipitation, for example, appears continuous at large scales, but it can be temporally decomposed into discrete spatial events (e.g., storms, fronts) inside which rainfall is continuous. We describe such phenomena as behaving discretely-continuous.

This thesis presents an event detection framework that both leverages existing techniques such as indicator kriging as well as a novel embedded-graph based algorithms. This event detection framework is applied to the spatial information fusion problem in order to more intelligently parameterize existing interpolation methods according to local spatial structure (i.e., presence or absence of events). Tests on simulated data demonstrate the efficacy of the event detection logic as well as help inform how to better apply this event detection logic to the fusion problem. Results of tests on real-world precipitation data compare favorably, and in some cases outperform,

traditional interpolation methods.

ACKNOWLEDGMENTS

To Dr. Ashok Samal I offer my greatest appreciation. He is both a great mentor and teacher. His rigor has time and again exposed my shortcomings, and his kindness and wisdom has helped me grow to overcome them. Without his patience, support, and good humor this thesis would not have happened. Dr. Samal, thank you.

I also thank Dr. LeenKiat Soh. I am forever grateful for his energy, creativity, and incredible punctualityhardly once did I submit even the most minor memo without a speedy and insightful comment in reply. His contributions pepper this thesis, and it would be but a shadow of itself without his help. Dr. Soh, thank you.

Furthermore, my appreciation to Dr. David Marx. His skepticism exposed many a faulty assumption, and his expertise informed many an approach. The entire Indicator Kriging angle (DECAF-IK) would never have been explored were it not for Dr. Marx. Dr. Marx, thank you.

A note of thanks to the National Science Foundation for providing the Integrative Graduate Education and Research Traineeship (IGERT), and by extension a thank you to the IGERT committee here at the University of Nebraska-Lincoln.

Finally, my eternal gratefulness to my wife Noelle for her love, support, and advice.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Spatial data abundance is the new norm, the consequence of sensor proliferation, inexpensive storage, and near-universal network access. Goods, services, people, and natural phenomena are tracked with increasing regularity and precision. Often, these data capture different aspects of the same phenomenon. For example, regional diesel prices, the presence of bad weather events, and the distribution of grain silos are all spatial information, and all may tell part of a larger story about the economy of a rural farming region. Yet, integrating and interpreting these disparate data sources can be difficult: some sources may overwhelm the analyst with torrents of data while other aspects, or even geospatial extents, may be underreported or poorly covered. When this process is automated such that algorithms are responsible for deriving useful information both more concise and complete, it is known as information fusion

Information fusion is defined by the International Society of Information Fusion as "the study of efficient methods for automatically or semi-automatically transforming information from different sources and different points in time into a representation that provides effective support for human or automated decision making" [7]. The information fusion process can be extended to the geospatial context to parse spatial

data, such as raster fields, vector objects, and point data. However, space exhibits properties that make the application of traditional non-spatial techniques cumbersome, suspect, or simply inappropriate [44]. A true geospatial information fusion engine must incorporate spatial logic in order to effect intelligible, useful output.

This thesis is motivated by the need to incorporate spatial logic into the information fusion process. We focus on a particular subset of spatial input types: point data. Point data is simple. It associates a point location (often represented as a Latitude, Longitude tuple) with a value (or vector of values) that measure the phenomenon of interest. Unlike raster data, coverage is necessarily limited and most locations are unknown. Furthermore, no information about the spatial structure is explicitly recorded. For example, consider trying to classify land use by point data: do two nearby *water* points record a common body, such as a lake, or do they record two parallel ditches along a highway? A raster approach is clearly better suited for the land classification problem; temperature data, however, tends to be very easy to interpolate: given two temperature measurements located near to one another, it is generally reasonable to assume that the temperature in-between can be estimated according to a linear gradient.

In other words, spatial phenomena are assumed to be either discrete or continuous [14]. Discrete objects are those that are distinct in space, such as a paved road or a parcel of land, and are modeled as vectors or objects. Continuous phenomena, on the other hand, are unbroken across the landscape, and are conceptualized as fields and modeled as rasters (e.g., temperature, airborne particulate concentrations). Both conceptualizations are necessary to understand the real world [47] and modeling interactions between them remains an active area of research [24] [31].

It has long been recognized that this dichotomization is imperfect [14]. Continuous phenomena can behave discretely when a barrier separates two otherwise neighbor-

ing locations. For example, air temperature behaves continuously, but a very sharp delineation is found at the rim of the Grand Canyon. Identifying and representing geospatial barriers is an established and well-researched problem [52]. On the other hand, is a forest a discrete object? It is treated so under a land use classification scheme, but Sorites paradox uncomfortably illuminates the imperfect relationship between abstraction and reality[2].

We raise the problem of phenomena which we label as *delineated continuous*. These phenomena can be clearly defined to be either *present* or *absent* at a particular point in space—that is, any phenomenon that can take a zero value (or equivalent). If the values types tend to be collocated (i.e., *present* points near to other *present* points), then the phenomenon acts discretely at large scales. Distinct regions can be identified, and their borders demarcated. However, within a specific *present* region the phenomenon is best described continuously. For example, precipitation happens in discrete events (e.g., cells, storms, fronts), but inside of the borders of those events the precipitation amount varies continuously. In fact, the model can be further extended to include an arbitrary number of distinct zones, as long as the phenomena behaves continuously within the zone. For the purposes of this thesis, we will restrict our analysis to the *present*/*absent* dichotomy.

Delineated continuous phenomena may be further subdivided into static and dynamic categorizations. The static variety have temporally fixed boundaries, such as a mountain range or the borders of a nation. These boundaries may themselves be dependent on the phenomenon at hand; for example, a mountain range may divide a watershed but have no impact on the migration of people, while national borders do not impact the flow of water but may restrict migration. Such delineations can be encoded *a priori* to divide the region of analysis into discrete subregions, each to be treated differently. This area has been comparatively well-researched, though under

such disparate topics as zonal analysis and spatial barriers and attractors [40] [58] [53]. Because these regions can be predefined, skilled analysts can take the time to either manually encode their extents or to verify the results of automated methods.

On the other hand, dynamic delineated continuous phenomena are difficult (and expensive) to encode *a priori*. These events may move as well as appear and disappear through time. For example, no two precipitation events share precisely the same boundaries, and the extent of a storm cannot be known *a priori*. Similarly, the extent of a pollution event, such as an oil spill or hazardous gas leak, may be discretely defined but change over time while the interior is characterized by continuous but variable pollutant concentrations. A fusion engine is only useful if it is highly automated, especially if it is fusing information across many time steps. Therefore, a spatial fusion engine that works with dynamic delineated continuous phenomena should be able to structure the space automatically.

1.1 Motivations

A fusion engine is only useful if it is highly automated, especially if it is fusing information across many units of time. Therefore, a spatial fusion engine that works with dynamic delineated continuous phenomena should be able to structure the space automatically. This thesis is motivated by the need to automatically structure dynamic delineated continuous space in order to inform the spatial fusion process.

Towards this end, we identify two immediate goals:

1. *Automatically identify individual delineated continuous events from sample point data.*

   Discrete events define the delineated continuous phenomenon space. If these

events can be identified from the sample point-data, they might be leveraged by a spatially aware fusion engine. Furthermore, we identify a sub-goal:

a) *The event detection process should be computationally efficient. Sub-polynomial time complexity is preferred.*

Information fusion engines are often associated with large data sets. Because of the proliferation of geospatially aware devices, the amount of available geospatial data is rapidly growing. Therefore, an efficient approach is ideal.

2. *Leverage this spatial structure (the identified events) to improve estimations at unknown points.*

We test the utility of the formalized spatial structure by applying it towards a simple fusion problem: spatial interpolation. While the particular problem is well researched and established methods are effective, we aim to demonstrate that our approach achieves comparable or improved results.

In the long term, we envision a geospatial data fusion engine that exploits many varieties of spatial data inputs—point observations, raster fields, object vectors, etc.—that capture many kinds of spatial phenomena, whether discrete, continuous, or somewhere in-between (e.g., delineated continuous). The event detection logic presented in this thesis would play a critical role in parsing sample-point data that captures delineated continuous events. Furthermore, we envision that the identified delineated-continuous events may be used to better recognize patterns across space and time (e.g., they may make for a useful input to data mining algorithms). We hope that this will better enable the automated fusion of delineated continuous phenomena with discrete phenomena and continuous phenomena to effect a higher-level picture of the overall system. For example, precipitation measurements (delineated continuous) may be

fused with watershed boundaries and water body extents (discrete) as well as soil moisture and transpiration rates (continuous) to arrive at a better understanding of the water cycle in a given region.

1.2 Contributions

In this thesis, we propose a method—Delineated-Aspect Continuous-Event Fusion (DECAF)—that automatically structures the dynamic delineated continuous event space for the purpose of making point estimations. We present two algorithmic realizations of this method, DECAF-Indicator Kriging (DECAF-IK) and DECAF-Embedded Graph (DECAF-EG).

Both DECAF approaches effectively structure the delineated continuous event space into *present* events and *absence* space. We demonstrate the usefulness of this approach by leveraging the derived spatial structure to match and occasionally out-perform the traditional interpolation techniques Inverse Distance Weighting and kriging. The results are mixed, but evidence suggests that structuring delineated continuous holds promise for improving point estimation and, eventually, informing a geospatial information fusion engine.

Furthermore, we have created a software system to both implement and test these approaches. The system supports the following:

1. **Persistence.** Identified events are stored in a database to facilitate future analysis for the purpose of improved data fusion.

2. **Concurrency.** Distinct temporal slices can be computed concurrently, reducing computation time for large spatio-temporal datasets.

3. **Visualization.** Datasets and identified events can be visualized in-program.

The software is written in Python, allowing deployment to a variety of platforms as well as facilitating integration into existing geospatial programs, such as ESRIs ArcGIS.

1.3 Organization

The remainder of this thesis is organized as follows. Chapter 2 reports background information, Chapter 3 formally defines the problem, and Chapter 4 presents the DECAF logic as well as two realizations (DECAF-IK and DECAF-EG). Chapters 5 and 6 report the design and results of tests on simulated data. Chapter 7 considers modifications to the DECAF approach based on the simulation results. Chapters 8 and 9 report the design and results of tests on real-world precipitation data. Chapter 10 concludes the report.

Chapter 2

Background Information

In this chapter we review topics pertinent to DECAF. The first section discusses information fusion and takes the time to tease out definitions that have become, to a degree, confused over time and across disciplines. The second section discusses properties of space that are germane, including spatial autocorrelation and spatial boundaries. The third section focuses on the standard interpolation technique kriging, which DECAF both employs and is compared to. The fourth section wraps up by pausing to integrate information fusion and the geospace together.

2.1 Knowledge discovery and information fusion

The Knowledge Discovery in Databases (KDD) process defines the general problem whereby data too abundant or complex for standard analysis is refined for human digestion [21]. Large amounts of data are stored in databases—hence the name. KDD can be better understood through the "information hierarchy," which distinguishes between *data*, *information*, and *knowledge*. Applying precise definitions to these terms is as much philosophy as science, and even the well-developed field of Information

Science struggles to develop consistent demarcations [59]. For our purposes, however, it is sufficient to make only general distinctions. Data may be understand to be facts, signals, or symbols that have been removed from context and neglected by interpretation [41]. Data itself has no meaning and alone is useless. Information may be structurally identical data, but is relationally associated with other data to answer questions such as *who*, *what*, *when*, and *where* [1]. Knowledge is information collected together for a useful purpose, such as hypothesis testing. It can be understood to answer the *how* question [1].

Information fusion can be understood as a KDD problem, and is defined by the International Society of Information Fusion as "the study of efficient methods for automatically or semi-automatically transforming information from different sources and different points in time into a representation that provides effective support for human or automated decision making" [7]. In other words, information fusion is the process of synthesizing data (with emphasis on multiple kinds of data) to produce output of higher conceptual complexity (i.e., information).

Information fusion need not be understood strictly through the lenses of computer science. For example, the United Nation's Human Development Index may be understood as a non-automated information fusion output [38]. As seen in Figure 2.1, four data sources ("indicators") are mapped to three information types ("dimensions") to synthesize a single knowledge product ("index").

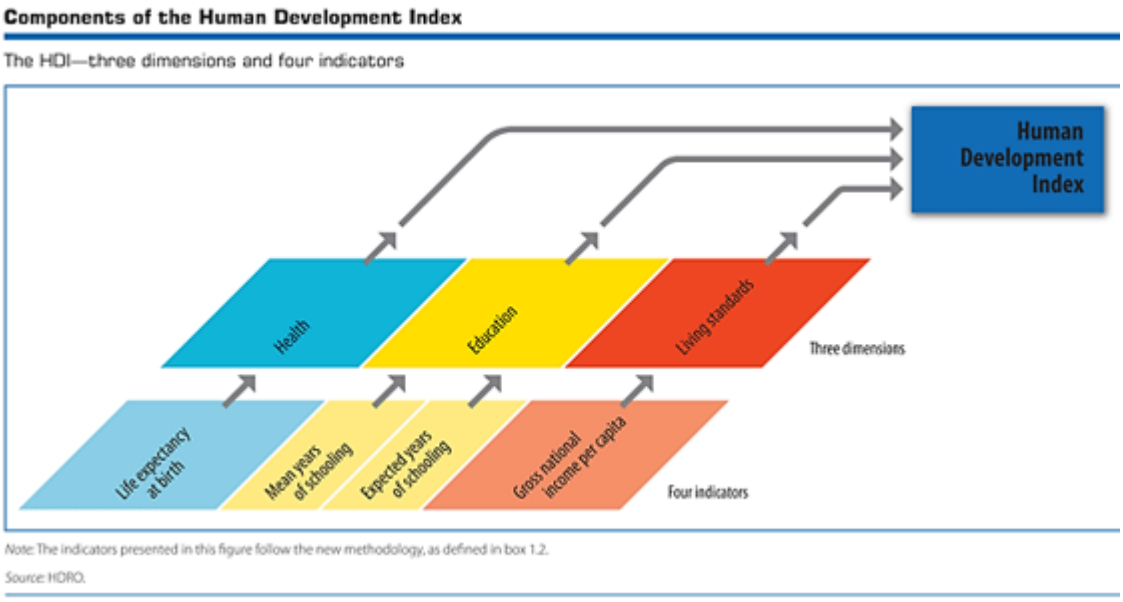


Figure 2.1: The methodology behind the United Nation's Human Development Index [38]

2.1.1 Information fusion, data fusion, and data integration

A brief detour into definition is now necessary. Above—and throughout this paper—we will refer to the process of amalgamating data and elevating it to information as *information fusion*. It is our opinion that this term best describes the process as well as the product (information). The term is also an old one firmly rooted in the information sciences [54]. The related term *data fusion* can be a source of confusion, and so warrants a brief discussion.

The definition of data fusion depends upon context. In the fields of robotics and remote sensing, the terms *data fusion* and *sensor fusion* are used interchangeably in addition to (more rarely) using the term *information* fusion [33] [10]. All three integrate data to produce information, and so are consistent with the notion of moving up the data-information-knowledge hierarchy. It could be argued that data fusion is

a subset of information fusion because data fusion tends to work with inputs that are minimally-filtered while information fusion can utilize a mixture of unprocessed data, processed data, and information. We do not find this distinction to be particularly useful, and instead to be the product of an attempt to shoehorn a popular term (data fusion) into a more logical one.

Second—and of more pertinence to this thesis—data fusion has a very different meaning in the geospatial sciences, where it was borrowed from the field of database management [44] [55]. In database management, the term data *fusion* is synonymous with data *integration* [5] [28]. Data integration is the process by which two or more databases are merged to produce a new database of improved quality [43]. This process involves mapping differing database schema [39], identifying shared objects (e.g., two different databases each record the same farmhouse) [22] [4], and resolving inconsistencies (e.g., two databases each associate the same farmhouse with different owners) [16]. The second step is similar to the sensor fusion meaning of the term, and hence confusion can result. Because this thesis is firmly rooted in the geospatial realm, we deliberately eschew the term *data fusion* for *information fusion* to avoid conflation with the idea of data integration.

2.2 Space

Humans have been filtering, synthesizing, and analyzing spatial information for millennia—and we tend to be good at it. Cartography's long and productive history attests to this fact. The great models of geography are intuitive, such as Von Thunen's agricultural circles and the gravity model of spatial interaction [52] [15]. This success is born from spatial reasoning—which is to say, space has certain properties that differentiate it from the nonspatial that, when recognized, act as a foundation upon which

to reason. We previously discussed discrete vs. continuous space in Chapter 1. We will continue to discuss space here, focusing on the idea of spatial autocorrelation.

When Tobler states in his First Law of Geography that "everything is related to everything else, but near things are more related than distant things" we are almost bewildered by its banality [51]. Yet, it illustrates an essential geospatial truth: spatial events are autocorrelated. Pelicans tend to live near other pelicans; crime tends to happen in the same part of town. Temperatures are not randomly distributed—Baltimore is not a balmy 80° while Washington D.C. is a chilly 30°—and neither are precipitation, vegetation cover, or the density of *Homo sapiens*. In essence, spatial phenomena are, to varying degrees, prestructured. While this can bedevil naive analysis (such as the application of traditional statistical measures, which assume independence), such structure can (and does) inform intelligent algorithms. For example, interpolation techniques—which are discussed in detail below—work because they exploit the fact that nearby points are similar: we can predict the value at point q because that value is related to the values of neighbors a, b, and c—and that relationship can be modelled using well-understood distance relationships.

Spatial autocorrelation can be quantified using different methods. Moran's I is a commonly used statistic that measures autocorrelation on a $(-1, 1)$ scale, where -1 describes a spatial set that is perfectly dispersed, 0 a set that is randomly distributed, 1 a set that is completely clustered [34]. Geary's C is a similar measure, but is more sensitive to localized pockets of autocorrelation [25].

These quantifications of spatial autocorrelation are interesting and can make for useful analysis, but they only summarize the space. They do not provide much useful information upon which to build a spatial information engine, as they do not inform about relationships across space or at particular locations. A more useful construct is the semivariogram.

The intuition behind the semivariogram is straightforward. Assuming some level of spatial autocorrelation, it is reasonable to expect nearby point pairs to be more alike than distant point pairs. If the value difference is found for all point pairs, it can be plotted against distance to produce a graph. From such a graph the precise relationship between distance and value difference might be derived.

Figure 2.2 shows such a graph, though instead of raw difference we use the semivariance (calculated using Equation 2.1, which we will consider in detail shortly). Unfortunately, this figure—referred to formally as an experimental semivariogram cloud—is messy. We do not observe an orderly relationship between distance and semivariance. In part, this is because geospatial data is often noisy and ill-behaved with many different factors at work. However, noise is not the only culprit.

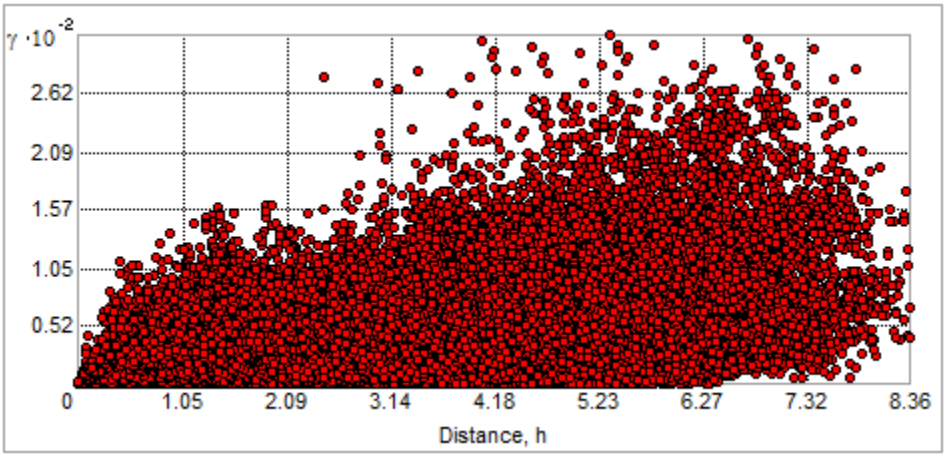


Figure 2.2: An example semivariogram cloud

Consider the bull's eye phenomenon in Figure 2.3. If we plot the relationship between point a and all possible points, we will observe an orderly, increasing semivariogram cloud. However, if we plot the relationship between point b and all possible points, the cloud will not be so meticulously ordered. For example, the semivariance between b-c is the same as that between b-d, but the distances are not. Furthermore,

the distances between b-d and b-e are the same, but the semivariance is not. In environments that are no as carefully ordered as our bull's eye, the cloud becomes very messy very quickly and ceases to be useful.

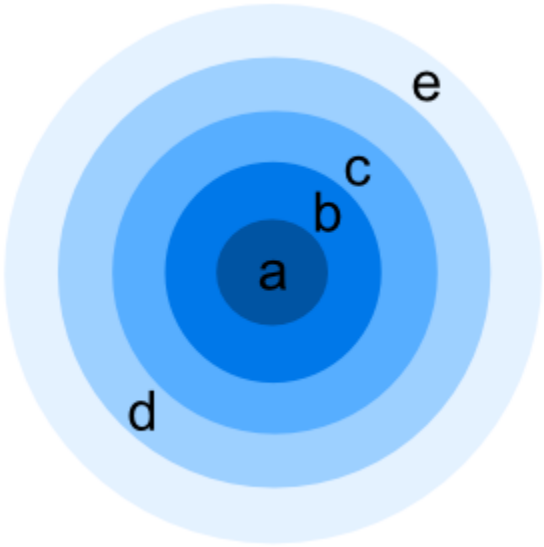


Figure 2.3: Five points in a well-ordered space

However, return to Figure 2.2. Note that we do observe a very general positive relationship between distance and maximum semivariance. This suggests that the semivariogram cloud might be useful if it can be cleaned up.

Consider the following equation, which calculates the semivariance by distance threshold:

$$\hat{\gamma}(h) = \frac{1}{2} \cdot \frac{1}{n(h)} \sum_{i=1}^{n(h)} (v_i - v_{h,i})^2 \tag{2.1}$$

where h is a distance threshold (i.e., the distance between points i and j is at least h), $n(h)$ is the number of point-pairs that are at least h distant, v_i is the value associated with point i, and $v_{h,i}$ is the value assocaited with a point at least h from point i. The variable h is drawn from the series of distance thresholds $H = \langle h_1, h_2, \ldots, h_n \rangle$—these distance thresholds should recreate a figure similar to a bulls eye (though the intervals

need not be uniform in size). In effect, Equation 2.1 bins point-pairs by distance and computes the semivariance within that bin. This is called the *experimental* semivariogram. An experimental semivariogram derived from the Figure 2.2 experimental semivariogram cloud is shown in Figure 2.4.

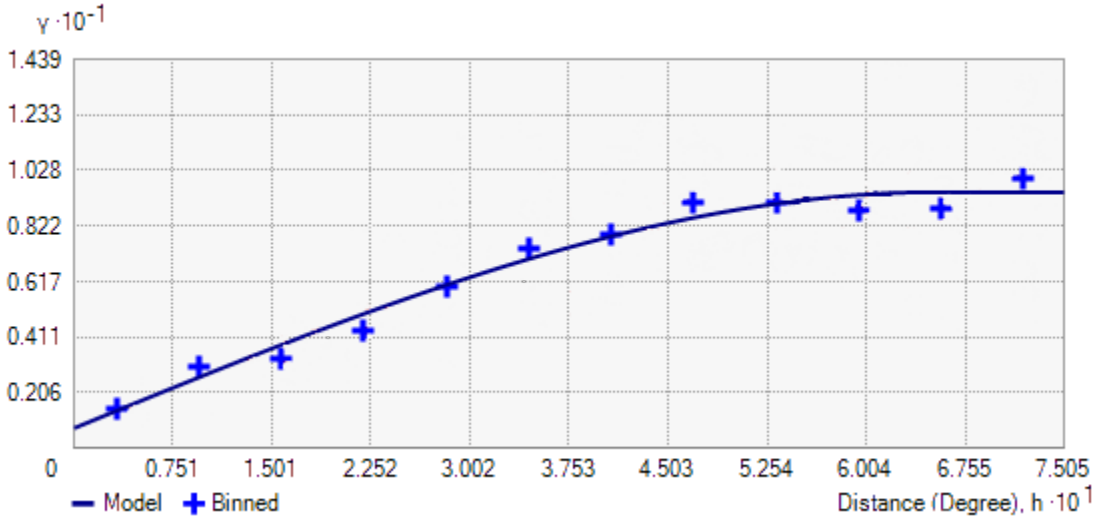


Figure 2.4: Example semivariogram. Produced in EsriArcGIS. Blue crosses represent semivariance bin values. The blue line is the fitted model.

The semivariogram is better behaved than the semivariogram cloud. To it we can fit a semivariogram function, which allows us to model the relationship between distance and semivariance. This is important, as this model can be used to inform predictive functions (e.g., kriging). Because semivariograms are so often used to inform a kriging function, not just any fitted function will do. The function must not produce negative variances, and it must be positive definite. While many semivariogram models exist only a few are commonly used: linear, circular, spherical, Gaussian, and exponential. These all share three parameters in common, and each parameter is easily interpreted:

Nugget. The nugget is the y-intercept of the fitted model. Conceptually, the nugget captures the fact that phenomena often have a minimal area of effect. The term originates from mining gold: gold may be distributed continuously at large scales, but at small scales it is highly clustered into nuggets. A gold nugget may or may not be found right beside another; this is the inherent minimal variance.

Sill. The sill is the horizontal asymptote the bounds the maximal values semivariogram model (y-axis).[1] Conceptually, this is the expected variance between two independent observations (i.e., two non-neighbor points).

Range. The range is the distance (x-axis) at which the curve levels off. Conceptually, this is the distance at which observations cease to be correlated (i.e., the neighborhood distance).

The semivariogram is powerful, but it suffers a limitation: it is a global measure. The semivariogram is computed from all sample points. If two regions in the space are expected to behave differently, the analyst must manually separate the sample points into two sets and compute two semivariograms. If a phenomenon behavies discrete-continuously, individual semivariograms must be computed for each discrete region. If the phenomenon is dynamic, then this process has to be repeated at each individual time step.

2.3 Interpolation

Suppose we have a set of points P that sample a space such that each $p \in P$ has an associated value v_p that measures a phenomenon of interest. Because P is composed of points, there are necessarily locations in our space that have not been sampled

[1] This is not necessarily a true asymptote: for example, a linear semivariogram model will not level off. In the case of these models, the sill must be estimated using alternative methods.

and so have no associated value—i.e., the value is unknown. How do we estimate the value v_q at such a point q?

Recall Toblers Law and the notion of spatial autocorrelation. Because space is so structured, we can reasonably surmise that points from P located near q might inform us about the value v_q. The simplest approach would be to simply assign v_q equal to the value of q's nearest neighbor.

A more sophisticated approach would be to develop some neighborhood of nearby points. This neighborhood could be constructed any number of ways: k points nearest to q, all points within radius r of q, sector-searches, etc.[2] Once this neighborhood is developed, the values simply need to be combined in some fashion. A naive approach would be to use a traditional measure of centrality, such as the mean or the median. We can do better, and we explore two traditional geospatial approaches—inverse distance weighting and kriging—below.

2.3.1 Inverse Distance Weighting

Inverse distance weighting (IDW) is intuitive. Instead of a simple mean, the averaging process is modified to weigh each value according to distance. IDW is a linear combination of the form

$$\hat{v}_q = w_1 v_1 + w_2 v_2 + \ldots + w_n v_n \tag{2.2}$$

where w_i is a weight assigned to value v_i and n is the number of points in the neighborhood of point q. In the case of the tradition mean computation, all weights are equivalent such that

$$w_i = \frac{1}{n} \tag{2.3}$$

[2]See Chapter 4 section 4.4.3 for a more detailed discussion of neighborhood search algorithms.

We want to modify these weights to reflect spatial information, so each weight is a function of distance. Each weight w_i can be made proportional to the distance between the p_i and q. This proportion can take many forms. The term "inverse distance weighting" can refer to many different such functions, such as the linear

$$\frac{1}{distance} \tag{2.4}$$

and the polynomial

$$\frac{1}{distance^2} \tag{2.5}$$

Others exist, but these two are the most common. For the purpose of this paper, when use IDW we use Equation 2.4.

Finally, these weights are normalized such that

$$\sum_{i=1}^{n} w_i = 1 \tag{2.6}$$

So, the linear inverse distance weighted function is of the form

$$\hat{v}_q = \frac{\sum_{i=1}^{m} \frac{v_i}{d_i}}{\sum_{i=1}^{m} \frac{1}{d_i}} \tag{2.7}$$

where m is the number of points neighbor to the estimation point q, d_i is the distance between the estimation point q and the point p_i, and v_i is the value of the point p_i.

2.3.2 Kriging

Kriging is more sophisticated than IDW, and satisfies many of IDW's shortcomings. We will discuss three of these improvements; for a complete discussion see Chapter 9 of [11].

Neighborhood selection. If IDW uses a maximum-radius search is to find neighbors, the size of the radius must be parameterized *a priori*. IDW itself has no mechanism by which to identify a reasonable search radius.

Kriging relies on the semivariogram. Recall that the *range* of a semivariogram was the distance at which points ceased to be correlated (and become independent of one another). This range makes for a reasonable maximum search radius, as points within this radius can inform the prediction and points outside of it cannot (at least on average). Now, while not typically implemented by IDW functions, the semivariogram *can* be used to manually parameterize IDW.

Point clustering. IDW only weights points based on distance. So, a dozen points all tightly clustered to the east of q will each be weighted similarly to one point located to the west. Yet, it is reasonable to think that the dozen points are highly correlated and capture much of the same information while the single point to the west probably captures new, interesting information. Kriging compensates for clustering, splitting weight among clustered points and effectively upweighting points located in alternative directions More accurately, kriging does not explicitly consider direction (that is, if we ignore anisotropy) but instead considers correlation. Highly correlated points are effectively treated as a single point, which because of spatial autocorrelation tends to result in clusters of point q. Note, however, that this measure of correlation is *not* computed from the values of the points but from the fitted theoretical semi-variogram—data is not considered, only location via modelled correlation.

Error map. In practice, IDW and kriging tend not to make wildly different estimations [6]. Kriging, however, produces an estimation of error. Now, the usefulness

> of this error is limited—as [6] notes, it is really just a map of the distance to the nearest point scaled by the covariance function. Furthermore, it is not trivial for a novice to produce "good" error maps (unlike actual prediction maps, which tend to be robust to parameterization). Much of the skill of an experienced analyst is associated with appropriately parameterizing the kriging method to build an accurate error map. But, the quantification of error, however imperfect, is often just as if not more important than the prediction itself [17].

To explain the logic behind kriging, we will begin by focusing on error. In fact, the error map described above is not just a useful byproduct of the kriging method, but that estimation error is essential to the kriging process. The following is based on [11] and [6].

Suppose we define error to be

$$\epsilon = q_v - \hat{q}_v \tag{2.8}$$

where q_v is the true value at point q and $\hat{q_v}$ is the estimated value. If we assume that the sample values follow a normal distribution, then the error values should also be normal with a mean μ_ϵ and standard deviation σ_ϵ. Furthermore, let $E()$ be the expected value of a variable; then, $\mu_\epsilon = E(\epsilon)$ at a given point q. Suppose we know the value at point p. From this it follows that

$$\mu_\epsilon = E(p_v) - E(q_v) = \mu - \mu = 0 \tag{2.9}$$

if we assume that there is no trend in the data. In other words, if the values do not consistently change in one direction, then we expect the error resulting from an estimation based exclusively on the nearest neighbor (p) to be equal to the mean error

μ_ϵ. Finally, we expect that difference, on average, to be zero.

In other words, in the absence of trends, *the average error is zero.*

Next, the variance is of the form

$$\sigma_\epsilon^2 = E((\epsilon - \mu_\epsilon)^2) \tag{2.10}$$

We just demonstrated that $\mu_\epsilon = 0$. Therefore, the error variance is equivalent to the error itself squared. We will skip a couple mathematical steps to arrive at a very important conclusion: *the error variance is equivalent to the squared difference in value between point-pairs that share a similar distance.* This should sound familiar: this is (almost) precisely what the semivariogram computed. Therefore,

$$\sigma_\epsilon^2 = 2\gamma(h) \tag{2.11}$$

where $\gamma(h)$ is the semivariogram (Equation 2.1) and the constant 2 makes the math prettier. Let us now consider a simple example. Suppose we are estimating point q from points p and r. To do so, a linear combination is used (same as for IDW):

$$q_v = w_1 p_{1,v} + w_2 p_{2,v} \tag{2.12}$$

and the error will be estimated using the equation

$$\sigma_\epsilon^2 = 2w_1\gamma(q, p_1) + 2w_2\gamma(q, p_2) - 2w_1 w_2 \gamma(p_1, p_2) \tag{2.13}$$

Observe that the estimated error depends on:

1. The relationship (semivariance) between the unsampled location (q) and the first sampled location (p_1).

2. The relationship (semivariance) between the unsampled location (q) and the second sampled location (p_2).

3. The relationship (semivariance) between the two sampled locations (p_1 and p_2). This term, in particular, is interesting because it captures the relationship between the two known points. Consider these scenarios:

   - *The two points are nearby (e.g., clustered).* The correlation should be high—or, alternatively, the variance should be low. This term (which is subtracted from the others) is low, and the estimated error is larger as a result. These two points were similar, and so should it is reasonable to think that the second point only contributed marginal information.

   - *The two points are distant (e.g., dispersed).* The correlation should be low. Therefore, the variance is high. The overall error is lower because of the presumption that these two points each contribute distinct (and therefore more useful) information.

Let us now take this two-point example and generalize it to the case of n points. We will need n terms of the form $2w_i\gamma(q, p_i)$ and n^2 terms of the form $2w_iw_j\gamma(p_i, p_j)$. We arrive at the equation

$$\sigma_\epsilon^2 = 2\sum_{i=1}^{m} w_i\gamma(q, p_i) - \sum_{i=1}^{m}\sum_{j=1}^{m} w_iw_j\gamma(p_i, p_j) - \gamma(q, q) \tag{2.14}$$

Note that $\gamma(q, q)$ captures the nugget effect.

From here, kriging becomes an optimization problem: the weights in the linear combination will be formulated such that the error is minimized overall. We now consider two varieties of kriging in particular.

2.3.2.1 Simple Kriging

In simple kriging, we assume that the mean is constant across the phenomenon. This is a simplifying assumption, which makes the series of equations we must optimize comparatively simple set of equations

$$\sum_{i=1}^{m} w_i \gamma(p_i, p_j) = \gamma(p_i, q) \tag{2.15}$$

We adjust the weights to equalize the variances; some calculus shows that this also minimizes the overall error.

Once the weights have been calculated, they can be plugged into the linear combination and used to estimate q_v.

2.3.2.2 Ordinary Kriging

Ordinary kriging is more complicated than simple kriging because it drops the assumption of the constant population mean. Instead, we assume that it is constant in the neighborhood around q [6].

Now, the results in a more complicated set of equations (there are now more unknowns than equations and Lagrangian multipliers must be used to get around this fact). For the sake of brevity, we do not report this derivation and instead skip to the matrix form of problem:

$$AB^T = C \tag{2.16}$$

Matrix A contains the n^2 pairwise semivariances among known points (note, most

software packages use the correlation matrix instead, but the intuition is the same):

$$A = \begin{bmatrix} \gamma(v_1, v_1) & \gamma(v_1, v_2) & \cdots & \gamma(v_1, v_m) & 1 \\ \gamma(v_2, v_1) & \gamma(v_2, v_2) & \cdots & \gamma(v_2, v_m) & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \gamma(v_m, v_1) & \gamma(v_m, v_2) & \cdots & \gamma(v_m, v_m) & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{2.17}$$

Matrix C contains the semivariances between the estimation point q and all known sample points:

$$C = \begin{bmatrix} \gamma(v_1, q) \\ \gamma(v_2, q) \\ \cdots \\ \gamma(v_m, q) \\ 1 \end{bmatrix} \tag{2.18}$$

Finally, matrix B is the set of weights (and the Lagrangian variable)

$$B = \begin{bmatrix} w_1 & w_2 & \cdots & w_m & \gamma \end{bmatrix} \tag{2.19}$$

So, if we use the matrix equation $B^1 = A^{-1}C$ we can solve for the weights. Note that this requires that we invert the matrix A, which is computationally expensive ($O(n^3)$).

2.3.2.3 Indicator Kriging

Indicator kriging is nothing more than the adaptation of ordinary kriging to Boolean data.[3] Instead of reporting a continuous variable, indicator kriging maps the two categorizations to the integers 0 and 1. Ordinary kriging proceeds as normal, producing a continuous estimation between 0 and 1. Finally, the range (0, 1) is mapped back to the original categorizations and the final output is discrete [11] [27]. Indicator kriging will be discussed in more detail in Chapter 4 in relation to the DECAF-Indicator Kriging algorithm.

2.3.2.4 Cokriging

Cokriging is the process by which covariate variables are incorporated into the kriging process [27]. In this way, cokriging leverages information related to but distinct from the phenomenon of interest to make better estimations. While cokriging is not neither used nor studied in the course of this thesis, but it warrants a passing mention because of its similarity to the information fusion process. In effect, cokriging performs spatial information fusion, and may play an important role in future spatial fusion engines.

2.3.2.5 Zonal kriging

We end with zonal kriging because it is uniquely relevant to delineated-continuous phenomena.

The term zonal kriging was formalized by William Wingle, though the basic process is probably as old as kriging itself [57]. In short, if distinct regions are best described by distinct semivariograms, then the space is manually divided, distinct semivariograms are computed, and distinct interpolations are stitched together. Wingle

[3]Indicator kriging can be generalized to categorical data with an arbitrary number of classifications, but this requires the use of cokriging and is beyond the scope of this thesis

partially automates this process by improving upon the stitching process, allowing for fuzzy and gradient boundaries.

If the number of distinct spaces is large—whether the result of a large and variable geographical extent or because of dynamism across time—then this method becomes very labor intensive. Each zone must be predefined (the borders must be manually constructed). Furthermore, each zone must be manually identified.

2.4 Spatial information fusion

We define the geospatial fusion process as follows:

> Geospatial information fusion is the process of automatically synthesizing high-level, concise, and integrated information from complementary data sources located at different points in space and time, each of which represent related aspects of a phenomenon, to provide effective support for decision making or scientific discovery.

Incorporating space into the fusion process is a non-trivial task. Many of the assumptions integral to the traditional information fusion problem, such as assumptions of normality and independence, simply do not apply in a spatial context. If spatial properties are ignored or treated in a non-spatial way, synthesized output is probably suspect; on the other hand, if these traits are recognized they might be leveraged to improve output.

Again, spatial information fusion need not be understood through the lenses of computer science. Such an example is the United States Drought Monitor produced by the University of Nebraska in consortium with the United States Department of Agriculture, and the National Oceanic and Atmospheric Administration [9]. In

this case, data such as temperature and precipitation are synthesized into a map of estimated drought severity across the United States. An example map is shown in Figure 2.5.

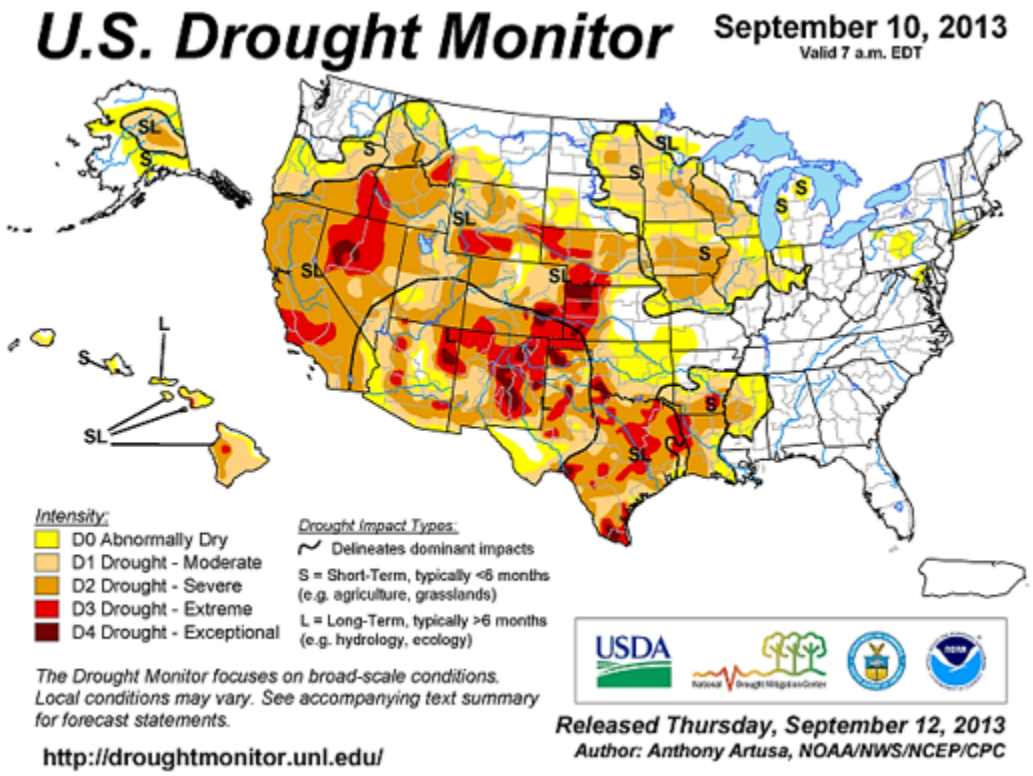


Figure 2.5: Example of U. S. Drought Monitor map [3]

The Drought Monitor is an example of *geospatial* information fusion. Geospatial information fusion anchors the spatial position of data onto the surface of the Earth (typically through the means of a latitude, longitude coordinate pair).

Most traditional examples of automated geospatial information fusion are extensions of sensor fusion. Primarily, they are focused around combining satellite imaging data across the electromagnetic spectrum to produce novel but useful maps, often focused on land-use classification [20] [45] [49]. They also involve fusing raster data between source sets to improve land-use classification [46].

An almost uniquely interesting approach is explored by Carrara et al [8]. This paper explores the creation of environmental indicators by deriving individual phenomenon indicators from remote-sensed raster map data. These indicators are combined at the continent-wide scale using fuzzy set theory. Related papers other fuzzy logic methods, such as Dempster-Shafer [29], but tend back towards either sensor fusion or database integration.

We have discussed two spatial traits in particular: 1) discrete vs. continuous space (Chapter 1) and 2) spatial autocorrelation. We intend to use existing methods—the interpolation techniques just described—to leverage autocorrelation. This thesis introduces a novel approach for automatically structuring space that behaves in a mixed discrete/continuous way. This structure is used to selectively parameterize interpolation techniques to improve estimations at unknown points.

Chapter 3

Problem Definition

This chapter formally specifies the fusion subproblem explored in this thesis.

3.1 The general geospatial fusion problem

Let us consider again our definition of the geospatial information fusion process:

> Geospatial information fusion is the process of automatically synthesizing high-level, concise, and integrated information from complementary data sources located at different points in space and time, each of which represent related aspects of a phenomenon, to provide effective support for decision making or scientific discovery.

We can formalize this definition by defining the geospatial information fusion process as a function

$$f(P, q, t, K) \rightarrow V \tag{3.1}$$

where P is a dataset defined below, q is the location of interest, t is time, K is associated domain knowledge, and V the output of the function (adapted from [33]).

Each element is defined below.

3.1.1 Input

As defined above, the inputs are P, q, t, and K.

3.1.1.1 Defining *P*

A dataset P is a set of sample observations made on a common phenomenon, where P is given by

$$P = \{p_1, p, ..., p_n\} \tag{3.2}$$

where n is the number of total observations. Each observation is of the form

$$p = \langle A, q, t, y, \Delta y \rangle \tag{3.3}$$

where A is the aspect label defined below, q is the associated location, t is time, y is the observed value, and Δy is a measurement of uncertainty associated with a given observation (e.g., quality metadata).

A is an aspect label taken from a restricted set of labels $\{l_1, l_2, ..., l_a\}$ where a is the number of different aspects of the phenomenon under observation. Each observation is associated by its label with a particular aspect. For example, suppose the phenomenon of interest is the population density of an animal species over space and time (such as a migratory bird population). Available aspects of this phenomenon (A) may include photographs from automated cameras, incidental observations reported by citizens, direct evidence such as tracks or abandoned nests, and indirect evidence such as density of predatory species.

3.1.1.2 Defining *q*

The location q may be specified at a single point, along a linear structure, or for a region:

1. The location of a point is specified by its geographic coordinates (latitude and longitude):

$$q^p = \langle Latitude, Longitude \rangle \tag{3.4}$$

2. The location of a linear structure can be specified by a polyline

$$q^l = \langle q_1^p, q_2^p, \ldots, q_n^p \rangle \tag{3.5}$$

   where n is the number of points in the linear structure and each point q^p is of the form given in Equation 3.4.

3. The location of a region can be represented by a polygon as follows

$$q^r = q^l \tag{3.6}$$

   where a line segment is understood to exist between the final point and first point in the polyline vector. The area interior to this hull is considered to be the region.

3.1.1.3 Defining *K*

K represents domain knowledge relevant to the fusion process. By definition, K is dependent upon the problem domain, and we cannot further specify it without a context. Examples of K, however, can include spatial and temporal dependencies

(e.g., business competitors tend to co-locate, precipitation is a leading indicator for stream flow) or environments (e.g., attractors and barriers).

3.1.2 Output

We have defined output V of function f as "high level, concise, and integrated" information. Clearly, a more precise definition of V must rely on context. For example, V may be outputs as disparate as population density of an animal species (to borrow from our previous example) at a point location (q^p), flow volume in a stream (q^l), or estimated demographic information in a county (q^r). It is reasonable, however, to expect a minimal V to be of the form

$$V = \langle v, \varepsilon \rangle \tag{3.7}$$

where v is the synthesized value and ε is a measure of error. Error is important; current geospatial literature argues that information without an associated quantification of quality is, at best, of limited value [30] [26]. Error introduced or propagated by the information fusion process needs to be recorded to inform and protect the consumers of the fusion output.

3.2 Geospatial information fusion for delineated continuous phenomena

To build a complete fusion engine is no trivial task, and describing one is not our intention. Rather, in this thesis we consider a simplified fusion problem: how to select and fuse observations from a sample set P to produce an accurate output V at location q and time t where V is of the same conceptual complexity as P (e.g., if P

is a census of *Alligator mississippiensis* at points in the Okefenokee Swamp then v is an estimate of the number of alligators at point q).

The problem is similar to standard spatial interpolation; indeed, it may be argued that we are recapitulating geospatial interpolation as an information fusion problem. We are. Spatial interpolation may be thought of a special case of geospatial information fusion where only a single aspect phenomenon must be considered.[1] A geospatial fusion engine that relies on point data P *must* use some variety of interpolation logic, as does ours. Our approach is novel because it selectively applies interpolation logic by leveraging the spatial information latent to delineated continuous phenomenon.

Let us now reformulate the formal definition. The function appears the same as before, in the form

$$f(P, q, t, K) \rightarrow V \tag{3.8}$$

and the inputs t and K—location of time, and domain knowledge—remain the same. The inputs P and q as well as the output V are modified.

The input P remains a set of points of the form $\{p_1, , p_n\}$, but the aspect is restricted to a single dimension ($|A| = 1$) such that each $p \in P$ is simplified to $p = \langle q, t, y \rangle$. The input q is restricted to be a point of the form $q^p = \langle Latitude, Longitude \rangle$.

We previously defined the output V to be of the form $V = \langle v, \varepsilon \rangle$ where V was the synthesized value and ε is a measure of error. For the purposes of this paper, however, we will consider instead

$$V = \langle v \rangle \tag{3.9}$$

where v is the synthesized value. We explore the error associated with f through experimental analysis, but we have not yet taken the step to automatically compute

[1] See the discussion on cokriging in Chapter 2 for a traditional interpolation technique that incorporates additional aspects.

error output in f. We leave this for future work.

Chapter 4

The DECAF framework

In this chapter, we outline our Delineated-Event Continuous-Aspect Fusion DECAF approach. The first section will outline the basic logic of DECAF. The subsequent two sections will describe in detail two implementations of DECAF: DECAF Indicator Kriging (DECAF-IK) and DECAF Embedded Graph (DECAF-EG).

4.1 Intuition and overview

We seek to predict the value of a delineated-continuous phenomenon at point q. To do so, we must use known points (points in the set P) that are nearby (neighbor to) q. Choosing precisely what subset of P constitutes a "good" neighborhood for q is not necessarily straightforward, but it is essential—in fact, some argue that a good neighborhood is just as important to accurate estimation as a good interpolation algorithm [6]. A vanilla inverse distance weighted (IDW) approach typically relies on a user-specified cutoff distance or maximum number of neighbors (k-nearest neighbors). Kriging, on the other hand, uses a data-derived cutoff distance.

DECAF attempts to build better neighborhoods by modeling event extents in

a delineated continuous phenomena space. With perfect knowledge, a delineated continuous space can be categorized into two distinct areas:

1. **Present:** these are regions *inside* of delineated events. In *present* regions (that is, inside events) the phenomenon acts continuously.

2. **Absent:** there are regions *outside* of delineated events. In *absent* regions, estimation takes some default value (this may be zero, a predefined constant, or a random number drawn from a predefined distribution).

The motivation of DECAF is simple: by exploiting this spatial structure, we hope to improve predictions involving delineated continuous phenomena (and thereby improve the associated information fusion process). While a human expert may be able to manually categorize a delineated continuous space into discrete events, we want to automate this process so that it can be deployed across hundreds or thousands of distinct event spaces.

DECAF can be understood as a tripartite process: structure the space from P, locate structures relevant to q, and predict V at q.

We begin by structuring the event space into discrete *present* and *absent* regions. Most of DECAF's complexity arises in this step, and it is here that DECAF-EG and DECAF-IK differ. This process is demonstrated in Figure 4.1 (a) and (b) (following page). Second, DECAF checks whether q is in *present* or *absent* space. This step is comparatively trivial. The process is demonstrated in Figure 4.1 (c) and (d). Finally, if q is inside a *present* region, DECAF uses the points associated with that region for prediction; otherwise, q is presumed to be *absent* and is assigned the default *absent* value (typically zero). The precise method that can be used to predict a present q may vary. The process is demonstrated in Figure 4.1 (c) and (d).

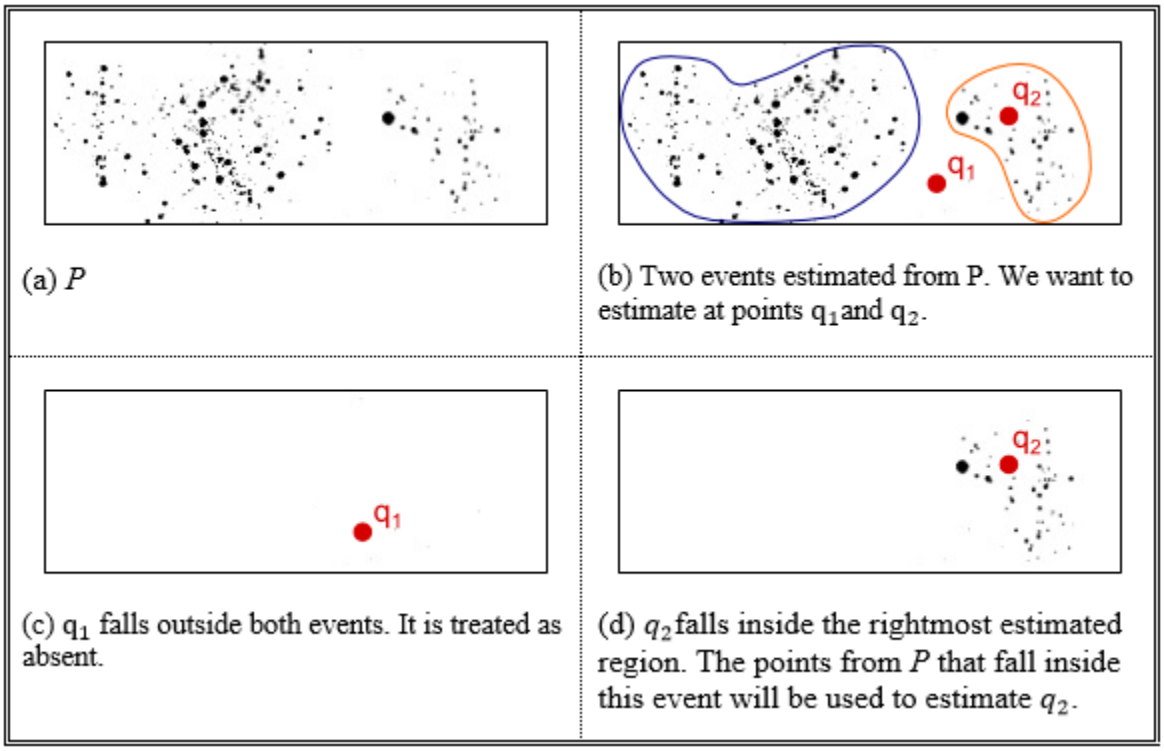


Figure 4.1: The DECAF approach to delineated continuous data

In short, DECAF first detects events before using those events to make predictions. Pseudo-code outlining this process is shown in Algorithm 1. Events (discrete *present* regions) are discovered in Line 1. In Line 2, q is associated with one of these events (or none, if it falls outside of all such events). If an associated event $\hat{e}$ is found, that event is used to approximate V (Lines 5-6). Otherwise, V is set as the default *absent* value in Line 3 (zero in this case).

We will now discuss these three steps in more detail.

4.1.1 Developing spatial context

We structure the spatial context by identifying the set of distinct events

$$E_t = \{e_1, e_2, \ldots, e_n\} \tag{4.1}$$

Algorithm 1 Delineated-Event Continuous-Aspect Fusion

```
1: function DECAF(P, q, t, K)
2:     Ê ←find_events(P)
3:     ê ←find_associated_event(q, t, ê)
4:     V ← 0
5:     if ê ≠ NULL then:
6:         V ←predict(ê, q)
7:     end if
8:     return V
9: end function
```

where t is a time stamp or date, n is the number of events, and each e_i is a region (polygon) located in the geospace that defines the extent of an individual event. The interior of e_i is a continuous spatial random field.

Since these regions are neither static nor predetermined, they have to be approximated from P. To this end we compose a set of event approximations

$$\hat{E}_t = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_m\} \tag{4.2}$$

where m is the number of detected events (if P is accurate then $m \leq n$) and each $\hat{e}_i$ is a subset of P. The hull of $\hat{e}_i$ is an approximation of the shape of e_i, and the members of $\hat{e}_i$ sample the spatial random field interior to e_i.[1] This hull may be the convex or concave; in either case, the goal is to properly compose each $\hat{e}_i$ to accurately reflect e_i.

Because we are interested in phenomena that behave delineated continuously, we can classify each p_i as either *present* or *absent*. Present indicates $v_i > \beta$ and absent $v_i \leq$, where β is a predefined threshold parameterized according to the domain. For example, all soil contains trace amounts of the potentially-harmful element Radon [23]. Yet, below a certain concentration (a threshold β) it meaningfully harmless,

[1] In the case of a completely continuous event (one that exhibits no discrete behavior) $E_t = \{e\}$, where e is composed of all points in P.

and so it can be treated as absent. Above this threshold, however, it is harmful to human health, with increasing doses causing more harm. Above the concentration threshold β, it behaves continuously.

Because each *present* point indicates the presence of an event e_i, we group *present* points to compose $\hat{E}$. Two neighboring *present* points do not necessarily record a common event, but neither do they indicate two separate ones. To settle this impasse, we introduce a bias: in lieu of information to the contrary, nearby *present* points record the same event. Adjacent *present* points are grouped; absent points split the groups. Figure 4.2 demonstrates this logic.

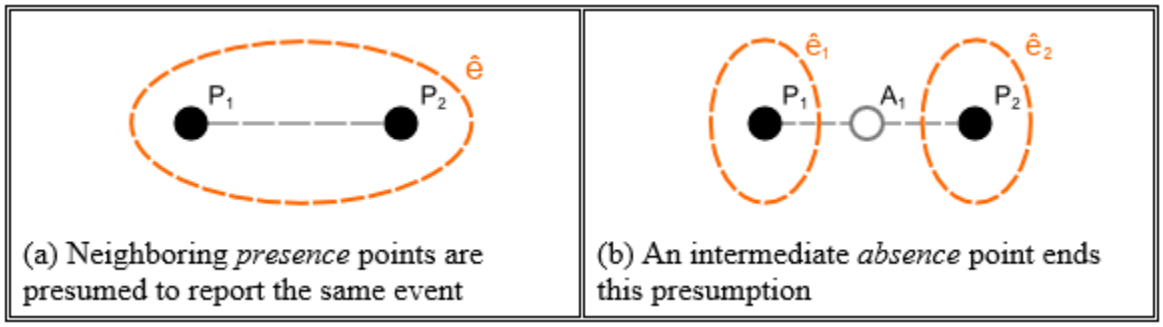


Figure 4.2: Forming event estimations by presence/absence association

The algorithm by which these events ($\hat{E}$) are detected can vary. We explore two different methods, DECAF-IK and DECAF-EG, later in this chapter. The first, DECAF-IK, uses an indicator kriging approach while the second, DECAF-EG, uses an embedded graph. Later, in Chapter 7, we will reconsider and modify DECAF-EG.

4.1.2 Finding associated events

The algorithm must now determine whether the point of interest q is located inside of any of the events $\hat{e} \in \hat{E}$ (Line 3 of Algorithm 1). Because each $\hat{e}$ has an associated bounding hull, this process is simple. For each $\hat{e} \in \hat{E}$, DECAF tests whether q is

spatially inside of $\hat{e}$ using the `find_associated_event` method in Line 2 of Algorithm 1. When a circumscribing $\hat{e}_i$ is found, an association is made between this $\hat{e}_i$ and q.

4.1.3 Computing V

As noted earlier, the prediction point q can fall inside some event $\hat{e}_i$ or, instead, fall outside all events. In this second case, the estimated value V for q is set to a predefined *absent* value. In most cases, this may simply be zero. In cases where a low level of noise is expected and well understood, V may be drawn from a known distribution.

If, on the other hand, q falls inside some event $\hat{e}_i$, the function `predict(`$\hat{e}, q$`)` will be used to estimate V. Because q is inside of the spatial extent of $\hat{e}$, predict must perform spatial interpolation. Instead of trying to invent our own, we simply leverage existing interpolation methods. These methods can be described using the general form

$$i(P, q) \rightarrow V \tag{4.3}$$

where P is the set of sample points where measurements are available and q is the prediction point. Different interpolation techniques select a subset of P in various ways. Some, such as Inverse Distance Weighting (IDW), are blind—they are parameterized with a maximum search radius and/or a maximum number of neighbors k. Others are partially data driven. Kriging, for example, estimates the range of spatial dependence (the range at which points are correlated) and only uses points within this range of q to estimate the value at q.

DECAF selectively composes subsets of P to parameterize the function `i(`P, q`)` for a specific q. This composition is based on the estimated events: only points in the event $\hat{e}_i$ that circumscribes q are used. So, our function `predict` simply encapsulates

a traditional interpolation method.

The choice of the interpolation algorithm is important. A kriging variant may be ideal, as kriging is robust and provides an estimation of statistical error [11]. However, kriging requires fitting a model to the semivariogram before prediction. A robust fit requires a minimum number of points (the industry rule of thumb is around 30 points [50]), and standard kriging packages require a minimum number of points before even attempting a fit [36]. Because the size of $\hat{e}_i$ is not guaranteed, we currently use the simpler IDW algorithm in our implementation.

4.1.4 Incorporating time

For the purposes of this thesis, we treat events as independent across time—what happens at time t has no impact on time $t+1$. Therefore, DECAF only uses observations associated with points in sample set P that match the parameterized time t. It makes no effort to leverage information latent across time. We leave this to future work.

4.2 DECAF-Indicator Kriging

DECAF has three basic steps: event estimation (the derivation of $\hat{E}$), association of q with an event ($\hat{e}_i$) (or no event), and using the associated event to estimate V. The second and third step were described in the previous section; they are straightforward and consistent across our DECAF implementations. The first step, event estimation, is where most of the logic resides, and it is here that the DECAF implementations are differentiated.

4.2.1 Overview

DECAF-Indicator Kriging uses a geostatistical approach to identify events, and so is intuitive to one familiar with geostatistical methodology. Simply put, indicator kriging is used to categorize the event space into regions of *presence* and *absence* using a grid approximation. Next, this raster is translated into a series of hulls, each hull representing a unique event.

Indicator kriging is really nothing more than ordinary kriging applied to a binary space; all observations are effectively 0 or 1 [11]. By dividing this space into discrete thresholds (e.g., between 0.1 and 0.3), indicator kriging (in conjunction with cokriging) can be used to interpolate an arbitrary number of discrete categorizations [19]. For our purposes, two categories will do: *presence* and *absence*. The output generated by indicator kriging at point q can be interpreted as a "probability that the unknown value is above the cutoff value" (assuming that the bottommost value has been mapped to zero and the topmost to one) [11]. For our purposes, then, 0 indicates *absence* and 1 indicates *presence*; an output of 0.75 can be translated as a 75% chance that point q is of type present (and therefore there is a 75% chance that q is inside some event $\hat{e}$).

Now, simply estimating whether or not q is present or absent is not very helpful. In the end, we seek a prediction of event magnitude—simple *presence*/*absence* is n0t sufficient. In the short run, we want to approximate the set of events E, and the solitary point q is not helpful. However, it is straightforward to imagine that with enough such estimated points a reasonably complete picture of the event space may begin to emerge. To accomplish this, we simply do what geostatisticians and GIS users have been doing for decades: we predict every point in a predefined grid, producing a raster representation of the *presence*/*absence* space.

From here, this binary map needs only to be translated into a set of hulls. Once accomplished, we have our approximation of the events E. Next, the points from P are mapped onto the individual events $e \in E$ (that is to say, the set P is surjective to E). Once complete, the standard DECAF algorithm can continue: the point q can be associated with some event $\hat{e}_i$, and the points from P mapped to this $\hat{e}_i$ can be used to approximate the value (V) at q.

4.2.2 Thresholding input data

The first step in the DECAF-IK process is to create from sample set P a new set P^B where each $p \in P$ is of the form

$$p = \langle A, q, t, y, \Delta y \rangle \tag{4.4}$$

as defined in Chapter 3. Now, however the associated value y is strictly binary ($y \in \{0, 1\}$).

This process is straightforward: every point $p_i \in P$ is considered; if the associated value $y_p > \beta$ (where β is a threshold, often but not necessarily zero) then $p_i^B = 1$ else $p_i^B = 0$.

4.2.3 Indicator kriging

Next, indicator kriging is performed. This process is straightforward, as it does not deviate from the standard indicator kriging procedure.

It is important, however, to consider the resolution of the grid. A low-resolution grid may be fast to compute but suffer inaccuracies. A high-resolution grid may achieve a better result, but at the cost of slower computation time. Resolution grows polynomially, and so can become expensive very quickly. Simultaneously, the utility

of increased resolution follows a traditional diminishing returns curve. Finding the most appropriate grid resolution is, unfortunately, domain-dependent.

Figure 4.3 shows two different resolutions. Map b is much smoother and, depending on the application, may be required. Map a is roughly forty times less dense, however, and is commensurately faster.

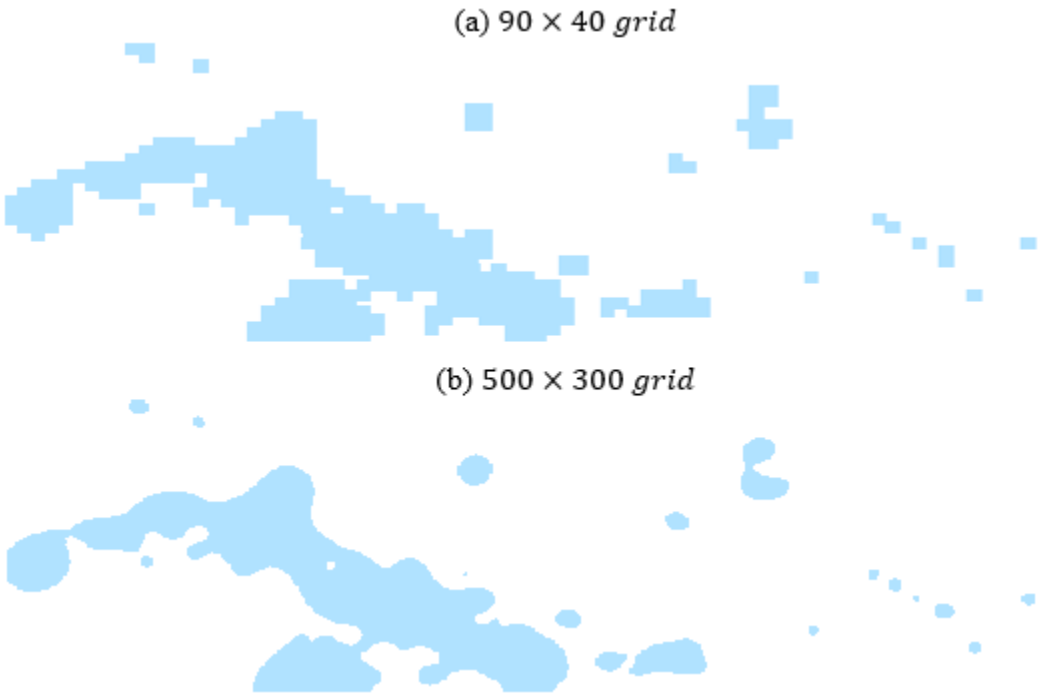


Figure 4.3: Example DECAF-IK presence/absence grids (blue is *presence*)

4.2.4 Boundary detection

At this point, the event space has been categorized into *presence* and *absence* regions. However, no relationships between individual presence grid cells have been developed—no notion of an event yet exists. So, even though q can now be predicted to be *present* or *absent*, points cannot be intelligently selected from P to perform

the estimation. Event extent estimations must first be extracted from the binary *presence/absence* raster.

Many edge detection methods have been developed in the fields of computer vision and raster analysis. While we could choose to borrow one of these, we instead find the mathematical study of alpha shapes to be more helpful. Alpha shapes are simply polygons that circumscribe all points in a point cloud; an example of a well-defined alpha shape is the convex hull. Non-convex alpha shapes are necessarily concave hulls, which are more difficult to define (except for the fact that all points in the cloud must fall inside the polygon, "goodness of fit" is a judgment left to human eye).

We choose this route for efficiency's sake: DECAF-EG requires a similar hull algorithm but has no associated raster: only a point-cloud method will work. On the other hand, we can trivially treat a raster as a point-cloud by translating each raster cell to its center point. Therefore, the same hull-finding approach can be used in both DECAF-EG and DECAF-IK, which saves the trouble of implementing two different algorithms.

To compute the alpha shape, we rely on the `alphahull` package from the R language [37].

Figure 4.4 on page 46 builds on Figure 4.3 to show the estimated events. Because the alpha hull method is an approximation based on cell centerpoints, it does not guarantee that the edge will follow the precise outline of the rasterized region. At high grid resolutions, however, it is effectively equivalent to a raster edge detection algorithm. Note that in (a) not all *presence* pixel groups have a corresponding hull. Only groups with >5 pixels are considered for hull approximation (this is both a limitation of the alpha-shape algorithm and a deliberate decision to match similar weeding logic in DECAF-EG).

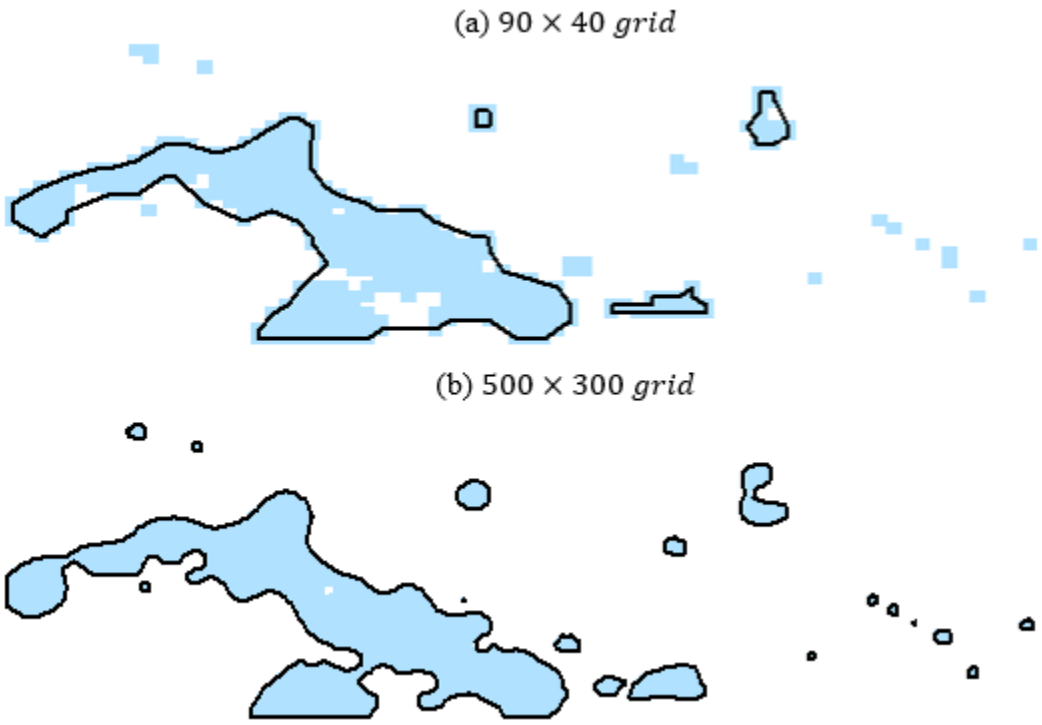


Figure 4.4: Example DECAF-IK event extent approximations (blue is *presence*, black is the concave hull)

4.2.5 Event filling

At this point we have a polygon that estimates the extent of event e_i. This polygon, in and of itself, does not help us estimate a point q beyond a simple prediction of *presence* or *absence*. If q falls inside of $\hat{e}_i$, we need to parameterize the prediction function `predict(`$\hat{e}, q$`)` with sample points from inside of $\hat{e}_i$. To do so, we need to "fill" our polygon with points from P.

This procedure is straightforward. We check each point $p \in P$ to see if it falls inside the polygon associated with each $\hat{e}_i$; if it does, the point is assigned membership to a set associated with $\hat{e}_i$.

4.3 Time complexity analysis: DECAF-IK

4.3.1 Developing spatial context

First, DECAF-IK must threshold all points in the sample set P. This is accomplished in time linear to P.

Second, an $m \times n$ grid is imposed on the event space. We assume that $m \approx n$, so we consider the grid to be size $O(m^2)$. Each point in this grid must be estimated by the indicator kriging function to determine *presence/absence*. The time complexity of ordinary kriging (of which indictor kriging is a variant) is $O(|P|^3)$ [48]. Therefore, the time complexity of grid estimation is $O(m^2|P|^3)$.

Finally, a polygonal hull must be estimated from the *presence/absence* grid. Our implementation uses the alpha-shape detection library alphahull. This package uses the Edelsbrunner Algorithm, which is of time complexity $O(n \log n)$ (Edelsbrunner relies on the Delaunay triangulation, which takes $O(n \log n)$ time) [37]. In our case, n is the number of points in the grid, defined above to be of size m^2. Therefore, polygonal hull estimation takes $O(m^2 \log m^2)$.

Therefore, developing spatial context takes time

$$O(m^2 \cdot |P|^3) + O(m^2 \log m^2) \tag{4.5}$$

At high-resolution grids, the right term may dominate; if the sample set is dense, the left term may dominate.

4.3.2 Computing V

The fusion algorithm can be decomposed into two steps. First, it must determine whether q is inside of an $\hat{e}_i$; in the worst-case, it is outside $\hat{E}$. Each `is_inside`

determination is linear to the number of edge points in $\hat{e}_i$; in the worst-case, the number of border points in $\hat{E}$ is equal to $|P|$.

Second, the points are interpolated, and the time complexity will vary with the choice of interpolation algorithm. IDW, which we use, is constant to the maximum number of neighbors allowed, k. Finding these k neighbors, however, is linear in $|P|$ if a distance matrix is used and logarithmic in $|P|$ if a k-d tree is used. Composing a distance matrix requires $O(n^2)$ time while composing the k-d tree requires $O(n \log n)$ time. We use the k-d tree, so IDW time complexity is $O(|P| \log |P|)$.

The time complexity of fusion is therefore:

$$O(|P|) + O(|P| \log |P|) = O(|P| \log |P|) \tag{4.6}$$

4.3.3 Incorporating time

If a single moment in time is considered, the time complexity is the total of event construction and fusion,

$$O(m^2|P|^3 + m^2 \log m^2) + |P|) = O(m^2|P|^3 + m^2 \log m^2) \tag{4.7}$$

If time is incorporated via a time window τ, where $\tau = (t_s, t_f)$, then the time complexity rises to:

$$O(|\tau| \cdot (m^2 \cdot |P|^3 + m^2 \log m^2)) \tag{4.8}$$

The union operation is linear to the number of points in $\hat{E}_s$ through $\hat{E}_f$. Assuming these are similar, we arrive at:

$$(|\tau| \cdot (m^2|P|^3 + m^2 \log m^2)) + O(2 \cdot |P|) = O(|\tau| \cdot (m^2|P|^3 + m^2 \log m^2)) \tag{4.9}$$

Given the linear rate of time and the non-linear proliferation of spatial data sources, it is reasonable to argue that in most applications $\tau \ll |P|$.

4.3.4 Aggregated time complexity

The time complexity of DECAF-IK is

$$O(|\tau| \cdot (m^2|P|^3 + m^2 \log m^2)) \tag{4.10}$$

and we assume that $\tau \ll |P|$. Therefore, the overall time complexity of DECAF-IK is cubic in $|P|$ and square in m:

$$O(m^2|P|^3 + m^2 \log m^2) \tag{4.11}$$

4.4 DECAF-Embedded Graph

DECAF-IK has certain strengths. First, indicator kriging, which it relies upon, is an established and well-understood geostatistical procedure. Second, it can be performed at an arbitrarily-fine resolution. However, it is computationally expensive: the indicator kriging approach is slow and is impractical for a large P over a fine-resolution space

DECAF-Embedded Graph (DECAF-EG) uses a different method to approximate the event set E. DECAF-EG treats the point set P as a graph embedded in geospace and uses graph component analysis to find $\hat{E}$. The result is an approach that is $O(n \log n)$ in the size of P and is free from the constraints of a grid—and, therefore, is much more scalable.

4.4.1 Overview

The algorithm for DECAF-EG's `find_events(P)` function is outlined in Algorithm 2. DECAF-EG is otherwise identical to DECAF-IK; the input, the logic to find event approximation $\hat{e}$, and the logic to use $\hat{e}$ to estimate q are unchanged (Lines 1, 3, and 4-6 of Algorithm 1, respectively).

Algorithm 2 DECAF-Embedded Graph's find_events algorithm

```
1: function FIND_EVENTS ALGORITHM(P)
2:     G ← graph()
3:     G.nodes ← P)
4:     for x ∈ G.nodes do
5:         if x.v > β then:
6:             N ←find_neighbors(x, G)
7:             for m ∈ N do
8:                 G.add_edge(x, m)
9:             end for
10:        end if
11:    end for
12:
13:    Ê ←connected_components(G)
14:    find_hulls(Ê)
15:
16:    clean_events(Ê, K)
17:    fill_events(Ê, P)
18:    snap_boundaries(Ê)
19:    return Ê
20: end function
```

DECAF-EG begins by building an embedded graph G where each node $n \in G.nodes$ is associated with a $p \in P$ (Lines 1 and 2 of Algorithm 2). In other words, each sample point is a node in the graph. Next, in Lines 3-7, edges are added to the graph G to encode the adjacency relationships between points. After the adjacency relationships are established, the graph is then divided into connected components (Line 9) and the bounding hull of each connected component computed (Line 10). Each component estimates an event e, and so the set of connected components is

$\hat{E}$ (the approximation of E). Finally, in Lines 12-14, the connected components are refined to improve the accuracy of $\hat{E}$.

Three examples of these $\hat{E}$ are given in Figure 4.5 on page 52. Note how the composition of $\hat{E}$ varies considerably through time: (a) is a sparse graph with a handful of small clusters, (b) is characterized by a large cluster covering the entire southeast with several small clusters dotting the west, while in (c) a single cluster dominates the entire space. It is reasonable to posit that a global approach to characterize this space across time would be prone to inaccuracy (i.e., regions of *presence* and *absence* must be identified independently at eat time t). Also, as (b) demonstrates, it may be equally inaccurate to treat events uniformly across space (e.g., using a semivariogram to characterize this entire space would be inappropriate).

We now review the algorithm in detail.

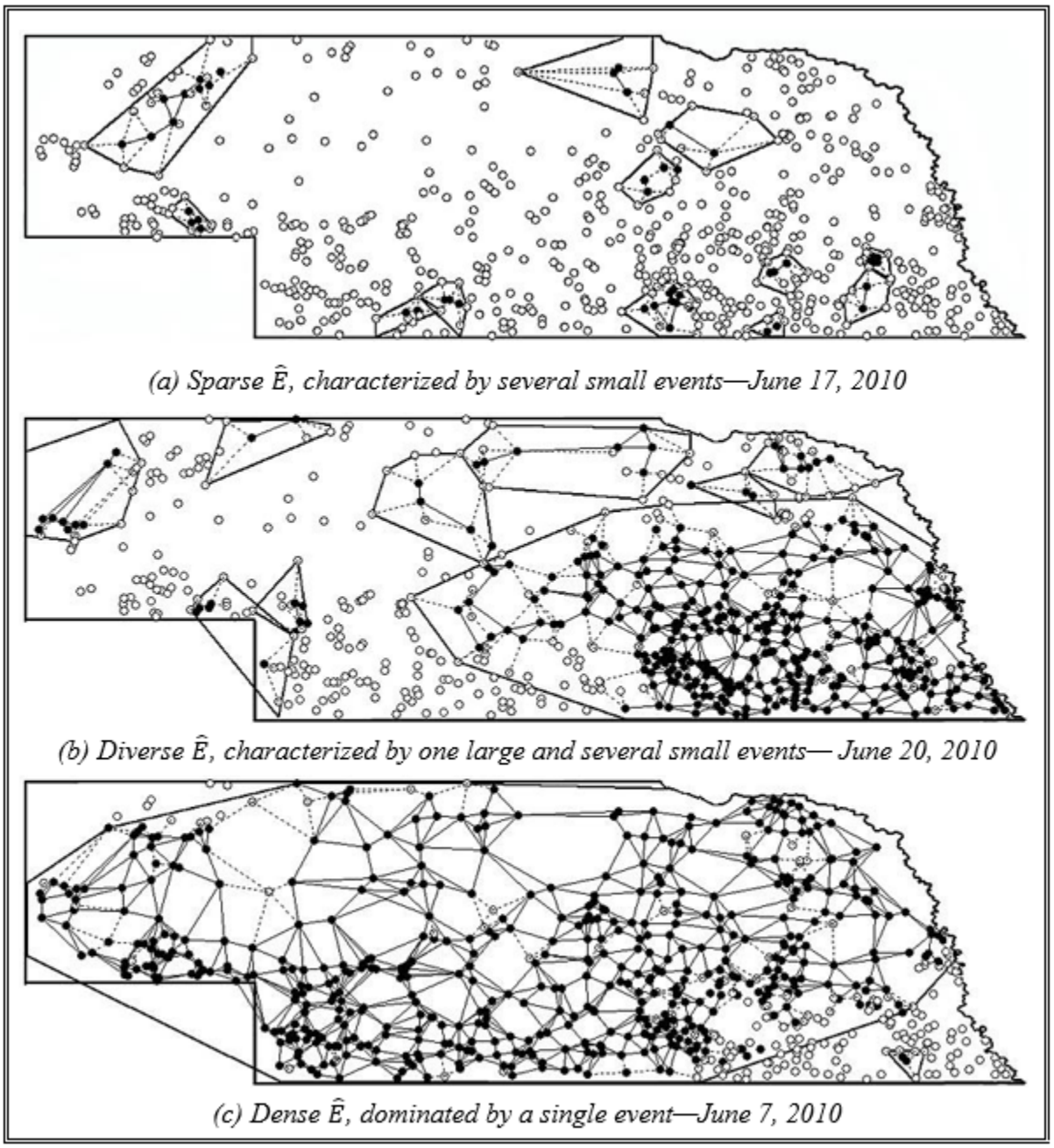


Figure 4.5: The approximated precipitation events ($\hat{E}$) in the state of Nebraska (US) on three different days in June 2010

4.4.2 Initialize embedded graph

The graph is initialized to a completely unconnected graph $G = \langle V, E \rangle$ with the points from P as nodes, i.e. $V =$ `GetPoints(`P`)` and $E \neq \emptyset$ where `GetPoints` returns the points from the dataset P.

4.4.3 Encode spatial relationships

Next, edges are added to encored neighborhood relationships between points. Consider Figure 4.6. Point a is surrounded by several points; which of these points should be considered neighbors—*b-c*, *b-c-d*, *b-c-d-e*?

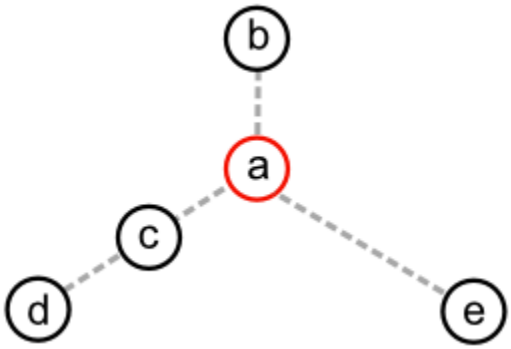


Figure 4.6: Example neighborhood

There exist a number of traditional approaches to finding neighbors. Let us start with two:

1. **Distance threshold:** all points inside of the specified radius are considered neighbors. The threshold is specified *a priori*. Using this method on Figure 4.6, we could arrive at neighborhoods *b-c*, *b-c-d*, or *b-c-d-e* but never just *b* or *c*.

2. **Count threshold:** the nearest k points are considered neighbors. The parameter k is specified *a priori*. Using this method on Figure 4.6, we could arrive at

a variety of neighborhoods. For $k = 3$, then c-b-d; for $k = 1$, we could choose between b and c.

These methods are not robust to differences in point density across the event space, however. The first method can result in wildly different numbers of neighbors depending on the density of nearby points. The second method, instead, can produce neighborhoods of wildly different physical extent—which, for DECAF's purposes, this is not a weakness; in fact, it is exactly the behavior that we want.

Remember, we want to group *present* points for the purpose of discovering events. However, if an *absent* point exists between two *present* points, the two *present* points are presumed to be of separate events. Therefore, it is the *nearest* neighbor that is important—regardless of how close that neighbor actually is.[2] The same information is collected in spare and dense regions: who is my nearest neighbor (and, consequently, is this neighbor *present* or *absent*)?

If the points in P are spatially clustered, this method has problems. For example, consider point a in Figure 4.7 on page 55. If the neighborhood of a is defined to be the four nearest neighbors, all of those neighbors will be located to the west of a. Because we seek to estimate event boundaries, we would like to know something about a's neighbors to the east as well (not to mention north and south). So, instead, we use another traditional method, the sector search.

[2]The semivariogram could be used to set a maximal extent. In section 4.4.5.1, we consider the problem of excessively distant absent points.

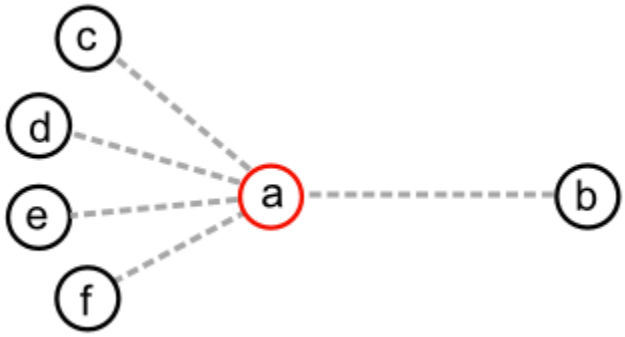


Figure 4.7: Example neighborhood II

3. **Sector search:** the space around the point is divided into sectors (typically of the same size). Either method (1) or (2) is subsequently used to identify the nearest point(s) in each sector. The number of sectors is specified *a priori*. Considering Figure 4.6 once again, we might use *b*-*c*-*d* but never *e*.

Several varieties of sector search were tested for DECAF-EG. These included different numbers of sectors, sectors that were dynamically positioned based on the angle of the closest (according to method 2) neighbor, and sectors that preferred points located towards the sector interior as opposed to the sector edge. We then found that a simple quadrant (north, south, east ,west) nearest-neighbor search was the most robust, consistently producing the most "reasonably" shaped neighborhoods (where reasonableness is understood to reflect a visual aesthetic).

Each neighbor relationship is encoded by an edge in the graph. The value associated with that edge encodes the type of relationship. We identify three types of neighborhood relationship:

1. **Interior** edge: Between *present* points

2. **Border** edge: Between a *present* and an absent point

3. **Exterior** edge: Between *absent* points

The utility of these classifications is revealed in the next section when point clusters are composed using these edge-encoded relationships. Note that the third type of edge relationship is optional; edges between *absent* points need not be added to the graph to find point clusters. In fact, the logic is simpler if we simply treat these relationships as implicit, and so we choose not to add these edges.

4.4.4 Identifying approximated spatial events

The set of spatial events ($\hat{E}$) begins to emerge from the spatial neighborhood graph initialized in the previous section. Clusters of *present* points are connected together, along with adjacent *absence* points (the *border* edge relationship). If these clusters can be identified and formalized, they may make for good event approximations.

To this end, we now refine the definition of an event ($\hat{e}_i$) to be the set points $\{p_1, p_2, , p_m\}$ that form a *connected component* in G, where m is the number of points in the cluster (and $m \leq |G|$). The connected component logic is a variation of the canonical connected components algorithm shown in Algorithm 3 on page 57. The canonical algorithm is based on a traditional breadth-first search. At the top level, the nodes of the graph are considered individually; if a node x has not be visited, then the function `compose_component()` is called. This second function "visits" all nodes accessible from x—that is, all members of the component.

DECAF modifies the canonical connected components algorithm slightly (noted in red in Algorithm 4 on page 58). Recall that in the previous section edges between nodes were classified as either *interior* or *border* (we chose to ignore *exterior* edges), where *interior* referred to an edge between two present nodes and border to an edge between a *present* and an *absent* node. DECAF retrieves this classification in Line

Algorithm 3 Canonical connected components algorithm

```
 1: function COMPOSE_COMPONENT(x, G)
 2:     component ← list()
 3:     Q ← queue()
 4:     Q.add(x)
 5:
 6:     while Q is not empty do
 7:         u ← Q.remove()
 8:         for v ∈ G.adjacent_nodes(u) do
 9:             if not v.visited then:
10:                 component.add(v)
11:                 v.visited ←TRUE
12:                 Q.add_node(v)
13:             end if
14:         end for
15:     end while
16:
17:     return component
18: end function
```

13 of Algorithm 4. If the edge is *interior*, the function behaves normally (neighbor v is added the breadth-first queue). However, if the edge is not *interior*, then neighbor v is treated as a leaf (terminating) node. Neighbor v is still added to the component, but it is not used to find additional nodes.

This logic raises two questions. First, why does DECAF-EG stop at *border* edges? Second, why does DECAF include neighbor x of a *border* edge despite treating that neighbor as a terminator?

Why stop at border edges? A *border* edge indicates a connection to an *absence* node. A node of type *absence* should not connect one $\hat{e}_i$ to another $\hat{e}_j$—that is to say, we do not want to connect two clusters by absent points only. Connections based on absence points logically indicate a break between two unique events (see Figure 4.2 for a visualization of this principle).

Why include border neighbors? From the associations of *present* neighbors emerge

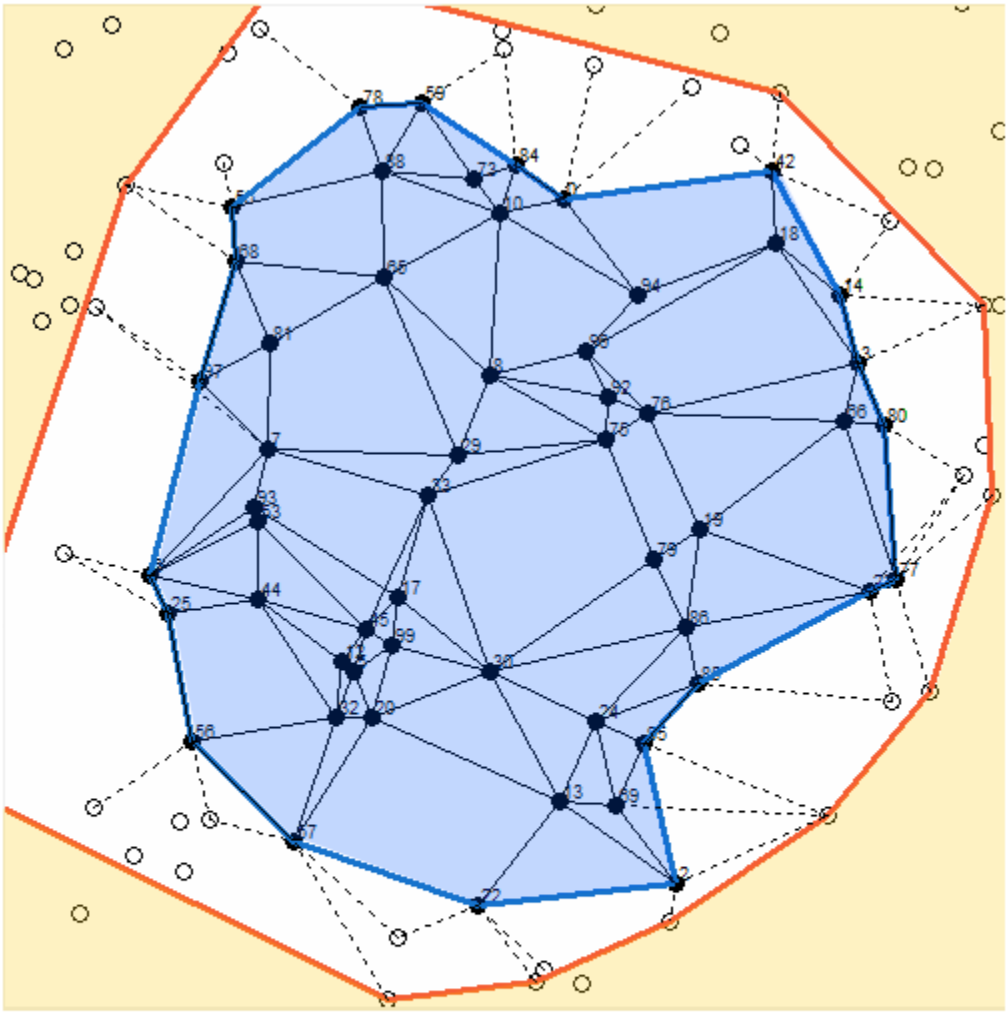
Algorithm 4 DECAF-EG connected components algorithm

```
1: function COMPOSE_COMPONENT(x, G)
2:     component ← list()
3:     Q ← queue()
4:     Q.add(x)
5:
6:     while Q is not empty do
7:         u ← Q.remove()
8:         for v ∈ G.adjacent_nodes(u) do
9:             if not v.visited then:
10:                component.add(v)
11:                v.visited ←TRUE
12:                edge ← G.get_edge(u, v)
13:                if edge.type ==INTERIOR then:
14:                    Q.add_node(v)
15:                end if
16:            end if
17:        end for
18:    end while
19:
20:    return component
21: end function
```

clusters of adjacent *present* points. The hull drawn about these points approximates the minimal extent of the event. The *absent* points adjancent to this hull define the maximum extent of the event. Now, the space is categorized in three ways:

1. **Interior:** Inside a *present* hull, where the event is presumed continuous.
2. **Border:** Outside of a hull but not beyond its *absent* neighbors. This is a region of uncertainty—the event may extend into this region; it may not.
3. **Exterior:** Outside a hull and beyond *absent* neighbors, where it is presumed no event exists.

Figure 4.8 illustrates these three regions by example.



- Black points indicate *presence*; white points indicate *absence*.
- Black lines indicate neighborhood relationship between two *present* points; dashed between a *present* and an *absent* point.
- The blue shaded region is the *interior*. The white shaded region is the *border*. The yellow shaded region is the *exterior*.

Figure 4.8: Illustration of event *interior*, *border*, and *exterior*.

To collapse this space into two categories, we introduce a second bias: overestimation of extent is preferable to underestimation. We would rather suffer false positives

than false negatives. So, all adjacent *absent* points are added to $\hat{e}_i$ such that the hull of $\hat{e}_i$ is an approximation of the maximal extent of the underlying discrete event. We have now composed both minimal and maximal extents, which we denote the *min cover* and the *max cover*, respectively.

$\hat{E}$ is the set of all connected components $\{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_n\}$ which signify the set of events E.

4.4.5 Event refinement

The events derived by the steps described above provide an accurate estimation of the true events. However, they can be further improved using a set of post-processing steps. All of these operations are non-essential, but may improve results in certain domains.

4.4.5.1 Pruning

If *present* point p is adjacent to a sparsely-sampled subregion, certain neighbors of p may be (very) far away. Consider Figure 4.9. Is it still reasonable to associate b with the event, or has the assumption of continuity become tenuous? If a well-known distance of spatial independence is known for the phenomena it may be included in the domain knowledge K, and the neighborhood search can be bounded.

We choose to use a more data-driven alternative based on the distribution of edge-lengths in event $\hat{e}_i$ to identify outliers. The distribution of edges in $\hat{e}_i$ tends to be right-skewed, for the central nodes create a dense network of short edges (limited at zero distance) while the peripheral nodes might be much further away. We therefore use

$$m = \texttt{median}(\hat{e}_i.edges) \tag{4.12}$$

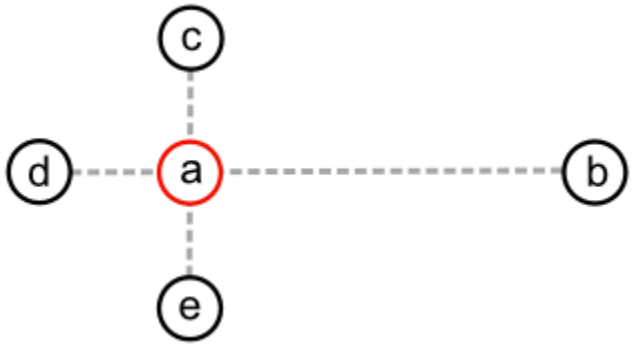


Figure 4.9: Example neighborhood III

and

$$SD = \texttt{StandardDeviation}(\log \hat{e}_i.edges) \tag{4.13}$$

to define the outlier threshold

$$prune\_threshold = m + k \cdot m \cdot SD \tag{4.14}$$

where k is a constant and may be parameterized by K. Edges longer than this threshold are considered outliers and are removed. The above formula was selected because it is very conservative and removes very few edges, and so is effective at removing exceptional outliers and exceptional outliers only. The use of the median and the log-transformed standard deviation enforce this conservativeness, and k simply allows this conservatism to be tweaked according to domain (for the real-world tests in Chapter 9, we use $k = 5$).

Because each $\hat{e}_i$ is treated uniquely, cluster structure may vary across regions. In densely-sampled regions the threshold automatically becomes tighter than in sparsely-sampled ones.

4.4.5.2 Merging

Cluster merging is only applicable to situations where a threshold defines presence/absence. Two clusters may share leaf nodes, and in situations where a non-zero threshold defines presence/absence, those leaves may be associated with trace values (i.e., $\beta \geq v > 0$). In certain domains, it may be logical to merge two events connected by trace values (e.g., a thunderstorm briefly weakens as it moves across the landscape). A possible solution would be to merge the two events once a predefined threshold—either count number or a percentage of shared nodes—is passed such that

$$\hat{e}_k = \hat{e}_i \cup \hat{e}_j \tag{4.15}$$

Our implementation of DECAF-EG, however, uses a nonparameterized merging logic. Given two events $\hat{e}_i$ and $\hat{e}_j$, count the number of shared absent and trace nodes. If trace > absent, merge. This method makes use of all cross-event connection information (all absent and trace nodes), and chooses to merge only when the evidence for connection begins to outweigh separation.

In truth, it is reasonable to speculate that the "most appropriate" merging method varies across problem domains. Demonstrating this is beyond the scope of this paper and is left for future research.

4.4.5.3 Splitting

Just as it may be appropriate to merge two estimated events, it may be equally appropriate to split one estimated event into one or more smaller events. Consider Figure 4.10 on page 63. A single *present* point connects two otherwise distinct clusters. Currently, DECAF-EG does not implement logic to handle this situation. It may be appropriate, however, to use more sophisticated graph connectivity measures

to implement a splitting algorithm in future work.

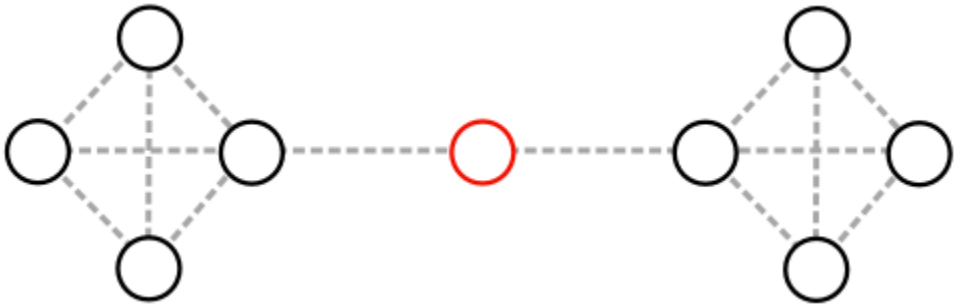


Figure 4.10: Two clusters connected by a single point.

4.4.5.4 Weeding

A cluster may be composed around a single presence point. It may be advantageous to weed out such clusters. A threshold may be established *a priori* for the minimum number of presence points required; for large data sets this may be derived from a distribution of the clusters.

4.4.6 Boundary snapping

We seek to approximate the extent of event e_i (a polygon) using $\hat{e}_i$ (a set of points), which necessitates fitting a hull to the point members of $\hat{e}_i$. To do so, we compute the alpha hull (concave hull) from the points that compose the connected component associated with $\hat{e}_i$. Computations are performed using the same Edelsbrunner Algorithm implementation provided by the R package alphahull used in DECAF-IK.

However, this approach results in a boundary problem. The maximal extent of an estimated event is dependent upon on the points in the connected component, which in turn are dependent upon P. Any estimation point q that falls beyond the maximal spatial extent of P will necessarily fall outside of all estimated events. Because P

is sampled from within our (in practice finite) space, the regions between the edge of the sample space and the nearest $p \in P$ will always be empty. A bias towards absolute absence is untenable.

To resolve this, when a member m of $\hat{e}_i$ is adjacent to the edge of the geographic space, the hull is drawn through the point on the edge closest to m. In effect, we are "snapping" the boundary of the estimated event to the boundary of the event space when there are no intermediate points between the estimated event and the event space boundary.

4.4.7 Event filling

Unlike in DECAF-IK, most of the points located inside of event estimation $\hat{e}_i$ are known without even considering the hull of the event because they are members of the connected component used to estimate $\hat{e}_i$. However, points (typically *absent*) may be interior to the hull of $\hat{e}_i$ without being members of the connected component associated with $\hat{e}_i$. It is only reasonable to add such points to $\hat{e}_i$, as the hull of $\hat{e}_i$ denotes the boundaries of a continuous event.

4.5 Time complexity analysis: DECAF-EG

4.5.1 Developing Spatial Context

The time complexity of initializing the neighborhood graph is dependent upon the neighborhood algorithm used. If a distance matrix is calculated *a priori*, then `find_neighbors()` can be resolved in $O(k \cdot |P|)$ time, where k is the number of neighbors and P is the set of sample points. If a k-d tree is used instead, then `find_neighbors()` can be resolved in $O(k \cdot \log |P|)$. If all points are present then the function is called $|P|$ times.

The resulting time complexity ranges from $O(|P|^2)$ to $O(|P|\log|P|)$ depending on the implementation. Our implementation uses a k-d tree, and so takes $O(|P|\log|P|)$ time.

$\hat{E}$ is constructed using the traditional connected component algorithm, which has time complexity linear to the number of nodes and edges in G. The number of edges is in the worst-case $k \cdot |P|$, so time complexity is $O(k \cdot 2|P|) = O(|P|)$.

Cleaning is optional but we consider the time complexity nevertheless. Pruning examines all edges twice in sequence, once to build the cluster histograms and once to compare, so time complexity is $O(|P|)$. Merging is linear in the number of points shared by clusters along their borders, which must be less then $|P|$. Weeding is linear in $|\hat{E}|$, which also must be less than $|P|$. The time complexity of cleaning is $O(|P|)$.

Boundary snapping is linear in $|P|$; in the worst-case all points are near the border of the space, so time complexity is $O(|P|)$. Event filling is linear in the number of enveloped absent points, which much be strictly less than $|P|$.

Using a k-d tree, we arrive at time complexity

$$O(|P|log|P|) + O(|P|) + O(|P|) + O(|P|) = O(|P|\log|P|) \tag{4.16}$$

4.5.2 Generating V

The method is identical to that used in DECAF-IK (section 4.3.2). The associated time complexity is $O(|n|)$

4.5.3 Incorporating Time

This analysis is similar to the DECAF-IK analysis in section 4.3.3.

If a single moment in time is considered, the time complexity is the total of event

construction and fusion,

$$O(|P|\log|P| + |P|) = O(|P|\log|P|). \tag{4.17}$$

If time is incorporated via a time window τ, where $\tau = (t_s, t_f)$, then the time complexity rises to $O(|\tau| \cdot |P|\log|P|)$. The union operation is linear to the number of points in $\hat{E}_s$ through $\hat{E}_f$. Assuming these are similar, we arrive at

$$O(|\tau| \cdot |P|\log|P|) + O(2 \cdot |P|) = O(|\tau| \cdot |P|\log|P|) \tag{4.18}$$

Given the linear rate of time and the non-linear proliferation of spatial data sources, it is reasonable to argue that in most applications $\tau \ll |P|$.

4.5.4 Aggregated time complexity

The time complexity of DECAF-EG is

$$O(|\tau| \cdot |P|\log|P|) \tag{4.19}$$

and we assume that $\tau \ll |P|$. Therefore, the overall time complexity of DECAF-EG is

$$O(|P|\log|P|) \tag{4.20}$$

Note that the time complexity of DECAF-EG is significantly less than DECAF-IK, which was $O(m^2|P|^3)$ where m^2 is the dimensionality of the *presence/absence* grid.

4.6 Summary

To review, the DECAF algorithm is designed to structure the event space between *present* and *absent* regions in order to selectively use standard interpolation techniques to estimation point values. We hope to improve upon the standard techniques by avoiding overestimation in *absent* regions and reducing underestimation in *present* regions.

The DECAF-IK implementation exploits indicator kriging to differentiate between *present* and *absent* regions. Indicator kriging is a well-established geostatistical method. However, DECAF-IK has high computational complexity.

The DECAF-EG implementation uses an embedded graph approach to effect the *presence*/*absence* estimation. Though this methodology is novel, it has a lower computational complexity than DECAF-IK.

Chapter 5

Simulation Experimental Design

The previous chapter presented two algorithms, DECAF-IK and DECAF-EG, to address the stated problem of fusing geospatial delineated continuous events. In this chapter, we test the efficacy of these methods in response to a number of different factors. The primary goal of these tests is to check the accuracy of the DECAF *presence/absence* (PA) predictions under different conditions. The secondary goal is to test whether the PA logic is improving the accuracy of point-predictions.

The two DECAF algorithms will be compared to one-another as well as two standard approaches, kriging and inverse distance weighting (IDW). To effect this comparison, the methods will be tested through a series of controlled experiments based on simulated data.

5.1 Algorithms

Each of the methods can be parameterized. Furthermore, the terms IDW and kriging are umbrella designations for a wide variety of algorithms. To reduce ambiguity, we pause to further specify each of the methods.

DECAF. Both DECAF methods threshold the data to determine presence and absence. This threshold is 0.1. The interpolation algorithm used was IDW, capped at twelve neighbors.

The two varieties of DECAF were further specified as follows:

DECAF-IK. The alpha hull algorithm was paramaterized with a starting alpha value of 0.05, a failure increment of 0.05, and a max alpha of 0.1 (if this threshold was reached, a convex hull is computed instead). The *presence/absence* raster was computed at a resolution of 100×100.

DECAF-EG. Point neighbors were identified using a static sector search (considered north, south, east, and west). Clusters with fewer than six members were dropped.

IDW. The maximum number of neighbors considered was twelve. This threshold, though arbitrary, is borrowed from the industry standard geographic information system, Esri's ArcGIS [18]. The IDW function in ArcGIS uses twelve as the default cutoff value.

Kriging. Ordinary kriging was used. The semivariogram was fitted with a spherical model. The spherical model performed similar to other standard models—exponential, circular, Gaussian—in exploratory analysis. We did not consider more esoteric models. Furthermore, the literature contends that differences arising because of choice of model are typically dwarfed by other factors, such as data selection [6]. The semivariogram is fitted automatically for each test. We do not parameterize it *a priori*.

5.1.1 Dependent Variables

We are interested in the error associated with the four methods. For both tests, simulation and real-world, the predicted values outputted by the methods will be compared with known values. So, error is known.

We measure error in two ways: *presence/absence* prediction and magnitude of prediction error. The first considers only the accuracy of event presence predictions (i.e., event e_i envelopes location $\langle x, y \rangle$). The second considers the accuracy of the continuous prediction (e.g., predicted 2" of rain at a location that actually experienced 1" of true rain).

5.1.1.1 Presence/Absence

The DECAF approach is predicated upon accurate event detection logic. So, we want to quantify how well each method predicts *present* when the event is *present* and *absent* when otherwise. Because neither IDW nor kriging incorporate explicit *present/absent* logic, we do not expect them to perform well. However, we include them for the sake of comparison.

Prediction effectiveness is measured using the confusion matrix in Table 5.1. The confusion matrix is calculated for each method (DECAF-EG, DECAF-IK, IDW, and Kriging). From these matrices we derive five secondary measures.

Recall. Recall measures the ability of a method to detect present points. It is computed using the formula

$$recall = \frac{PP}{PA + PA} \tag{5.1}$$

Precision. Recall is useful, but it tends to favor methods that use a bias towards *present* (such as DECAF-EG). Precision measures the proportion of predicted

Table 5.1: Confusion matrix

		Predicted		
		Present	*Absent*	*Total*
Ground Truth	*Present*	**PP**	**PA**	PP + PA
	Absent	**AP**	**AA**	AP + AA
	Total	PP + AA	PA + AA	100

PP: Truth is *present*, predicted *present*
PA: Truth is present, predicted *absent*
AP: Truth is *absent*, predicted *present*
AA: Truth is *absent*, predicted *absent*

present points that are, in truth, present. Therefore, it penalizes incorrect *present* predictions. It is computed using the formula

$$precision = \frac{PP}{PA + AP} \tag{5.2}$$

F-score. A high recall may reflect a bias towards *present* points; a high precision may reflect a reluctance to predict *present*. The ideal method would have both high recall and precision. The f-score is a composite derived from the two. It is computed using the formula

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.3}$$

Specificity. Specificity measures the ability of a method to detect *absent* points; a high specificity means that the methods suffers from few false positives. It is computed using the formula

$$specificity = \frac{AA}{PP + AA} \tag{5.4}$$

Table 5.2: Method presence/absence measures

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG	...	...	...	...	...
DECAF-IK	...	...	...	...	...
IDW	...	...	...	...	...
Kriging	...	...	...	...	...

Accuracy. Accuracy is a composite based in part on the specificity. It measures the proportion of true results (the proportion of PP and AA) outputted by a method. It is computed using the formula

$$accuracy = \frac{PP + AA}{AA + AP + PA + AA} \tag{5.5}$$

These results are reported by method in the form of Table 5.2.

We expect the DECAF algorithms to outperform IDW and kriging at PA prediction, as neither IDW nor kriging are intended for this type of prediction.

5.1.1.2 Error Magnitude

In general, it is insufficient to only predict *presence/absence*. Methods that predict *present/absent* with equal effectiveness can be differentiated by the magnitude of error within the *present/absent* blocks. Even if methods predict *present/absent* with differing effectiveness, the question remains: is a method that suffers 10% error within PP preferable to a method that suffers 2% error within AA? The answer depends on domain, but, clearly, magnitude of error must be investigated.

To do so, we consider four measures derived from average difference. Average difference is the difference between the estimated and true value, averaged across all

verification points in the set. We do not consider Average Difference directly due to the problem of averaging out negative and positive values. Instead, we consider:

Absolute Average Difference (AAD). The absolute difference between the estimated and true value, averaged across all verification points in the set. In other words, *what is the average magnitude of error?*

Positive Average Difference (AD+). The average difference between the estimated and true value across only those differences that are positive. In other words, *what is the average magnitude of overestimation error?*

Negative Average Difference (AD-). The average difference between the estimated and the true value across only those differences that a negative. In other words, *what is the magnitude of underestimation error?*

Maximum Absolute Difference (MAD). The maximum difference between the estimated and true values from the set of verification points. In other words, *how poorly does this method predict in the worst-case?*

5.1.2 Experimental Design

We perform simulations in order to purposely control five independent variables: event size, event shape, and event orientation as well as two types of noise in the continuous field associated with the event. We use the simulations to consider how variations along these axes affect the dependent variables outlined above.

5.1.2.1 Independent Variables

The five independent variables can be broken into two groups. The first group characterizes the event boundary: event size, event shape, and event orientation. The second

group characterizes the event interior: uniform noise and spatial autocorrelation noise. Uniform noise is approximately measured by the nugget of a semivariogram, and so we will refer to as the *pseudo-nugget*. Spatial autocorrelation noise is approximately measured by the sill of a semivariogram, and so we will refer to it as the *pseudo-sill*.

5.1.2.2 The simulation process

The simulation process is outlined in Figure 5.1. It can be subdivided into three parts: truth generation, method simulation, and result aggregation.

First, a generator function builds a truth field. This field represents the "ground truth" of the phenomenon. This field can be created at an arbitrary resolution (for our tests, we used a $100x100$ grid).

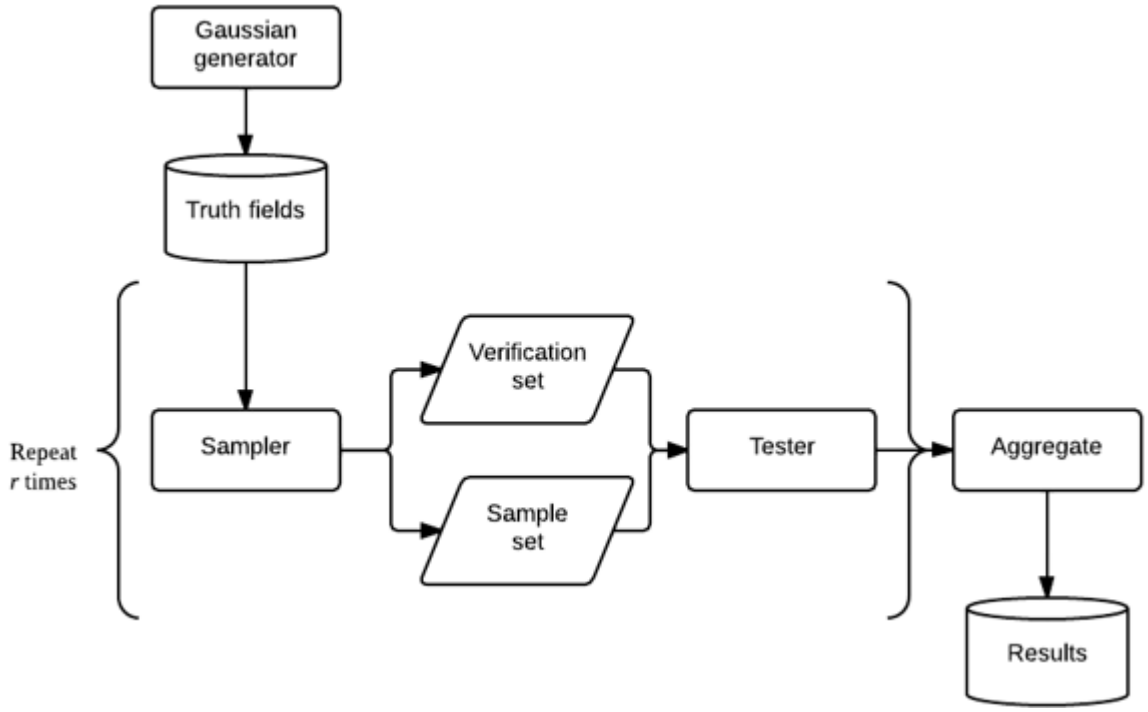


Figure 5.1: The simulation process

Second, two sample sets, a *sample* and a *verification* set, are derived from the truth

field. The sample set is given as the input to each of the four methods (DECAF-IK, DECAF-EG, IDW, and kriging) to be used to estimate values. The verification set defines the locations at which the four methods will estimate a value. The true value is known at each verification location, allowing error to be recorded.

Each of these two sets are constructed identically. First, x values are derived from a uniform distribution using a pseudo-random number generator. Next, y values are found using the same method. These are grouped into pairs. Each pair is associated with a z value, which is derived by sampling the value of the truth field at the associated x, y location. Figure 5.2 shows an example Gaussian event (a small circle with no noise) with three-hundred sample points (colored white) and one-hundred verification points (colored gold).

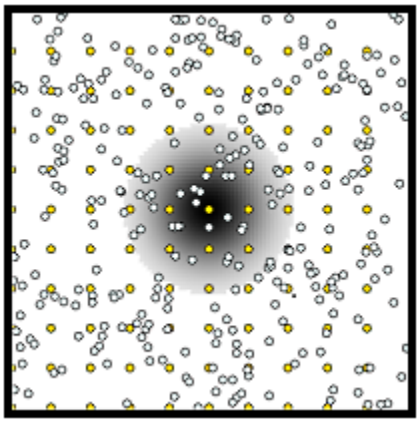


Figure 5.2: Example *sample* and *verification* sets

These sets are fed to each of the four methods. Errors are estimated. This set-construction, method-testing procedure is performed r times for the sake of statistical robustness.

Finally, the results are aggregated and stored for future analysis.

5.1.2.3 Building truth fields

The simulations are derived from the Gaussian distribution; the methodology is outlined in Algorithm 5. The parameter A is the maximum possible z value, the pair x_0, y_0 encodes the center point of the event, and the values θ_x and θ_y are the standard deviations of the x and y axes, respectively. A is manipulated to model phenomenon magnitude, x_0, y_0 for phenomenon location, θ for phenomenon orientation, and θ_x, θ_y for phenomenon shape. The algorithm returns a vector of (x, y, z) pairs, where x and y are the coordinate values and z is that associated value.

Algorithm 5 Generate Gaussian Event

1: **function** GENERATE($A, x_0, y_0, \theta, \sigma_x, \sigma_y$)
2: $a \leftarrow \frac{cos^2\theta}{2\sigma_x^2} + \frac{sin^2\theta}{2\sigma_y^2}$
3: $b \leftarrow \frac{-sin^2 2\theta}{4\sigma_x^2} + \frac{sin^2\theta}{4\sigma_y^2}$
4: $c \leftarrow \frac{sin^2\theta}{2\theta_x^2} + \frac{cos^2\theta}{2\sigma_y^2}$
5:
6: $X \leftarrow$sequence($x\_min, x\_max, increment$)
7: $Y \leftarrow$sequence($y\_min, y\_max, increment$)
8: $Z \leftarrow$list()
9:
10: **for** x, $y \in X$, Y **do**
11: $u \leftarrow a \cdot (x - x_0) \cdot (y - y_0) + c \cdot (y - y_0)^2$
12: $z \leftarrow A \cdot e^u$
13: Z.append(z)
14: **end for**
15:
16: **return** (X, Y, Z)
17: **end function**

In short, the algorithm builds the values for z by treating the x and y axes as two Gaussian distributions. The result is an elliptical shape in the $xy-$plane and a Gaussian shape in the $xz-$ and $yz-$planes. Figure 5.3 on page 77 shows an example output.

Post-generation, we can apply a threshold to create a definite event boundary.

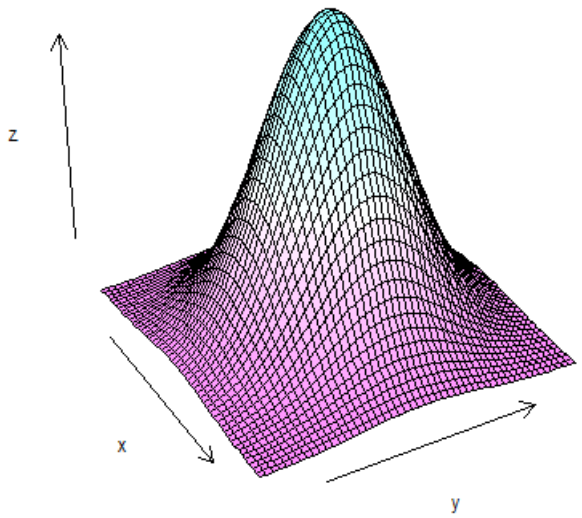


Figure 5.3: Two-dimensional Gaussian distribution

Figure 5.4 on page 78 shows a thresholded two-dimensional Gaussian distribution.

At this point, we have simulated an event that can be described as behaving delineated continuouslya discrete event border is identifiable, and the interior of that event can be modelled as a continuous phenomenon.

Finally, noise can be added to the distribution. Noise is added by iterating through all presence points and adding a value sampled from the normal distribution $N(\theta, \sigma^2)$, where σ^2 is parameterized according to the particular test.

We choose to add two types of noise, pseudo nugget and pseudo sill noise. The first is uniform noise, noise that behaves uniformly across (or independent of) space. In this case, σ^2 is set as 5% of A (the maximum value) for "low" noise and 10% for "high" noise. A two-dimensional Gaussian distribution with added pseudo-nugget noise is shown in Figure 5.5 on page 78.

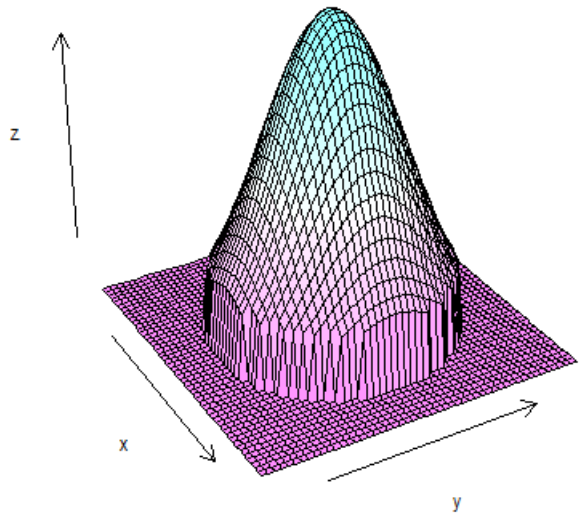


Figure 5.4: Thresholded two-dimensional Gaussian distribution

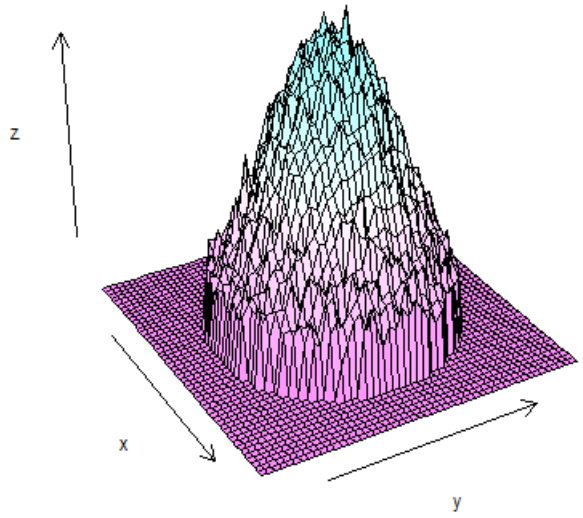


Figure 5.5: Thresholded two-dimensional Gaussian distribution with pseudo-nugget noise

5.1.2.4 Simulation experimental design

The experimental design is subdivided into three parts. First, we consider the effects of event structure (size, shape, and orientation) on method accuracy. Second, we

consider the effects of event noise. Finally, we examine interaction effects between event structure and event noise.

Event Structure We examine the effects of event structure by manipulating three treatments: size, shape, and orientation. For each treatment, we consider two factor levels: large and small (size), circle and ellipse (shape), and $\theta = 0$ and $\theta = \frac{\pi}{4}$ (orientation). Combining the factors, and we arrive at six structures. These are presented in Table 5.3 on page 80.

We will use these six designs to explore the effect of event size, shape, and orientation on the predictive qualities of DECAF-IK, DECAF-EG, IDW, and kriging. Interactions between the three factors will also be considered.

Event Noise We examine the effects of noise by manipulating two treatments: pseudo-nugget and pseudo-sill. For each treatment, we consider three factor levels: none, low, and high. As with structure, we use a combinatorial experimental design. The design is outlined in Table 5.4 on page 81. The resulting structures are shown in Table 5.5 on page 82.

Structure-Noise Interactions Finally, we wish to consider whether there exists interactions between the noise and structural elements. A brute-force approach requires $6 \times 9 = 54$ combinations. To reduce this space, we restrict the noise table to two levels per noise factor (none, low, high $\leftarrow$ none, high). The reduced noise space is shown in Table 5.6 on page 83.

The resulting design requires $6 \times 4 = 24$ combinations. Many of these combinations are created for the structure-only and noise-only tests. Of these twenty-four, fifteen are new.

Table 5.3: Size and shape combinations

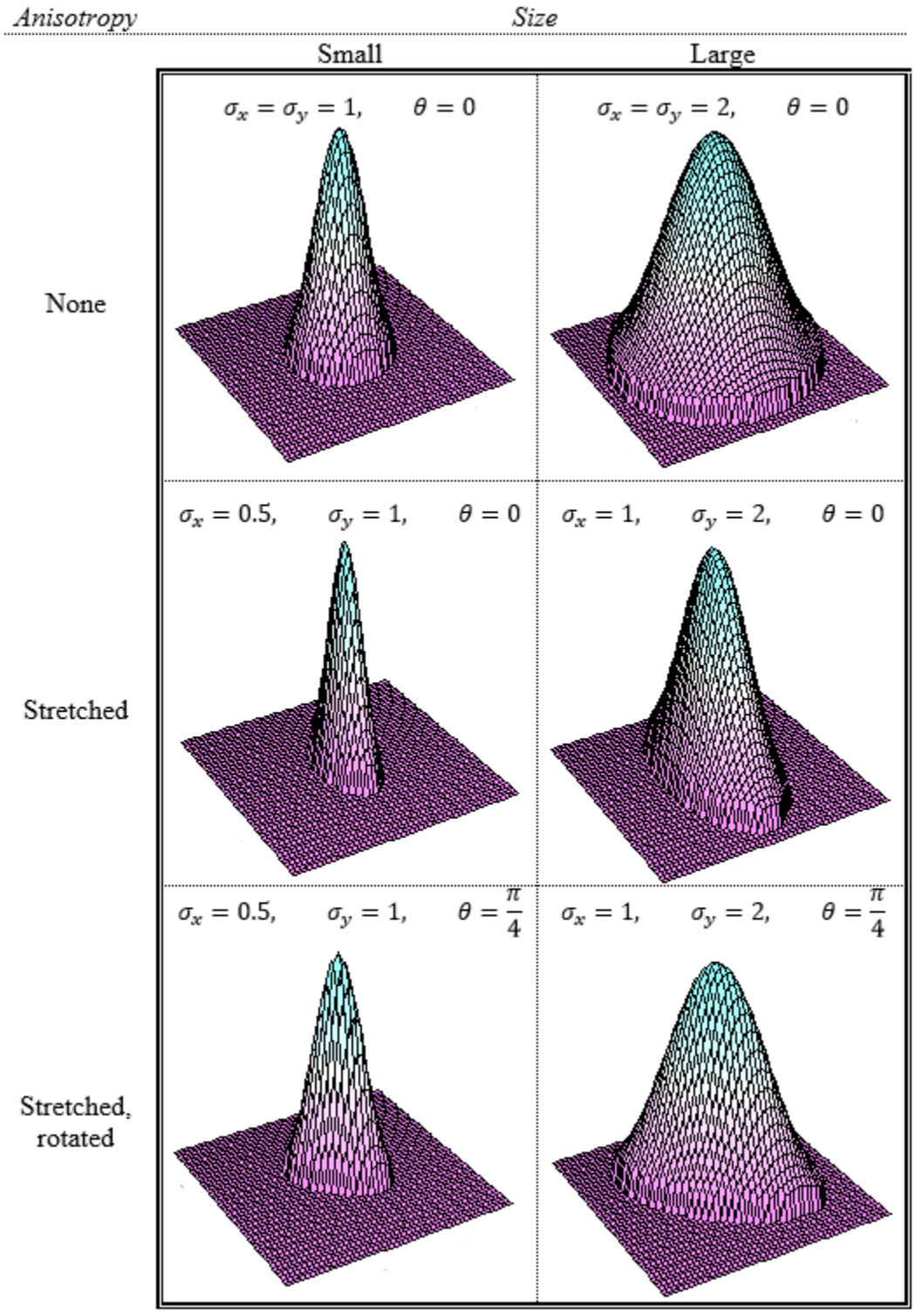


Table 5.4: Noise combinations

Sill \ *Nugget*	**None**	**Low**	**High**
None	No noise	Low nugget	High nugget
Low	Low sill	Low nugget, low sill	High nugget, low sill
High	High sill	Low nugget, high sill	High nugget, high sill

Aggregate In total, there are twenty-nine unique simulation structures. These are outlined in Table 5.7 on page 84. Simulations 1-6 address structure only, simulations 7-14 model noise, and simulations 15-29 consider interactions between structure and noise.

Table 5.5: Noise combinations illustrated

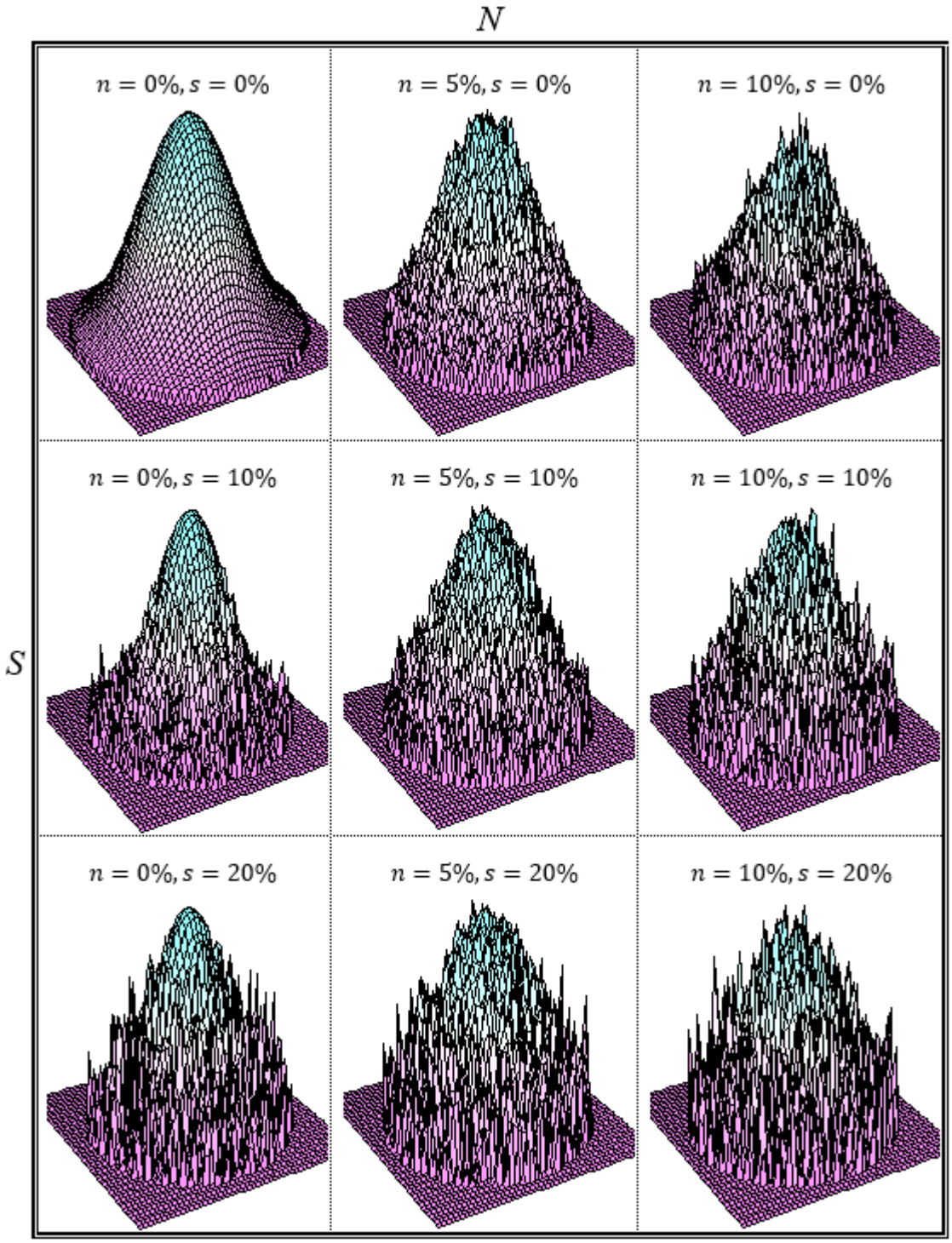


Table 5.6: Restricted noise combinations

		Nugget	
		None	**High**
Sill	**None**	No noise	High nugget
	High	High sill	high nugget, high sill

Table 5.7: Experiment design

	Size	*Shape*	*Orientation*	*Sill*	*Nugget*
1	small	circle	0°	none	none
2	small	stretched	0°	none	none
3	small	stretched	45°	none	none
4	large	circle	0°	none	none
5	large	stretched	0°	none	none
6	large	stretched	45°	none	none
7	large	circle	0°	none	low
8	large	circle	0°	none	high
9	large	circle	0°	low	none
10	large	circle	0°	low	low
11	large	circle	0°	low	high
12	large	circle	0°	high	none
13	large	circle	0°	high	low
14	large	circle	0°	high	high
15	small	circle	0°	none	high
16	small	circle	0°	high	none
17	small	circle	0°	high	high
18	small	stretched	0°	none	high
19	small	stretched	0°	high	none
20	small	stretched	0°	high	high
21	small	stretched	45°	none	high
22	small	stretched	45°	high	none
23	small	stretched	45°	high	high
24	large	stretched	0°	none	high
25	large	stretched	0°	high	none
26	large	stretched	0°	high	high
27	large	stretched	45°	none	high
28	large	stretched	45°	high	none
29	large	stretched	45°	high	high

Chapter 6

Simulation Results

The tests on the simulated data were designed to better elucidate the comparative strengths and weaknesses of the four methods. As per the previous chapter, a total of 29 simulated experiments were undertaken, each resulting in five outputs (presence-absence and four varieties of error magnitude) for each of the four methods—580 distinct tests. Reporting each and every test in exhaustive detail would be prohibitive. Therefore, we will begin by reporting patterns that tend to hold true across tests. Then, relevant results will be presented to answer the three questions that the simulations were designed to answer:

1. Does event structure impact the comparative performances of the four fusion methods in question (DECAF-IK, DECAF-EG, IDW, and kriging)?

2. Does event noise impact the comparative performances of the four fusion methods?

3. Finally, do there exist interaction effects between event structure and event noise that impact the comparative performances of the four fusion methods?

Tables and figures are judiciously presented in this chapter (and there are still many). A more thorough collection of tables and figures can be found in the Appendices.

6.1 Generalized results regarding the individual methods

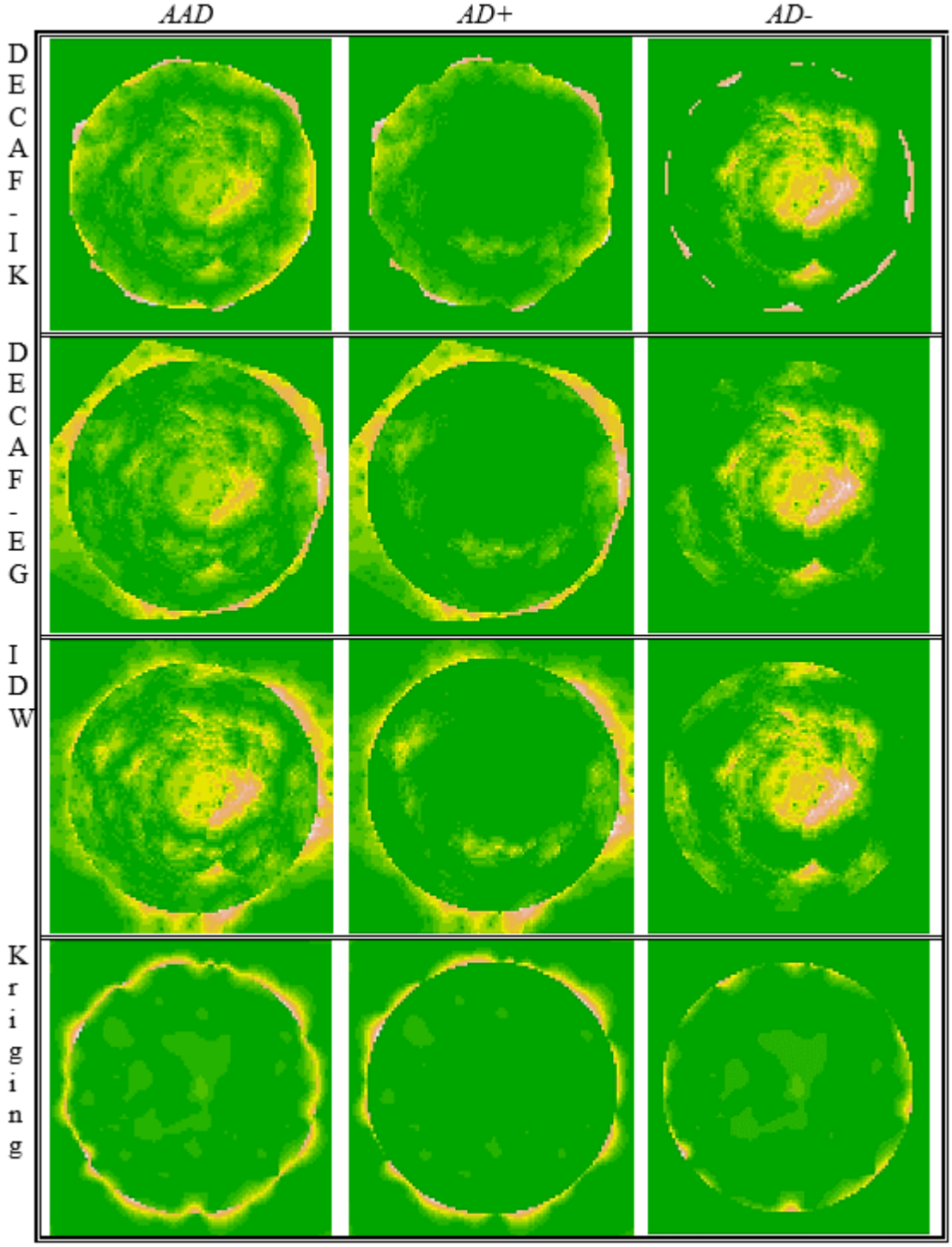
Table 6.1 on page 87 shows twelve error maps. The maps are derived from Simulation 4 (as reported by Table 5.7 on page 84), where events are modelled as large circles. Each row corresponds to one of the four tested methods (DECAF-EG, DECAF-IK, IDW, and kriging). Each column corresponds to one of the measurements of error magnitude: average absolute difference (AAD), average positive difference ($AD+$), and average negative difference ($AD-$). AAD reports strict error magnitude, ignoring direction; $AD+$ reports method overestimation; $AD-$ reports method underestimation. We do not explicitly present maximum absolute difference (MAD), though it is implicit in the maps provided: MAD is the hottest colored (maximum) point on the AAD map.

Before continuing, a couple notes must be made about these maps:

1. Maps do *not* report the results of the aggregated 100 simulations; instead, they report a single illustrative example run. They are naturally nosier then the aggregated results, of course, and therefore care should be taken to not generalize from idiosyncrasies.

2. Cool colors (green) indicate low error, warm colors (yellow, orange) indicate moderate error, and hot colors (red, pink, white) indicate high error. However, the color scale of individual maps is based upon the data contained within each

Table 6.1: Error maps derived from a single simulation run on large circles



map. Therefore, the scale is not necessarily common across maps—while white indicates high error in all maps, the exact magnitude associated with white varies (this is an artifact of the inherent limitations of the tool used to generate them). We do not consider this a problem for two reasons: first, methods experience similar maximum errors, so the scales are not wildly different; second, we use the maps simply to illustrate *where* different methods suffer problems—we study the magnitude of those problems later.

Despite their limitations, the maps illustrate certain trends that repeated in test-after-test and are supported by the aggregated results. Let us now consider each method in turn.

DECAF-IK. DECAF-IK is conservative when predicting present points. This has several effects. The first is visible in the $AD-$ map in Figure 6.1. A haphazard (but small) ring of underestimation is visible. These are locations that fall outside of the event extent estimated by DECAF-IK and so were estimated to be zero.

DECAF-IK takeaway I: DECAF-IK suffers some underestimation error on the event periphery as a result of its conservative bias.

In the same map, observe that underestimation is pronounced in the event center. Here we witness a problem inherent to all interpolation techniques: extremes are underestimated. The maximum values of our Gaussian event are located at the center; ergo, underestimation happens here.

DECAF-IK takeaway II: DECAF-IK suffers underestimation at the event center when the center is associated with the maximum value.

Now consider the $AD+$ map. Overestimation error is evident on the immediate interior of estimated event border. When the unknown point q falls inside of the event boundary, DECAF-IK uses IDW to perform the estimation. It parameterizes IDW with the sample points that fall inside this boundary (all points outside are ignored). Therefore, IDW suffers from a border problem. Estimations near the border must rely on values from the interior, which by the Gaussian nature of the event must be larger. Absent points on the exterior are not considered, so the gradient is lost. Consider Figure 6.1.

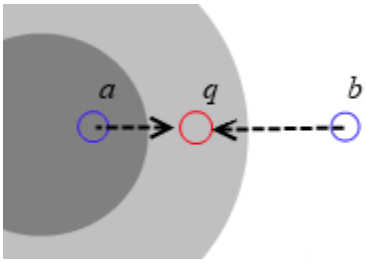


Figure 6.1: The simulation process

Points a and b are the source points (the points used by the method to make a prediction) and point q is the point we want to estimate. The background color indicates the magnitude of the effect (dark gray, high; light gray, low; white, absent). Suppose DECAF-IK successfully detects the border between *present* and *absent*; then, DECAF-IK will only use point a and will overestimate q as high. IDW will include the absent point c, offsetting the effect of point a.

DECAF-IK takeaway III: DECAF-IK suffers from event overestimation at the immediate interior of the event.

DECAF-EG. Start with the $AD-$ map. First, note that the exterior underestimation has disappeared. DECAF-EG is comparatively liberal at estimating event

extent, and so tends to include most or all of the event. Second, note that the interior is underestimated.

DECAF-EG takeaway I: DECAF-EG suffers underestimation at the event center when the center is associated with the maximum value.

Move to the $AD+$ map. The distinctive feature of this map is the obvious estimated event border. DECAF-EG extends the borders of its event estimation to the closest known *absence* points. The region between known *present* source points and known *absence* source points is a region of uncertainty; inside the *present* points the event may be presumed to be present while outside the *absent* points is may be presumed to be absent, but precisely where each region begins and ends is a source of confusion. However, DECAF-EG only uses points interior to the estimated event boundary to estimate at point locations interior to that same boundary. As a result, counterweighting absence points are ignored. The result is region of overestimation error.

DECAF-EG takeaway II: DECAF-EG suffers from event overestimation between the exterior of the true event and the immediate interior of its event estimation.

IDW. In many respects, IDW appears very similar to DECAF-EG—which, is not surprising considering that it is a component of the DECAF-EG method. In fact, the $AD-$ maps tend to be almost indistinguishable. The central region of underestimation error should be familiar by now.

IDW takeaway I: IDW suffers underestimation at the event center when the center is associated with the maximum value.

While IDW doesnt have the jagged overestimation border visible in DECAF-EG's $AD+$ maps, it tends to suffer from a significant overestimation corona

that extends from the true event boundaries.

IDW takeaway II: IDW tends to overestimate regions near the event but outside of the event boundaries.

Kriging. The kriging $AD-$ map is interesting. Compared to the other three methods, kriging doesnt suffer the same underestimation problems to the same degree—at least for *large* events. However, consider Table 6.2 on page 92, which reports $AD-$ maps for the four methods.

Kriging takeaway I: Kriging is less susceptible to underestimation at event centers when the center is associated with the maximum value—at least for larger events.

Next, notice the thin halo of underestimation along the event interior. While not as pronounced as DECAF-IK, it tends to be more pronounced than IDW and DECAF-EG.

Move to the AD+ map. As with IDW, an overestimation corona is distinctly visible around the event exterior. While this corona is of comparable magnitude to IDW immediately adjacent to the event border, it does not extend nearly as far into the *absent* space beyond.

Kriging takeaway II: Kriging tends to overestimate regions near the event but outside of its boundaries, but not as severely as IDW.

Magnitude is one measure of error to consider. The other is *presence/absence* (PA) prediction. Table 6.3 reports the PA results for the large circle simulations (the aggregated results for simulations of the variety shown in Table 6.1).

Ignore the magnitude of these various measures; they can differ from one simulation design to another, as we will see later. For the moment, consider only the

Table 6.2: $AD-$ maps for small circles

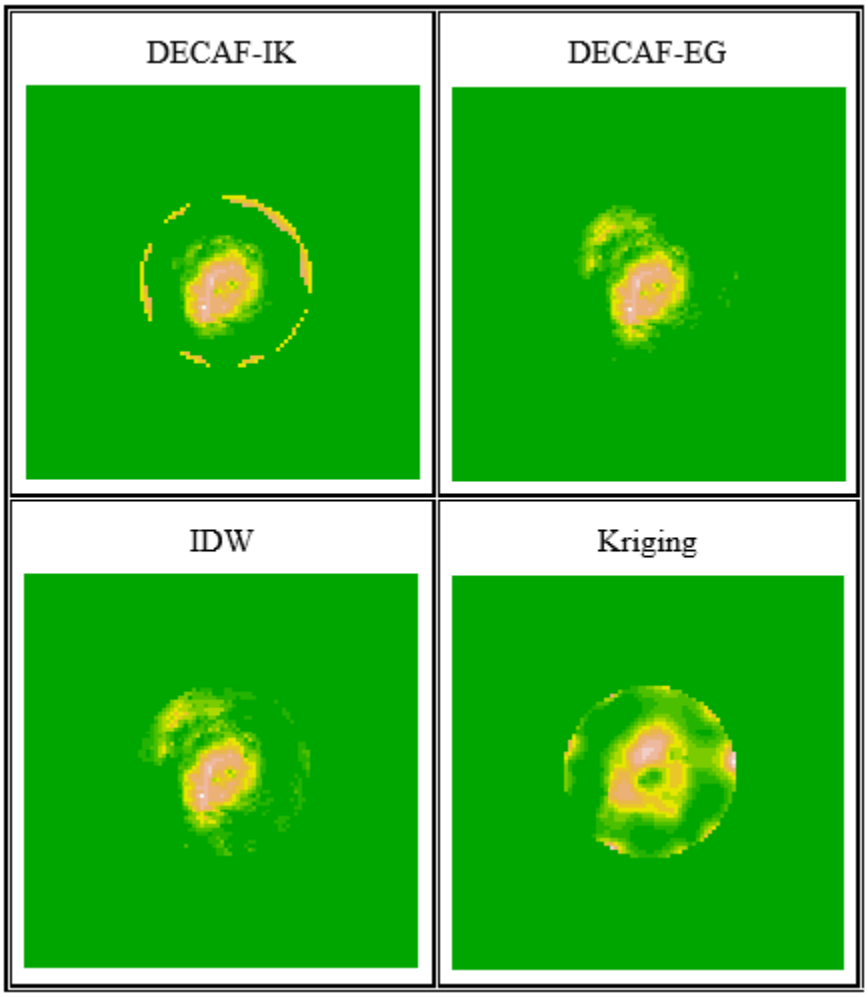


Table 6.3: Presence/Absence, large circle

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG	1.00	0.80	0.89	0.67	0.86
DECAF-IK	0.96	0.98	0.97	0.98	0.96
IDW	1.00	0.65	0.79	0.30	0.70
Kriging	1.00	0.72	0.84	0.49	0.78

relative positions of the methods to one-another; these remain remarkably consistent across tests.

The liberal tendencies of DECAF-EG, IDW, and kriging are captured by Recall. Recall measures the ability of a method to correctly identify a *present* point as *present*. These methods tend to predict *present* in lieu of information to the contrary, and so are good (very good in the case of well-behaved Gaussian events) at detecting *present* points. DECAF-IK's comparatively conservative nature hurts its recall score, but the impact is limited.

The liberal tendencies of DECAF-EG, IDW, and kriging are penalized by the Precision measure. Precision measures the proportion of predicted *present* points that are, in fact, *present*. IDW and kriging tend to do the worst (the two trade places from one test to another). DECAF-EG does better, but DECAF-IK tends to do very well.

Specificity measures the ability to detect *absent* points. DECAF-IK continues to be dominant while IDW and Kriging do poorly and DECAF-EG remains sandwiched in-between.

The F-score, a composite of the Recall and Precision measures, summarizes the above: DECAF-IK is very effective at differentiating between *present* and *absent* points. The Accuracy score is but further evidence.

Presence/Absence takeaway: DECAF-IK performs excellently. DECAF-EG does well. As expected, IDW and Kriging range from decent to terrible, depending on the exact measure used.

Table 6.4: Experiment Design, event structure

	Size	*Shape*	*Orientation*	*Sill*	*Nugget*
1	small	circle	0°	none	none
2	small	stretched	0°	none	none
3	small	stretched	45°	none	none
4	large	circle	0°	none	none
5	large	stretched	0°	none	none
6	large	stretched	45°	none	none

6.2 Event Structure

For event structure, we consider simulation types 1-6. The summary of these simulation types is given in Table 6.4.

Table 6.5 on page 95 reports the results of the analysis of variance test (ANOVA). The response variable for this test is absolute average difference (AAD). This test demonstrates that event orientation has no statistically significant effect on AAD. However, all remaining terms are reported as significant, warranting further investigation (the $size \times shape$ interaction is also insignificant, but the three way interaction between size, shape, and method demands its inclusion).

Furthermore, these results are consistent across dependent variables positive average distance ($AD+$) and maximum absolute difference (MAD) (the ANOVA tables can be found in the Appendices). Negative average distance ($AD-$) reports orientation as significant, so we will consider orientation in that case.

Table 6.5: Results of ANOVA (AAD)

	DF	Sums of Squares	Mean of Squares	F Value	P value
size	1	0.0433	0.0433	2043	$< 2.2e^{-16}$ ***
shape	1	0.0015	0.0015	72.8	$< 2.2e^{-16}$ ***
theta	1	0.0000	0.0000	0.86	0.35
method	3	0.0620	0.0207	975	$< 2.2e^{-16}$ ***
size * shape	1	0.0000	0.0000	1.79	0.18
size * theta	1	0.0000	0.0000	0.61	0.43
size * method	3	0.0091	0.0030	142	$< 2.2e^{-16}$ ***
shape * method	3	0.0008	0.0003	12.6	$< 3.8e^{-8}$ ***
theta * method	3	0.0000	0.0000	0.428	0.73
size * shape * method	3	0.0016	0.0005	25.1	$5.3e^{-16}$ ***
size * theta * method	3	0.0001	0.0000	1.14	0.33
Residuals	2382	0.051988	2.18E-05		

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

Table 6.6: Presence/Absence measures, small circles

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG	1.00	0.73	0.84	0.94	0.95
DECAF-IK	0.91	0.97	0.94	0.99	0.98
IDW	1.00	0.51	0.68	0.84	0.86
Kriging	1.00	0.28	0.43	0.56	0.62

6.2.1 Event size

6.2.1.1 PA accuracy

Let us first consider the effectiveness of presence/absence prediction across event sizes. Table 6.6 summarizes the *small* size results; Table 6.7 reports on size *large* (both appear on page 95.

Table 6.7 should be familiar; it is the same as Table 6.3 from page 92. Note that the generalized observations hold true in Table 6.6. DECAF-IK continues to impress,

Table 6.7: Presence/Absence measures, large circles

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG	1.00	0.80	0.89	0.67	0.86
DECAF-IK	0.96	0.98	0.97	0.98	0.96
IDW	1.00	0.65	0.79	0.30	0.70
Kriging	1.00	0.72	0.84	0.49	0.78

DECAF-EG is a notable second, and IDW and kriging perform comparatively poorly.

Size has an impact on certain PA prediction measures. Recall is effectively unchanged across size, but precision (and, hence, F-score) increases. Note that this may be less a function of size and more a result of the proportion of P compared to A in the dataset. The large event occupies well over fifty percent of the total event space—so, methods that tend to predict P will naturally improve their precision score. This hypothesis is supported by sharp drop in specificity scores: DECAF-EG, IDW, and kriging all struggled to avoid false positives.

6.2.1.2 AAD

DECAF-IK is the clear winner at PA prediction, but the results presented in Table 6.8 show that this does not necessarily translate into a reduction of error magnitude. The table reveals two distinct groups: the three IDW-based methods (DECAF-IK, DECAF-EG, and IDW) are outperformed by kriging.[1]

The interaction plot in Figure 6.2 illustrates the table results (for future results, we will use the interaction plot only).

Clearly, smaller events result in less AAD (average absolute error). This is not necessarily interesting, however, as the large empty spaces in the *small* event space

[1]Individual means and confidence intervals are computed using a t-test. Statistical differentiation is determined using Tukey's Honest Significant Difference test)

Table 6.8: Results for t-test on AAD by method at large and small sizes

		Mean	Confidence Interval	Statistical grouping
Small	DECAF-IK	0.018	(0.0162, 0.0191)	a
	DECAF-EG	0.017	(0.0159, 0.0187)	a
	IDW	0.014	(0.0134, 0.0157)	
	Kriging	0.007	(0.0064, 0.0072)	
Large	DECAF-IK	0.027	(0.0245, 0.0269)	
	DECAF-EG	0.028	(0.0270, 0.0291)	b
	IDW	0.028	(0.0269, 0.0268)	b
	Kriging	0.010	(0.0098, 0.0105)	

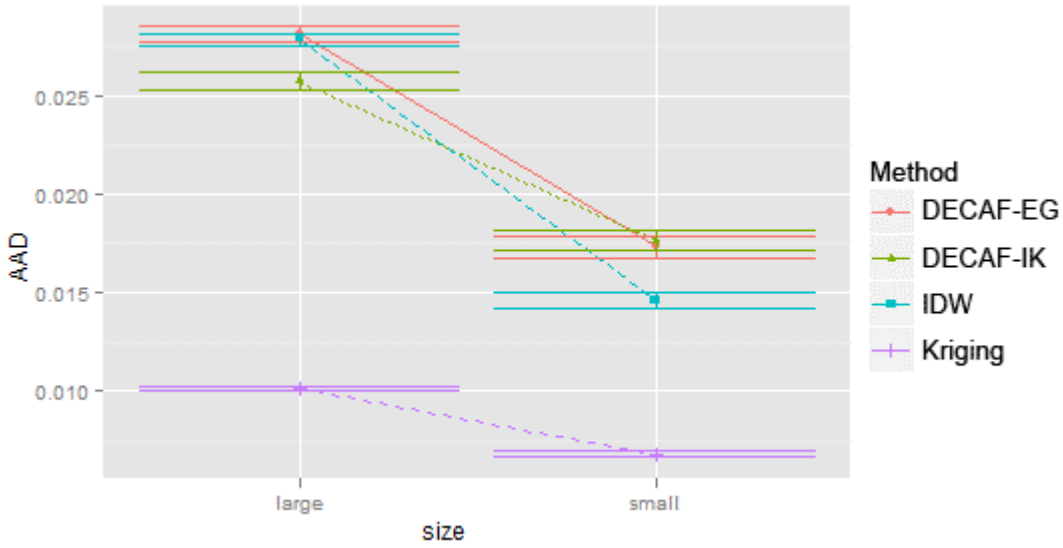


Figure 6.2: AAD method by size interaction plot

will naturally reduce error. More interesting, however, is the behavior of DECAF-IK, DECAF-EG, and IDW relative to one-another. At the large scale, DECAF-EG and IDW are statistically indistinguishable, as may be expected. DECAF-IK slightly outperforms both at this scale. At the small scale, however, IDW slightly outperforms the two DECAF methods, which become indistinguishable.

6.2.1.3 AD+

The *AAD* measure is agnostic to the direction of error. To better understand the effect of event on size the overestimation of error, let us consider the interaction plot for *AD+* (Figure 6.3).

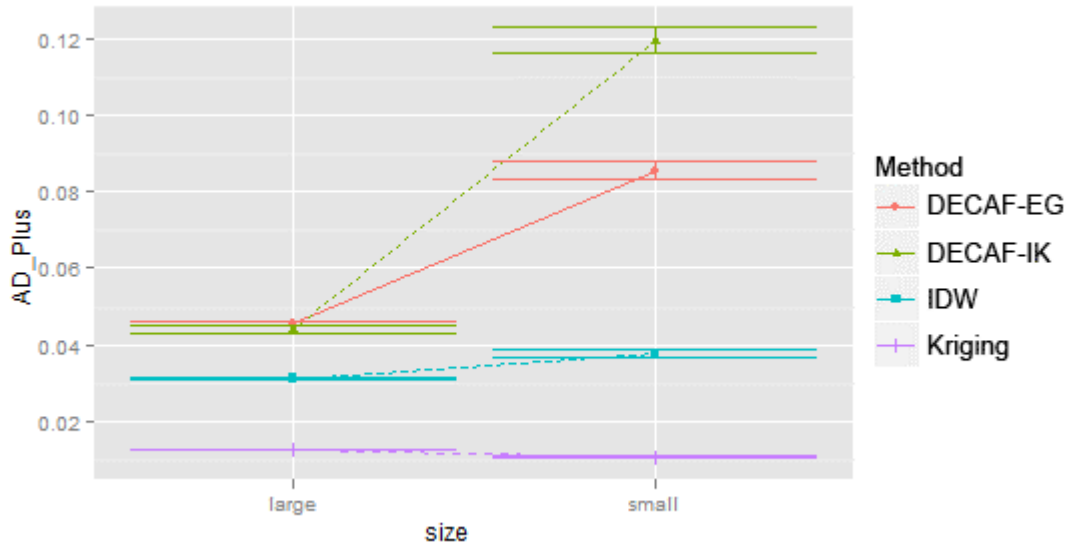


Figure 6.3: AD+ method by size interaction plot

Kriging suffers almost no estimation error; IDW suffers significantly more—both have statistically significant interaction effects, but the magnitude of the differences is too minor to warrant much attention. The DECAF methods are more interesting. Both suffer some error at the large scale, but the two differentiate at the small scale. DECAF-IK, despite its excellent *present/absence* predictions, strongly (relative) overestimates small events. DECAF-EG suffers a similar problem, but not to the same degree.

6.2.1.4 AD-

Now consider underestimation error. The interaction plot, Figure 6.4, shows that all methods struggle with smaller events (note that the y-axis is negative: lower values are larger underestimation errors). Remember the discussion from the start of the section, where we noted that DECAF-IK, DECAF-EG, and IDW tended to underestimate the center of Gaussian events—this clearly holds true. We also noted in that same discussion that kriging was more prone underestimate at smaller rather than larger scales—this also holds true, though kriging's underestimation error remains much smaller than that of its peers. DECAF-IK suffers from the larger $AD-$; clearly, it is failing to leverage its excellent event detection rate.

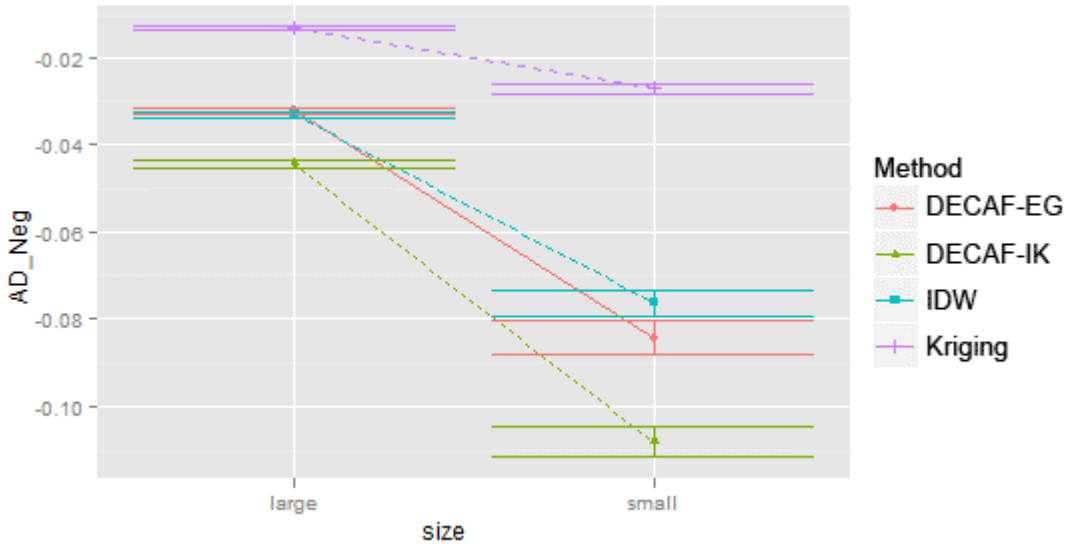


Figure 6.4: AD- method by size interaction plot

6.2.1.5 MAD

The interaction plot for worst-case errors reveals the familiar KrigingIDWDECAF-EGDECAF-IK order of ascending error (Figure 6.5). All methods do worse on small

Table 6.9: Presence/Absence measures, differences between small circles and ellipses

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG	0.00	-0.09	-0.06	0.02	0.01
DECAF-IK	-0.05	-0.04	-0.05	0.00	0.01
IDW	0.00	-0.13	-0.13	0.03	0.02
Kriging	0.00	-0.14	-0.18	-0.03	-0.06

events; the consistency of error increase across methods is interesting.

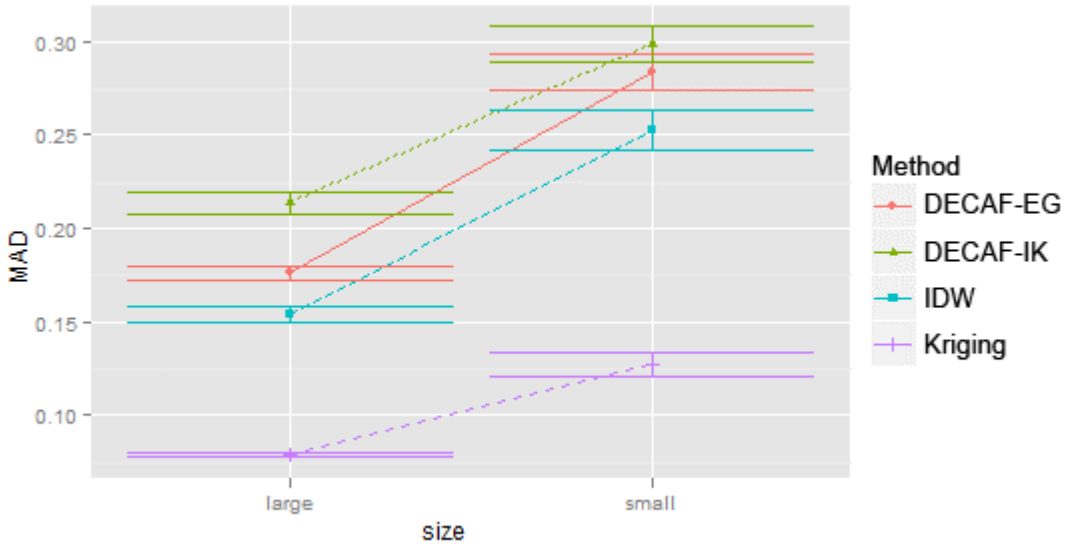


Figure 6.5: MAD method by size interaction plot

6.2.2 Event shape

6.2.2.1 PA accuracy

Let us begin by analyzing the effectiveness of *presence*/*absence* prediction across event shapes. Tables 6.9 and 6.10 may look similar to Tables 6.6 and 6.7 on page 95, but be careful—they are constructed differently and report different results.

Table 6.10: Presence/Absence measures, differences between large circles and ellipses

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG	0.00	-0.07	-0.04	0.18	0.04
DECAF-IK	-0.03	-0.01	-0.02	0.01	0.01
IDW	0.00	-0.09	-0.07	0.38	0.07
Kriging	0.00	-0.24	-0.19	0.07	-0.09

Table 6.9 concerns small shapes. It was constructed by building a PA measure table (just like Table 6.6, but for ellipses instead of circles); then, the values in Table 6.6 were subtracted from these to produce a different (this) table. Table 6.9 reports the difference in PA measures between shapes (ellipse, circle) at the same size (small). Table 6.10 reports the same, but for large sizes. (Because there was a $size \times shape$ interaction, we must compare at both sizes).

So, what can we gather from these tables? First, Recall doesnt change—DECAF-EG, IDW, and Kriging all continue to have perfect Recall, and DECAF-IK continues to be close. Precision falls in *all* cases, indicating that the number of false positives increases with ellipse-shaped events. Note that DECAF-IK is effected the least, however, and kriging is effected the most. The F-Scores reflect this fact, with the kriging F-score plummeting for both sizes.

Specificity is the most interesting measure, remaining effectively unchanged for smaller shapes but exhibiting erratic behavior at larger scales. IDW improves its performance from a dismal 0.3 for circles to a remarkable 0.68 for ellipses; DECAF-EG improves by an impressive 0.18 from a mediocre 0.67 to a respectable 0.85. This behavior is probably the result of the fact that the ellipse takes up less of the total event space than does a circle; therefore, there are simply more *absence* points that are distance from (and therefore cant be improperly influenced by) *present* points.

Changes in accuracy are unremarkable.

Table 6.11: Presence/Absence measures, differences between large circles and ellipses

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG	0.00	-0.09	-0.07	0.11	0.06
DECAF-IK	-0.07	-0.04	-0.06	0.00	0.02
IDW	0.00	-0.18	-0.17	0.19	0.11
Kriging	0.00	-0.34	-0.40	-0.03	-0.13

Table 6.11 shows the same difference in measures, but between small and large ellipse, $small - large$ (we are isolating the effect of size in ellipses). It appears that all methods have higher precision with large ellipses than small ones. In specificity, we see again that IDW and DECAF-EG perform comparatively well on larger ellipse. Kriging sees a drop in accuracy at larger scales; IDW sees improvement.

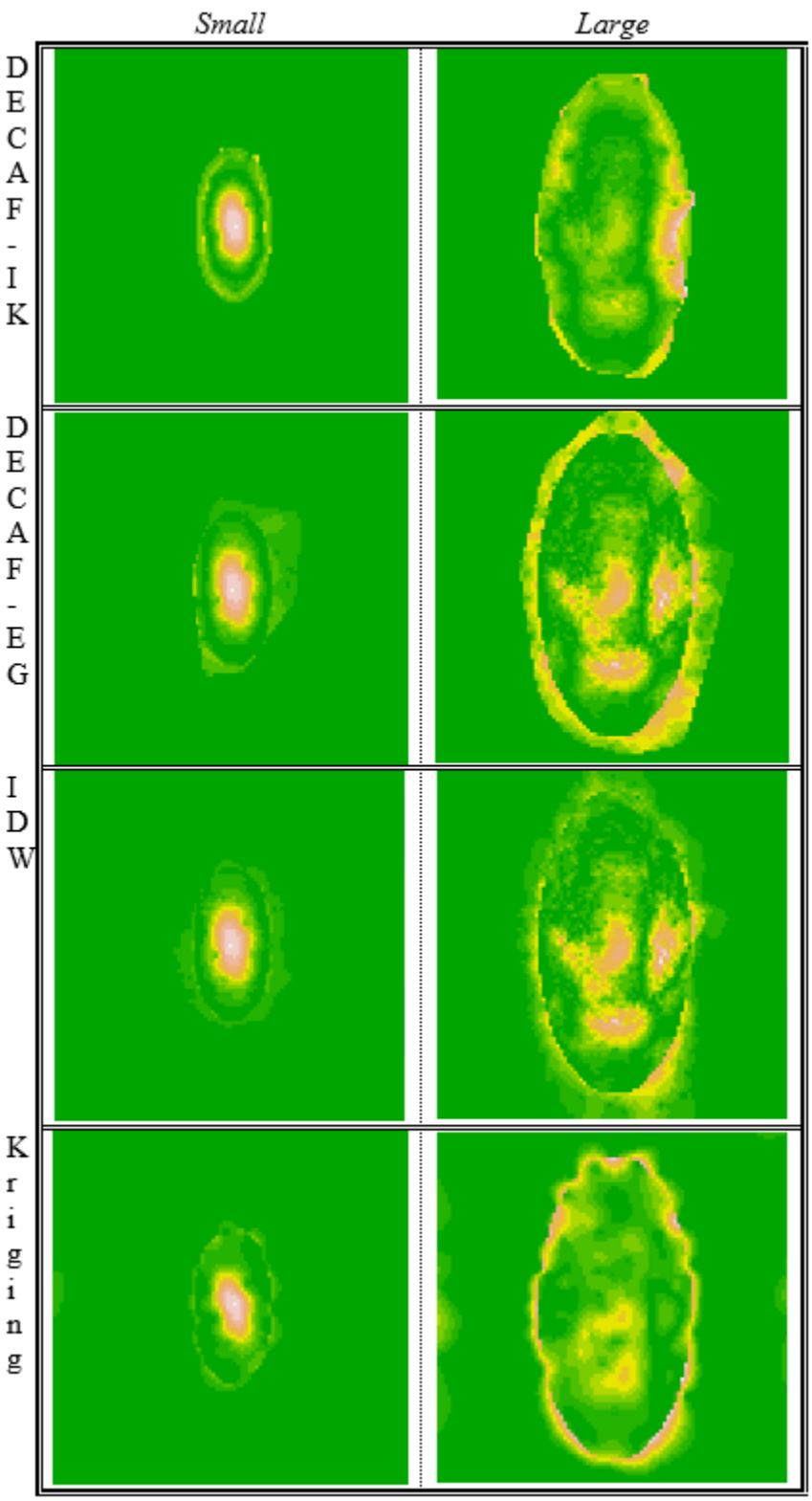
6.2.2.2 AAD

Table 6.12 on page 103 shows the AAD visualizations for small and large ellipses. When compared to our earlier circle visualizations (Table 6.1 on page 87), these suggest that not much is happening as a result of the change in shape. The same ring pattern occurs, where underestimation error dominates the center and overestimation error dominates the exterior.

For the magnitude measures, we will only show the interaction plots for small events. The stories are very similar across size. The interaction plots for large shapes can be found in Appendices. The AAD interaction plot for small ellipses is shown in Figure 6.6.

This plot looks interesting, but it is difficult to discern exactly what is happening. Let us return to this plot after considering the $AD+$ and $AD-$ plots.

Table 6.12: Ellipse visualizations (AAD)



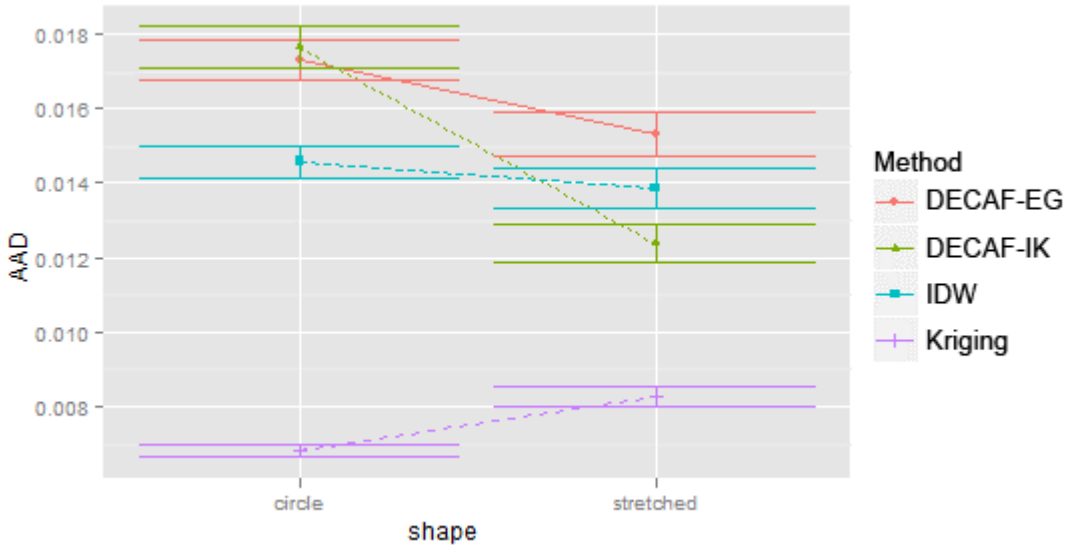


Figure 6.6: AAD method by shape interaction plot, size small

6.2.2.3 AD+

The $AD+$ interaction plot is given in Figure 6.7. All methods suffer greater overestimation error when estimating elliptical events.

6.2.2.4 AD-

The $AD-$ interaction plot is given in Figure 6.8.

All methods also suffer greater underestimation error when estimating elliptical events. Intuitively, the AAD plot should also show the four methods suffering greater AAD for elliptical events—yet, with the exception of kriging, it did not. In fact, both DECAFs showed *improvement*.

Remember, DECAF-IK and DECAF-EG do better than their counterparts at correctly identifying *absent* points, with DECAF-IK doing the best. Note that DECAF-EG improves on the AAD plot when working with ellipses, but DECAF-IK does even better. But, then, why would only AAD (but neither $AD+$ nor $AD-$) reflect

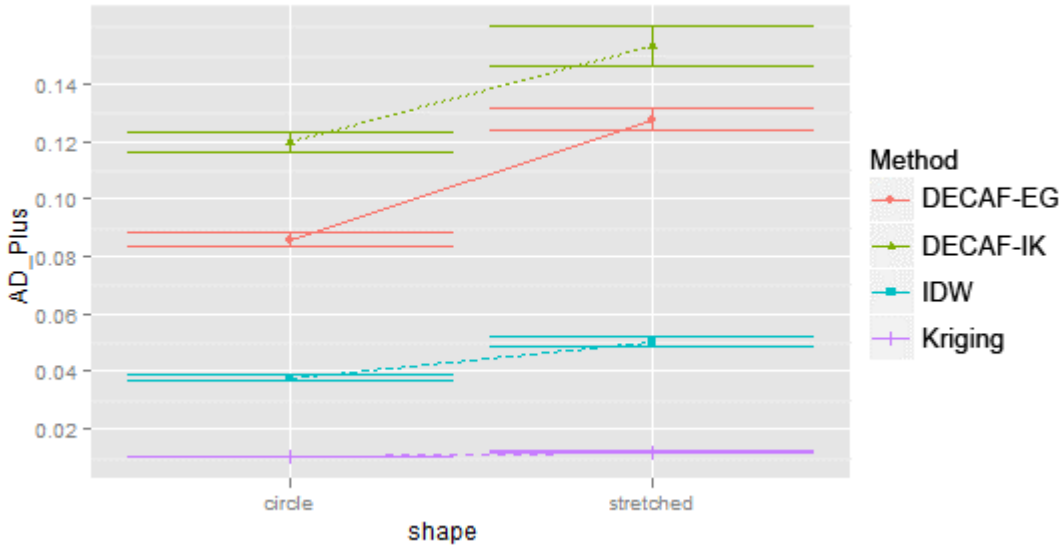


Figure 6.7: AD+ method by shape interaction plot, size small

improvements resulting from improved *present*/*absent* prediction accuracy?

This is an artifact of how $AD+$ and $AD-$ are computed. $AD+$ is the measure of average error at those points that are overestimated; $AD-$ is the measure of average error at those points that are underestimated. Points estimated with perfect accuracy do not factor into either $AD-$ or $AD+$. AAD, on the other hand, is the measure of error at all points, so points estimated with perfect accuracy do get factored into AAD. Because error stored as a floating point number, just about the only time "perfect" (no) error occurs is when an *absent* point is correctly predicted! DECAF-IK is best positioned to benefit from this peculiarity; DECAF-EG is not too far behind.

6.2.2.5 MAD

The MAD interaction plot simply recapitulates the fact that ellipse estimates suffer higher error than circles. See Appendices for the plots.

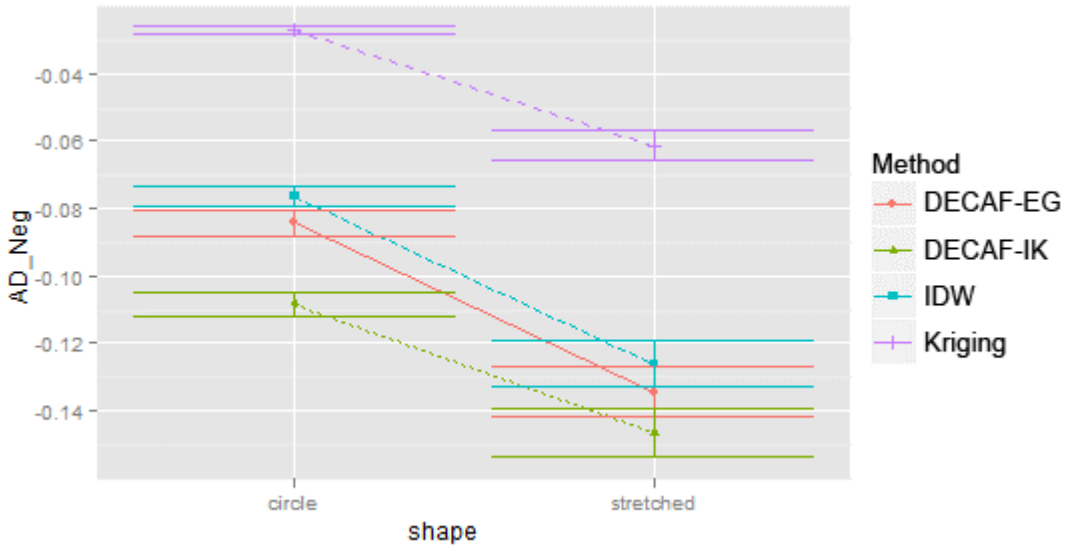


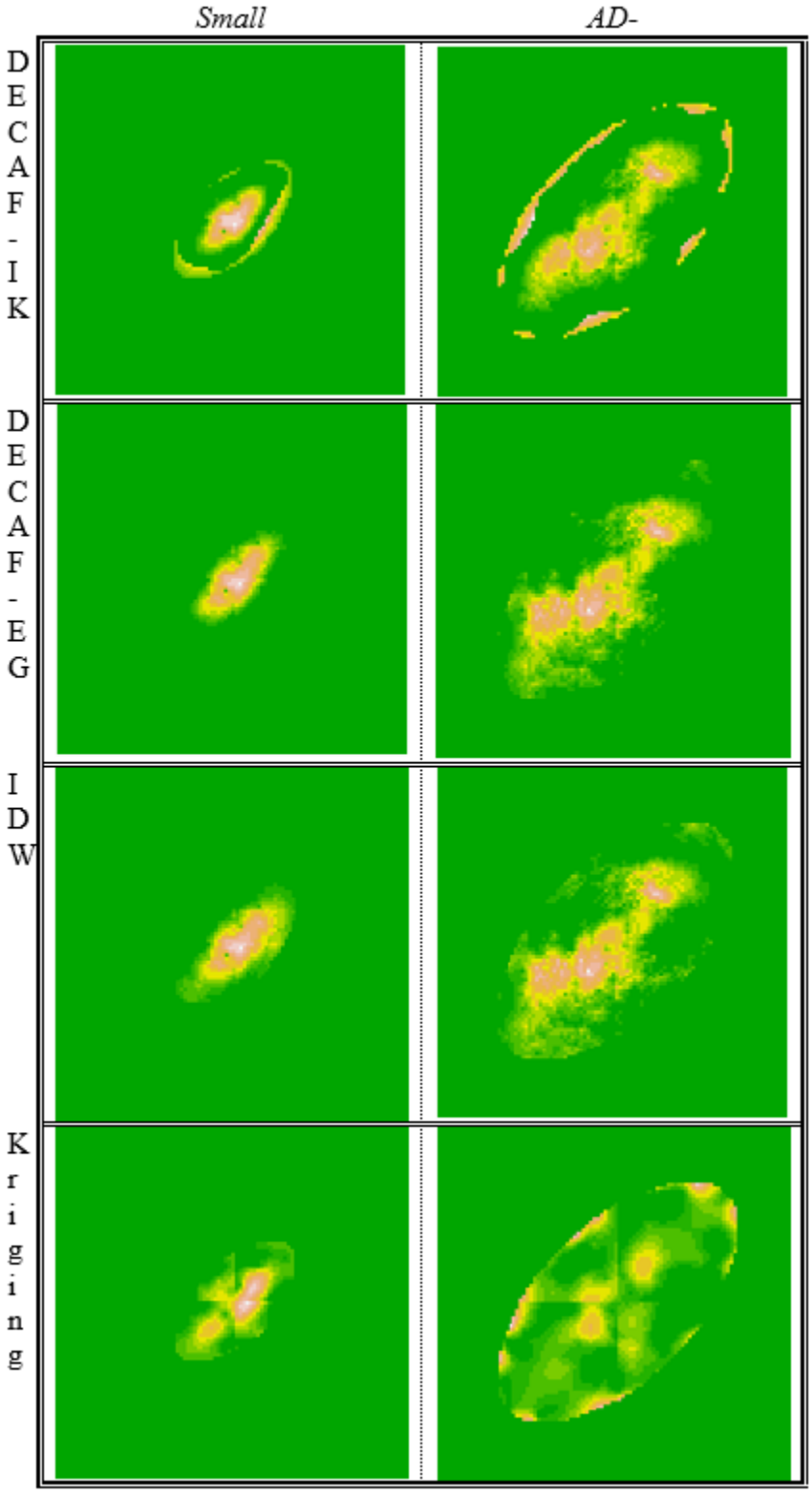
Figure 6.8: AD- method by shape interaction plot, size small

6.2.3 Orientation

According to the ANOVA test performed on the AD- measure, orientation (θ) has a statistically significant effect at the $\alpha < 0.05$ threshold (this table can be found in the Appendices). This is much less significant than the thresholds crossed for most of the other significant results, such as shape and size (which typically exceed the $\alpha < 0.001$ threshold). It warrants a brief investigation, however.

Table 6.13 on page 107 shows the visualizations of $AD-$ for the orientated events, large and small. A visual inspection does not reveal anything out of the ordinary, with the exception of an odd seam effect along the cardinal directions in the kriging plot.

Table 6.13: Orientation visualizations



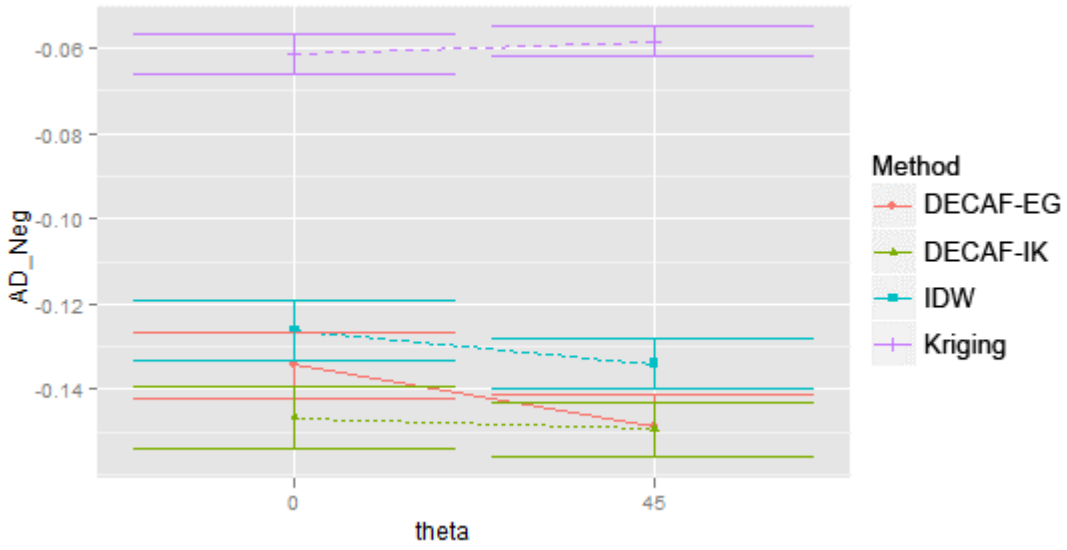


Figure 6.9: AD- method by orientation interaction plot, size small

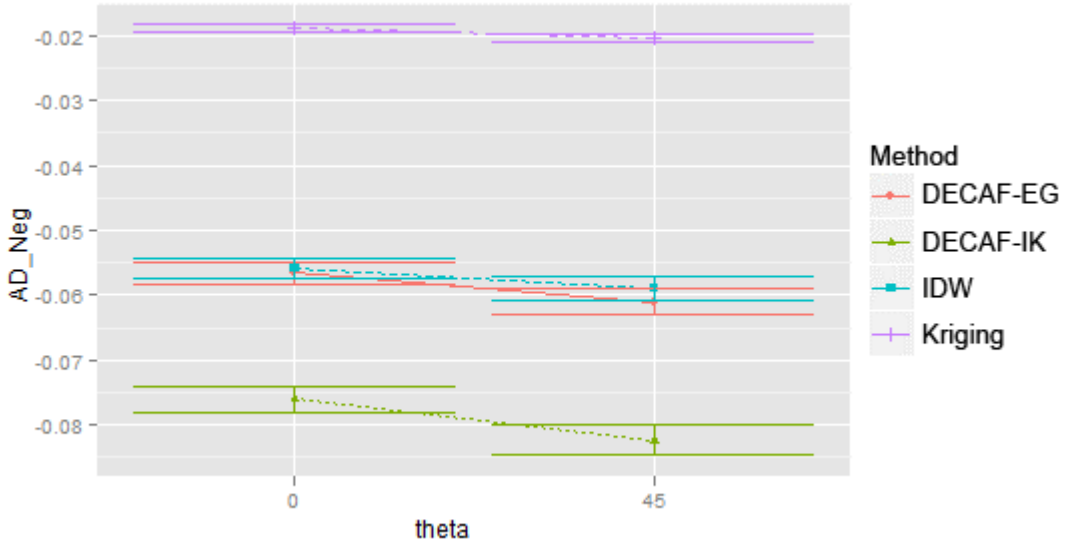


Figure 6.10: AD- method by orientation interaction plot, size large

The interaction plots are a little more revealing. Figure 6.9 shows the interaction across orientations (0° and 45°) at the small scale; Figure 6.10 shows the interaction

at the large scale.

Kriging and IDW are effectively indifferent to event orientation. DECAF-IK shows no effect at the small scale, but, interestingly, suffers increased error at the large scale when orientated at $\theta = \frac{\pi}{4}$. DECAF-EG shows the only noteworthy increase in underestimation error when oriented.

6.2.4 Noise

To analyze the effect of noise, we consider simulation types 4 and 7-14. All of these are large circles; only the noise levels vary. The summary of these simulation types is given in Table 6.14.

Table 6.14: Experiment design, noise

	Size	*Shape*	*Orientation*	*Sill*	*Nugget*
4	large	circle	0°	none	none
7	large	circle	0°	none	low
8	large	circle	0°	none	high
9	large	circle	0°	low	none
10	large	circle	0°	low	low
11	large	circle	0°	low	high
12	large	circle	0°	high	none
13	large	circle	0°	high	low
14	large	circle	0°	high	high

Table 6.15 reports the results of the analysis of variance test (ANOVA). The response variable for this test is absolute average difference (AAD). All terms are significant; this holds true for the $AD+$, $AD-$, and MAD response variables as well (these can be found in the Appendices).

Table 6.15: ANOVA results, AAD for noise

	DF	Sums of Squares	Mean of Squares	F Value	P value
sill	2	2.179	1.090	5189.1	$< 2.2e^{-16}$ ***
nugget	2	1.339	0.669	3188.0	$< 2.2e^{-16}$ ***
method	3	0.016	0.005	25.7	$< 2.2e^{-16}$ ***
sill * nugget	4	0.075	0.019	88.8	$< 2.2e^{-16}$ ***
sill * method	6	0.034	0.006	26.7	$< 2.2e^{-16}$ ***
nugget * method	6	0.015	0.003	12.2	$1.2e^{-13}$ ***
sill * nugget * method	12	0.012	0.001	4.8	$5.7e^{-8}$ ***
Residuals	3564	0.748	0.000		

Significance: ***$\alpha < 0.001$ **$\alpha < 0.01$ *$\alpha < 0.05$

6.2.4.1 AAD

Figure 6.11 is an interaction plot across nugget levels. Figure 6.12 is an interaction plot across sill levels.

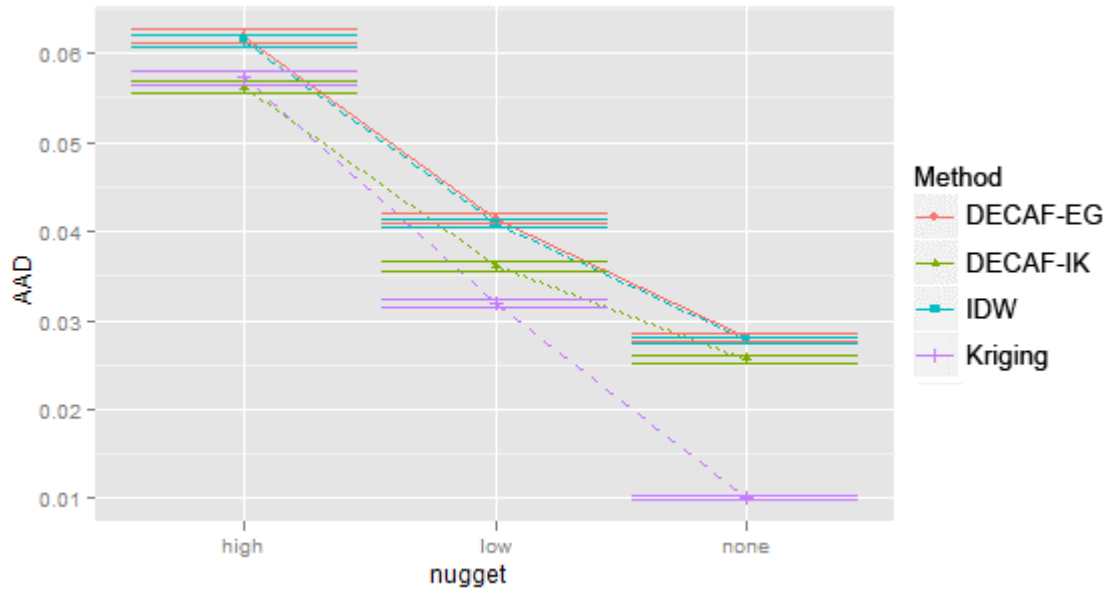


Figure 6.11: AAD method by nugget interaction plot

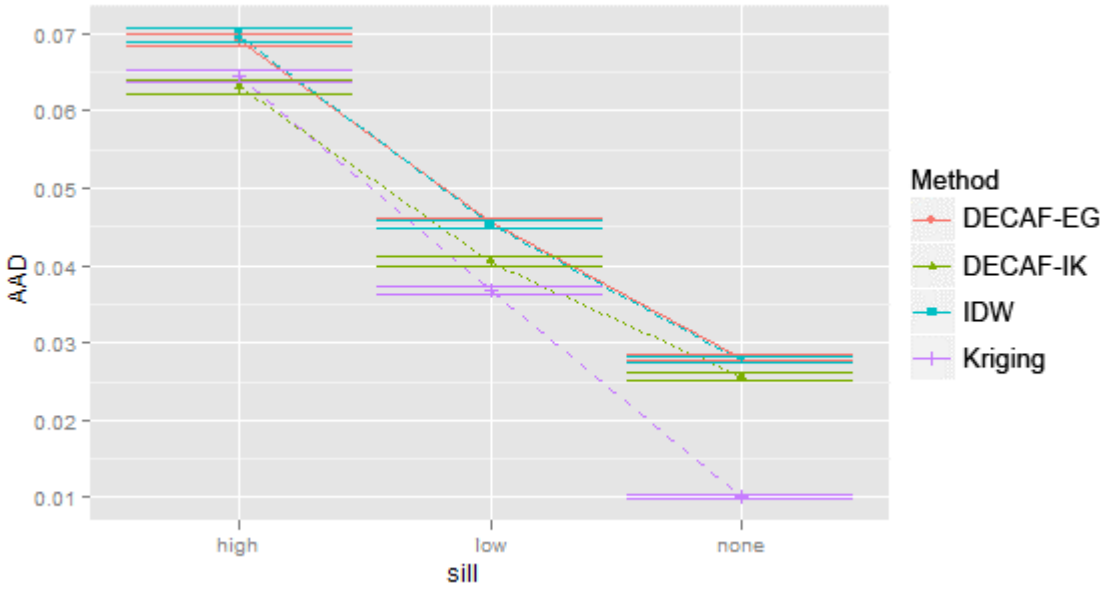


Figure 6.12: AAD method by sill interaction plot

Clearly, as expected, increased noise causes increased error. More interestingly, kriging and DECAF-IK exchange places: kriging outperforms all other methods by a wide margin when no noise is present, but its advantage over DECAF-IK is erased when noise is high. In the case of pseudo-nugget noise, it is statistically indistinguishable from DECAF-IK; DECAF-IK statistically outperforms kriging under high pseudo-sill noise conditions.

6.2.4.2 AD+

Figure 6.13 shows the *AD+* pseudo-nugget interaction plot (the pseudo-sill plot is almost identical and can be found in Appendices).

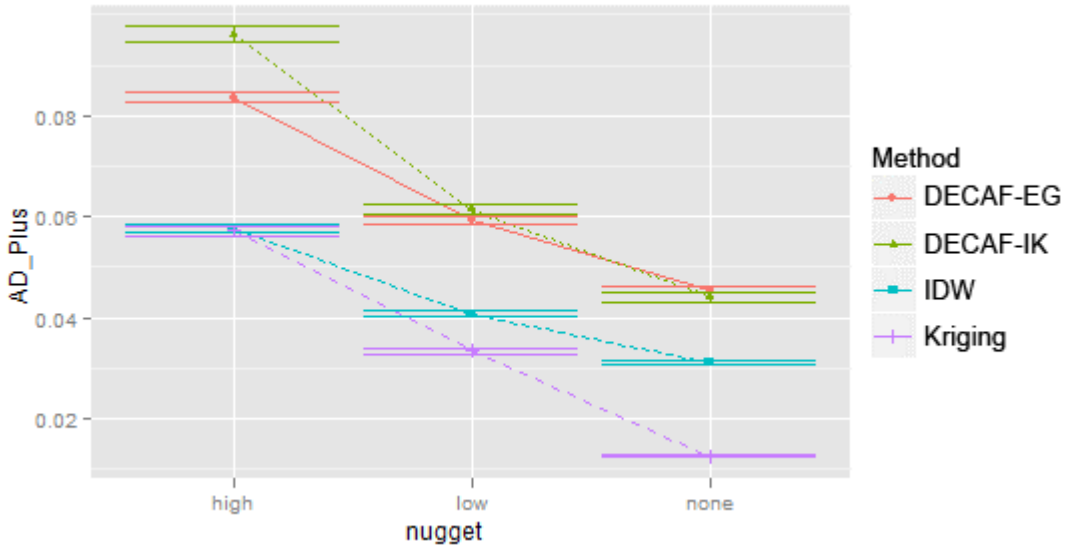


Figure 6.13: AD+ method by nugget interaction plot

6.2.4.3 AD-

Figure 6.14 shows the $AD-$ pseudo-nugget interaction plot (the pseudo-sill plot is almost identical and can be found in Appendices).

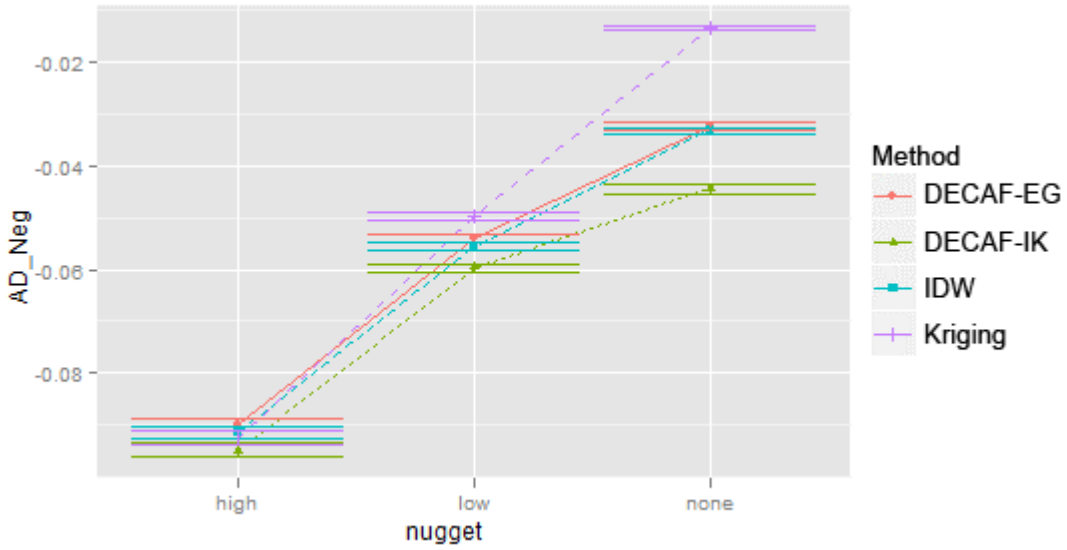


Figure 6.14: AD- method by nugget interaction plot

Here, all methods start in their by-now familiar order of underestimation error. Interestingly, all converge at high levels of noise.

Note DECAF-IK matches the performance of kriging in the AAD metric but is handily outperformed in $AD+$ and only matches kriging in $AD-$. We saw this pattern before in the event *shape* analysis: DECAF-IK's correct *absent* predictions are captured by AAD but in neither $AD+$ nor $AD-$.

6.2.5 Noise × event structure interactions

To analyze the effect of *noise* × *event structure* interactions, we consider simulation types 1-6, 8, 12, and 14-29. Only noiseless and high-noise tests are considered. The summary of these simulation types is given in Table 6.16.

Table 6.17 on page 115 reports the results of the ANOVA test on AAD. We are interested in interactions of the type *method* × *noise* × *structure parameter*; there

Table 6.16: Experiment design, noise × structure interaction

	Size	*Shape*	*Orientation*	*Sill*	*Nugget*
1	small	circle	0°	none	none
2	small	stretched	0°	none	none
3	small	stretched	45°	none	none
4	large	circle	0°	none	none
5	large	stretched	0°	none	none
6	large	stretched	45°	none	none
8	large	circle	0°	none	high
12	large	circle	0°	high	none
14	large	circle	0°	high	high
15	small	circle	0°	none	high
16	small	circle	0°	high	none
17	small	circle	0°	high	high
18	small	stretched	0°	none	high
19	small	stretched	0°	high	none
20	small	stretched	0°	high	high
21	small	stretched	45°	none	high
22	small	stretched	45°	high	none
23	small	stretched	45°	high	high
24	large	stretched	0°	none	high
25	large	stretched	0°	high	none
26	large	stretched	0°	high	high
27	large	stretched	45°	none	high
28	large	stretched	45°	high	none
29	large	stretched	45°	high	high

are three of two: $nugget \times sill \times method \times size$ and $sill \times method \times shape$.

The ANOVA tests for $AD+$, $AD-$, and MAD show similar (or fewer) interactions. The associated tables are provided in Appendices.

6.2.5.1 Size × noise interaction

Figure 6.15 shows the interaction between *noise* and *method* for small circles. Test $s01$ is noiseless, test $s15$ is no pseudo-sill/high pseudo-nugget, test $s16$ is high pseudo-

Table 6.17: ANOVA results for noise × structure interaction, AAD

	DF	SS	MS	F Value	P value
sill	1	0.361	0.3615	7839.3	$< 2.2e^{-16}$ ***
nugget	1	0.199	0.1990	4315.9	$< 2.2e^{-16}$ ***
method	3	0.067	0.0223	484.5	$< 2.2e^{-16}$ ***
theta	1	0.130	0.1302	2823.5	$< 2.2e^{-16}$ ***
shape	1	0.313	0.3127	6782.0	$< 2.2e^{-16}$ ***
size	1	1.617	1.6169	35063.7	$< 2.2e^{-16}$ ***
sill * nugget	1	0.023	0.0226	491.0	$< 2.2e^{-16}$ ***
sill * method	3	0.012	0.0039	85.1	$< 2.2e^{-16}$ ***
nugget * method	3	0.012	0.0038	83.5	$< 2.2e^{-16}$ ***
sill * theta	1	0.019	0.0194	419.6	$< 2.2e^{-16}$ ***
nugget * theta	1	0.019	0.0185	401.3	$< 2.2e^{-16}$ ***
method * theta	3	0.000	0.0000	0.7	0.566
sill *shape	1	0.049	0.0491	1063.7	$< 2.2e^{-16}$ ***
nugget * shape	1	0.037	0.0373	809.1	$< 2.2e^{-16}$ ***
method * shape	3	0.000	0.0000	0.6	0.634
sill * size	1	0.182	0.1824	3956.5	$< 2.2e^{-16}$ ***
nugget * size	1	0.112	0.1119	2425.8	$< 2.2e^{-16}$ ***
method * size	3	0.003	0.0009	20.2	$5.0e^{-13}$ ***
theta * size	1	0.035	0.0347	753.1	$< 2.2e^{-16}$ ***
shape * size	1	0.111	0.1105	2396.4	$< 2.2e^{-16}$ ***
sill * nugget * method	3	0.004	0.0012	27.1	$< 2.2e^{-16}$ ***
sill * nugget * theta	1	0.002	0.0020	42.9	$6.1e^{-11}$ ***
sill * method * theta	3	0.000	0.0000	0.4	0.757
nugget * method * theta	3	0.000	0.0001	1.6	0.187
sill * nugget * shape	1	0.005	0.0052	113.4	$< 2.2e^{-16}$ ***
sill * method * shape	3	0.001	0.0002	3.7	0.011
nugget * method * shape	3	0.000	0.0001	1.6	0.189
sill * nugget * size	1	0.011	0.0110	239.0	$< 2.2e^{-16}$ ***
sill *method * size	3	0.002	0.0007	15.2	$7.6e^{-10}$ ***
nugget * method * size	3	0.003	0.0010	22.5	$1.7e^{-14}$ ***
sill * theta * size	1	0.005	0.0050	109.0	$< 2.2e^{-16}$ ***
nugget * theta * size	1	0.006	0.0063	136.7	$< 2.2e^{-16}$ ***
method * theta * size	3	0.002	0.0005	11.0	$3.5e^{-7}$ ***
sill * shape * size	1	0.026	0.0259	562.4	$< 2.2e^{-16}$ ***
nugget * shape * size	1	0.014	0.0142	307.7	$< 2.2e^{-16}$ ***
method * shape * size	3	0.004	0.0013	27.3	$< 2.2e^{-16}$ ***
sill * nugget * method * theta	3	0.000	0.0000	0.1	0.957
sill * nugget * method * shape	3	0.000	0.0001	2.4	0.070
sill * nugget * method * size	3	0.001	0.0004	9.6	$2.5e^{-6}$ ***
sill * nugget * theta * size	1	0.001	0.0011	23.7	$1.2e^{-6}$ ***
sill * method * theta * size	3	0.000	0.0000	0.2	0.895
nugget * method * theta * size	3	0.000	0.0000	0.3	0.854
sill * nugget * shape * size	1	0.002	0.0020	44.3	$3.0e^{-11}$ ***
sill * method * shape * size	3	0.000	0.0000	0.2	0.888
nugget * method * shape * size	3	0.000	0.0000	0.3	0.847
sill * nugget * method * theta * size	3	0.000	0.0000	0.3	0.858
sill * nugget * method * shape * size	3	0.000	0.0000	0.1	0.965
Residuals	9288	0.428	0.0000		

sill/no pseudo-nugget, and test $s17$ is high for both types of noise. Nothing interesting happens; the four methods all suffer increased error with increased noise and are typically ordered.

Figure 6.16 shows the same interaction but for large circles. Kriging and IDW begin in their standard positions with IDW suffering greater AAD, but at high noise levels the two trade places. The DECAFs begin at almost identical positions, but diverge as noise increases.

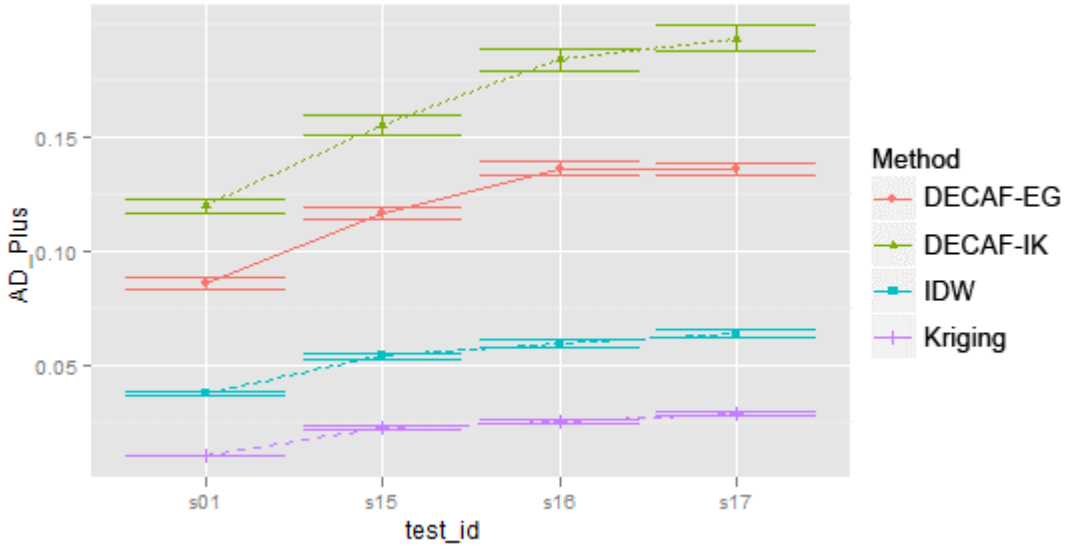


Figure 6.15: AAD noise by method interaction plot, small circle

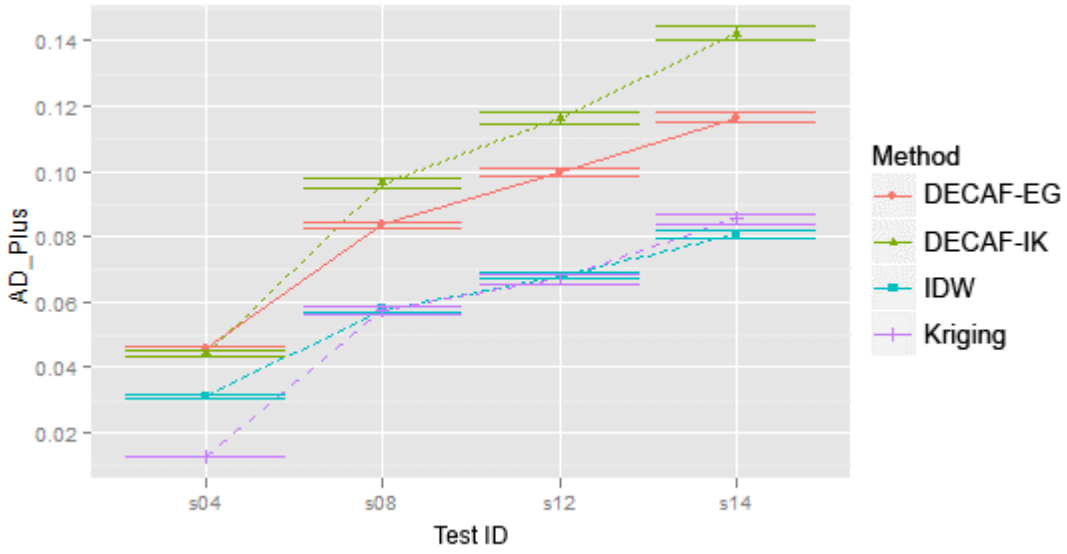


Figure 6.16: AAD noise by method interaction plot, large circle

6.2.5.2 Shape × pseudo-sill interaction

The interaction plot for pseudo-sill and method within large circles was shown in Figure 6.12 on page 111. Figure 6.17 is the interaction plot between pseudo-sill and method within large ellipses.

The effect of pseudo-sill within the large circle was interesting: DECAF-IK's *AAD* performance overtook krigings when pseudo sill was high. No such effect is visible in the ellipse interaction plot, however, as the methods are all comparatively well-behaved.

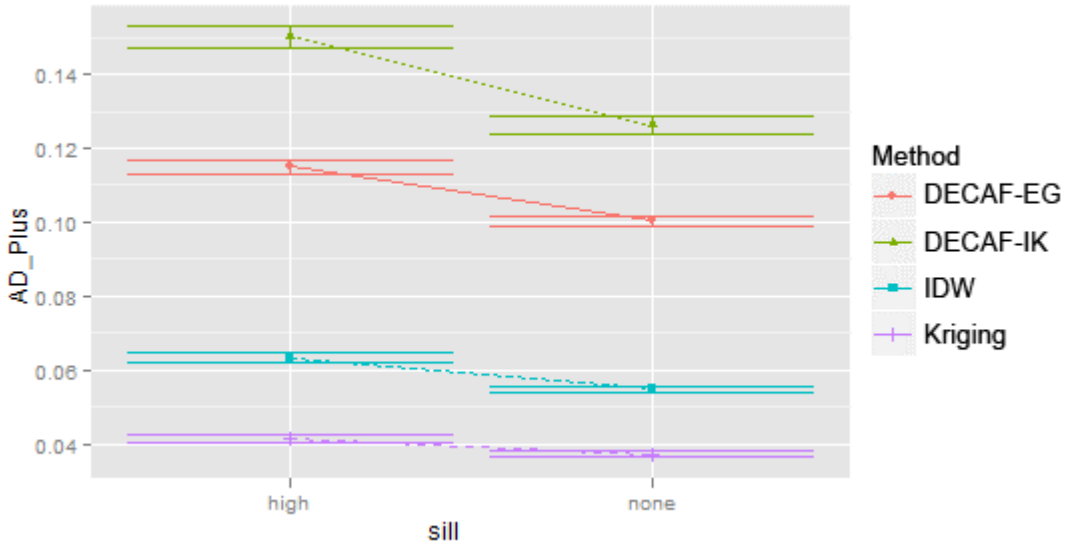


Figure 6.17: AAD noise by method interaction plot, large circle

6.2.6 Conclusions derived from simulation results

The above tests were performed to answer three questions regarding event structure.

1. **Does event structure impact the comparative performances of the four fusion methods in question (DECAF-IK, DECAF-EG, IDW, and kriging)?**

   Technically, yes. Differences were identified, most notably in size but also in shape. A minor orientation difference was identified for one measure ($AD-$).

   Practically, however, event structure as studied does not meaningfully impact the comparative performances of the four fusion methods. It may be instructive to note that kriging does better at larger event sizes, which is probably the result of more data with which to fit its models—but size does not affect kriging's performance relative to the other three methods.

2. **Does event noise impact the comparative performances of the four fusion methods?**

   Yes. High levels of noise reduced the efficacy of kriging to the point that it was indistinguishable from the other methods.

3. **Finally, do there exist interaction effects between event structure and event noise that impact the comparative performances of the four fusion methods?**

   Technically, yes. However, the results were not so dramatic as to be interesting. That DECAF-EG and DECAF-IK diverge at high levels of noise in large but not small objects is probably as likely a consequence of the experimental setup as a real effect—and, in any case, that divergence just maintains their standard relative positions.

The tests above failed to uncover event structure effects, which underscores the robustness of the general conclusions presented at this chapter's start. More importantly, it demonstrated that the DECAF methods are relatively insensitive to changes in event size, shape, or direction. They are relatively stable methods.

These simulations were not designed to uncover differences between DECAF and the traditional methods, though they were illustrative towards that end. We found that for simple Gaussian events with a minor threshold (10% in this case), kriging tends to outperform all other types in error magnitude tests. This is not unexpected.

In the next chapter, we will consider improvements to the DECAF approach based on unexpected results found in this chapter (namely, border behavior). We will also two additional experiments designed to better highlight DECAF's strengths.

Chapter 7

Improving DECAF-EG

The results from the simulation tests revealed a weakness in the DECAF approach. While DECAF had success detecting event boundaries, it failed to leverage this information to improve point-value prediction. In this chapter, we propose two new varieties of DECAF-EG to better exploit the spatial structure that DECAF captures.

The previous chapter revealed three weaknesses. First, DECAF-IK's conservatism caused underestimation at *present* points that fell immediately outside the estimated event boundary (Figure 7.1).

Figure 7.1: DECAF-IK underestimation along event border.

Second, DECAF-EG overestimated absent points that fell inside the estimated event boundary but outside the true event boundary (*overestimation type I* in Figure 7.2). Finally, both DECAF algorithms suffered overestimation just inside the event

boundary, the consequence of ignoring exterior *absent* points (*overestimation type II* in Figure 7.2).

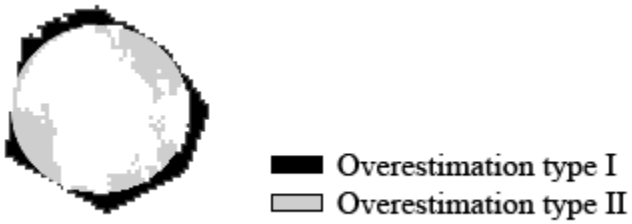


Figure 7.2: DECAF-EG Error along event border.

All of these problems result from the mishandling of the *border* region of the estimated event $\hat{e}$. Recall, in Chapter 4 we defined three subregions that the DECAF-EG process illuminates:

1. **Interior.** Inside a *present* hull, where the event is presumed continuous.

2. **Border.** Outside of a hull but not beyond its *absent* neighbors. This is a region of uncertainty—the event may extend into this region; it may not.

3. **Exterior.** Outside a hull and beyond *absent* neighbors, where it is presumed no event exists.

In a true delineated continuous event space, only two regions exist: *present* and *absent*—or, by this definition, *interior* and *exterior*. Either the event occurs at the location, or it does not.

However, information is imperfect in the estimated delineated continuous event space. Sample set P, however dense, cannot capture all locations. Space must exist between a *present* observation and an immediately adjacent *absent* neighbor. This region—the border region—is a region of inherent uncertainty. See Figure 7.3.

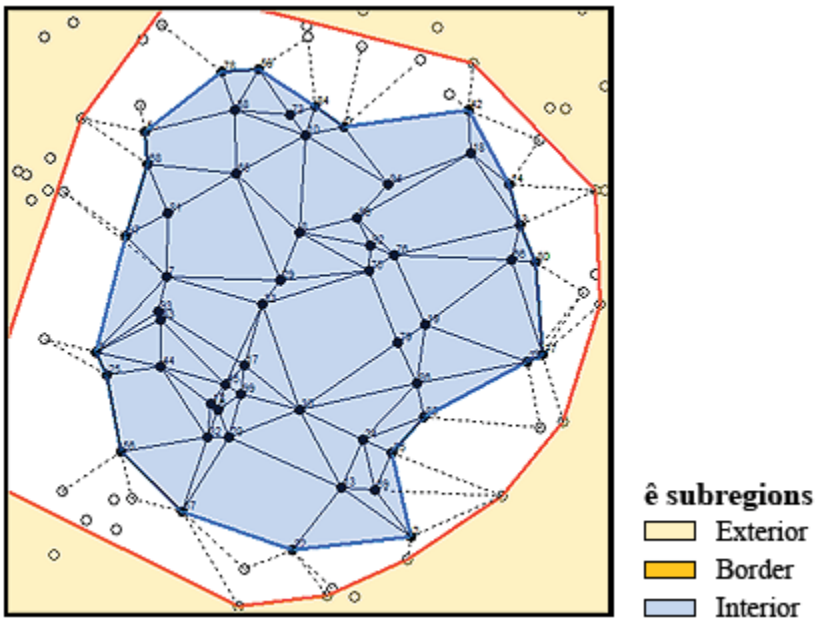


Figure 7.3: Example *interior*, *border*, *exterior* regions of an event estimation $\hat{e}$

DECAF-EG makes this division explicit, but it is no less important to DECAF-IK. DECAF-IK's indicator kriging-derived event boundary estimation can be thought of as tracing the midpoint between adjacent *present* and absent *points*. DECAF-IK effectively splits the *border* region between the *interior* and *exterior* regions. This is not an unreasonable thing to do, but it clearly results in error.

DECAF-EG, on the other hand, captures the entire *border* region by extending the event estimated boundary all the way to the nearest *absent* points. Effectively, it treats the *border* region as *interior*. The original expectation was that the interpolation process (IDW) would be effective at estimating these points—after all, towards the interior lie *present* points and towards the exterior lie *absent* points, and the weighted average between the two seemed to be an appropriate approximation for a point that may or may not be *present*.

Yet, overestimation happened. Why? The number of *absent* points is overwhelmed by the number of *present* points. Barring a very strange (a very non-

uniform) distribution of points in P, the number of points on the interior of estimation point q will greater than (and often much greater than) the number of nearby *absent* points that define the DECAF-EG exterior. Consider Figure 7.4, where q is estimated using four *present* points and two *absent* points; the two distant *absent* points are ignored even though they are probably just as informative as the pair of distant *present* points.

With this shortcoming in mind, let us consider how to improve the DECAF algorithms.

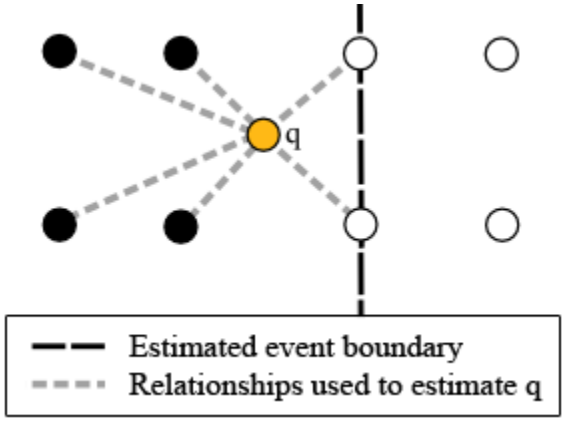


Figure 7.4: The DECAF-EG *border* overestimation problem

7.1 DECAF-IK

Recall that DECAF-IK uses indicator kriging to produce a grid of *present* probability. A cell with a value of 0.05 is considered to have 5% chance of being *present* (based on the sample set P provided). DECAF-IK thresholds at 50%, categorizing regions with > 0.5 *present* probability as *interior* and other regions as *exterior*. Clearly, we can alter this thresholding logic to produce a border region. For example, > 0.66

might be considered *interior*, < 0.33 might be considered *exterior*, and everything in-between *border*.

However, while the 0.5 logic was straightforward and easily defensible, selecting these new thresholds is probably more art than science. DECAF-EG, on the other hand, already creates these subdivisions as a consequence of its logic. Furthermore, DECAF-IK is *slow*. For these reasons, as well as time constraints, this "improved" DECAF-IK is not explored further by this thesis.

7.2 DECAF-EG

The first step towards improving DECAF-EG is simple. Previously, we only formally compute the hull at the *border*—this defines the maximal extent of the event. Now, we also compute a hull around all *present* members that fall inside the border hull; this new hull is the *interior* hull.

The pseudo-code for the DECAF algorithm is reprinted in Algorithm 6 for reference. The `find_events()` logic has been tweaked, as we just described. Now, we must alter the `predict()` function. Previously, this function was IDW parameterized with all points inside estimated event $\hat{e}$. From now on, we will refer to this original algorithm as DECAF-EG-1.

Algorithm 6 Delineated-Event Continuous-Aspect Fusion

```
1: function GENERATE(P, q, t, K)
2:     Ê ←find_events(P)
3:     ê ←find_associated_event(q, t, ê)
4:     V ← 0
5:     if ê ≠ NULL then:
6:         V ←predict(ê, q)
7:     end if
8:     return V
9: end function
```

Now, we must define a new `predict()` function. First, we must decide whether or not to treat estimations inside the *interior* differently. Previously, when estimating a point q that fell inside the *interior*, absent points from the *border* may or may not have been used. This could be problematic, as it may cause some underestimation. So, it could be argued that the estimations inside the *interior* should strictly use *interior* points.

The potential downside, however, is that we would be introducing a new border problem. Points at the edge of the *interior* may be overestimated because nearby *absent* points are ignored, causing gradient information loss.

Table 7.1 shows the results of a simulation using the two approaches (based on the $s01$ simulation—small circle, noiseless; absence region was cropped for figure). There appears to be a small improvement in underestimation error at the interior margins of the underestimated region, as we would expect. However, we also witness the expected increase in overestimation at the exterior of this same region.

We choose to be conservative and avoid the potential border effects. Therefore, interior points are estimated using *all* points from P that fall inside estimated event $\hat{e}$.[1]

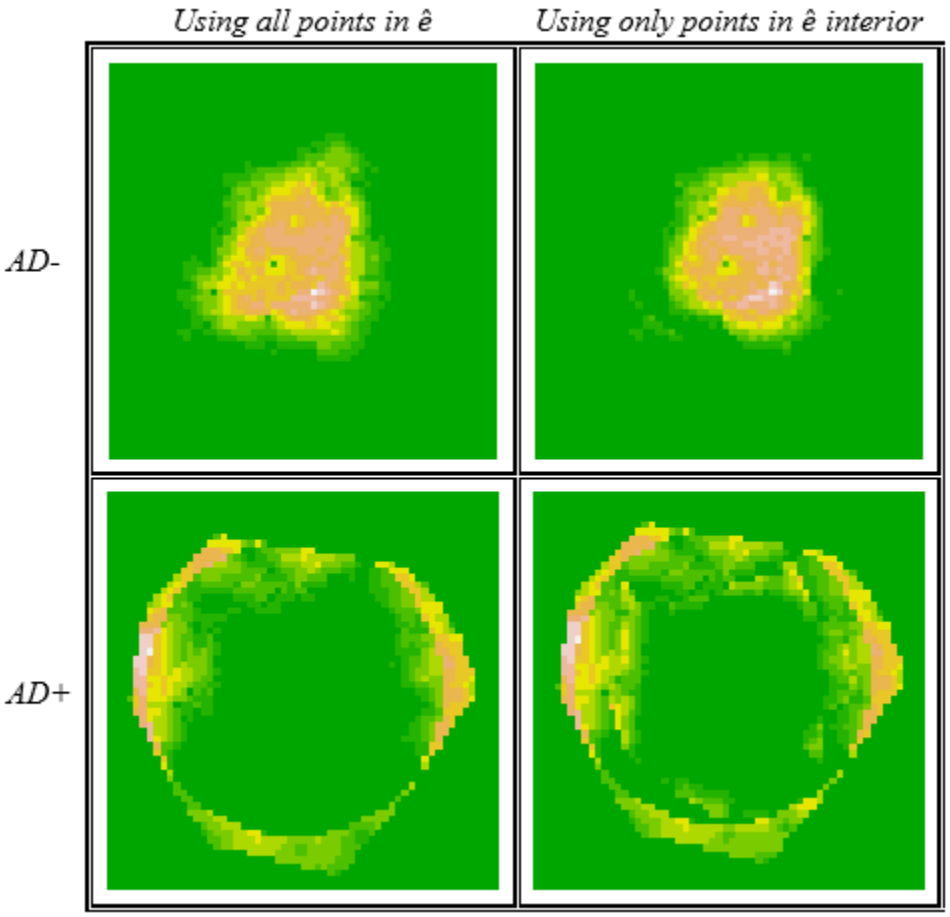
Second, we must decide whether or not to treat estimations in the *border* region differently, which we do.

7.2.1 DECAF-EG

The *predict* method of DECAF-EG-2 is outlined in Algorithm 7. Border logic has been introduced. If the estimation point q falls inside the interior, IDW is parameterized using all points inside the estimated event $\hat{e}$. If q falls outside of the event, it is

[1]To be clear, these points simply parameterize the interpolation function. For example, IDW, as implemented, would choose the nearest twelve points (maximum).

Table 7.1: Effects of using interior points to exclusively estimate values inside interior space



set to zero (or another default absent value). So far, this is identical to DECAF-EG-1.

However, DECAF-EG-2 checks whether q falls inside the *border* region. If so, IDW is used to estimate q. The IDW function is parameterized with all points. It chooses the k nearest points agnostic to their position inside or outside $\hat{e}$.[2]

In effect, where DECAF-EG-1 can be understood as the intelligent application of one IDW function (inside $\hat{e}$), DECAF-EG-2 can be understood as the intelligent application of two IDW functions (inside $\hat{e}$ *interior* and inside $\hat{e}$ *border*).

[2]In reality, the function is parameterized with the k-d tree so that distances are not recomputed. It uses this data structure to find the k nearest neighbors, preserving $O(n \log n)$ time complexity.

Algorithm 7 predict for DECAF-EG-2

```
1: function PREDICT(ê, q)
2:     v ← 0
3:     if q ∈ ê.interior_members then:
4:         v ← IDW(ê.interior_members)
5:     else:
6:         v ← IDW(ê.members)
7:     end if
8:     return v
9: end function
```

7.2.1.1 Time complexity

DECAF-EG-2 is nothing more than DECAF-EG-1 with a little more filtering logic for the IDW function. DECAF-EG-2 has the same time complexity as DECAF-EG-1, $O(n \log n)$.

7.2.2 DECAF-EG

DECAF-EG-3 is identical to DECAF-EG-2, but instead of using IDW to estimate q inside a *border* region it uses kriging. The motivation is straightforward: in the simulation tests, kriging outperformed IDW.

We have not used kriging in cases where the input is limited to points inside $\hat{e}$ (such as the DECAF-EG-1 logic or the DECAF-EG-2/3 *interior* logic). The fear is that in the real-world tests, the number of points from P that constitute an event estimation $\hat{e}$ may be very small—too small for the kriging implementation that we use to run. (Furthermore, using kriging causes time complexity to jump from $O(n \log n$ to $O(n^3)$).

However, the DECAF-EG-2 border logic uses all points in P. Set P is assumed to have enough points to fit a sensible semivariogram, so kriging becomes a viable replacement to IDW in our implementation. So, we create another variation of DECAF-

EG that uses kriging in *border* regions: DECAF-EG-3.

7.2.2.1 Time complexity

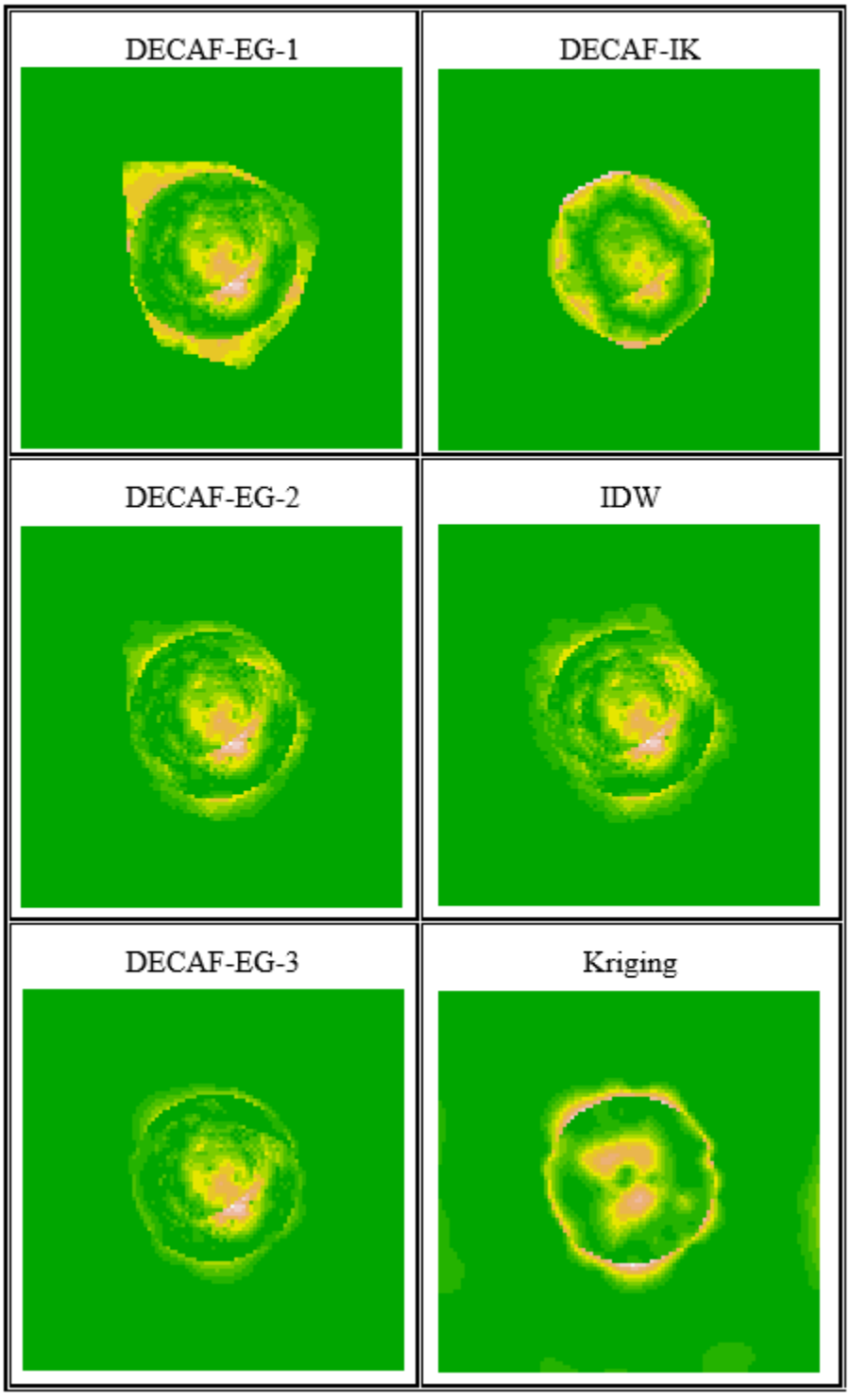
DECAF-EG-3 is at least as complex as DECAF-EG-1, $O(n \log n$. However, if so much as a single verification point falls inside a *border* region, kriging will be used. In the worst-case, all verification points fall inside *border* regions and kriging is used in all cases. Therefore, DECAF-EG-3 has a time complexity of $O(n^3)$.

7.2.3 Example: comparing DECAF-EG-2 and DECAF-EG-3

Figure 7.2 on page 129 shows the results of a sample test against DECAF-EG-1, DECAF-EG-2, DECAF-IK, IDW, and kriging. Note that both are significant improvements over DECAF-EG-1. It also appears that DECAF-EG-3 slightly outperforms DECAF-EG-2.

Because this is simply a visualization of a single simulation run, we will not claim that this conclusively demonstrates that the modifications to DECAF-EG are necessarily better. However, the evidence is sufficient enough for us to compare all three DECAF-EG methods in the following chapters.

Table 7.2: Comparison of methods on small circle, no noise, AAD



Chapter 8

Precipitation Experimental Design

We will now test the methods on real-world data. The simulated tests were very simple: only one event appears in the event space at a time, these events were shaped convexly, and the random fields inside events followed a Gaussian distribution. Though these tests revealed the efficacy of DECAF at *presence/absence* detection, traditional methods suffers less continuous error than the DECAF algorithms. We hypothesize that this will not hold true in a more complex environment.

This hypothesis is based on several observations. First, kriging presupposes a phenomenon that behaves homogenously—spatial relationships developed in one part of the space are presumed to hold true in another (otherwise, the semivariogram is no longer useful). Suppose one large event and one small event that exist in the same space; the spatial relationships existant within one event probably do not hold true in the other. Because DECAF treats events individually, it does not suffer the same weakness.

Second, complicated shapes can confuse kriging. For example, consider a C-shaped event; the C itself is composed of present events, everything else is absent. Kriging and IDW would be prone to predict the interior of the C as *present* (depending on

factors such as size, of course). Both DECAF-EG and DECAF-IK can handle concave shapes, and so are (comparatively) immune to non-simple shapes.

Finally, the simulated events had a low threshold "drop-off". If kriging fitted the curve perfectly, it was guaranteed the small amount of error shown in Figure 8.1 (a). We already have witnessed how sill noise, which increased this drop-off, caused more problems for kriging than other methods. If the drop-off is larger, as shown in Figure 8.1 (b), krigings low specificity may cause it significant problemsproblems that will likely be shared by IDW.

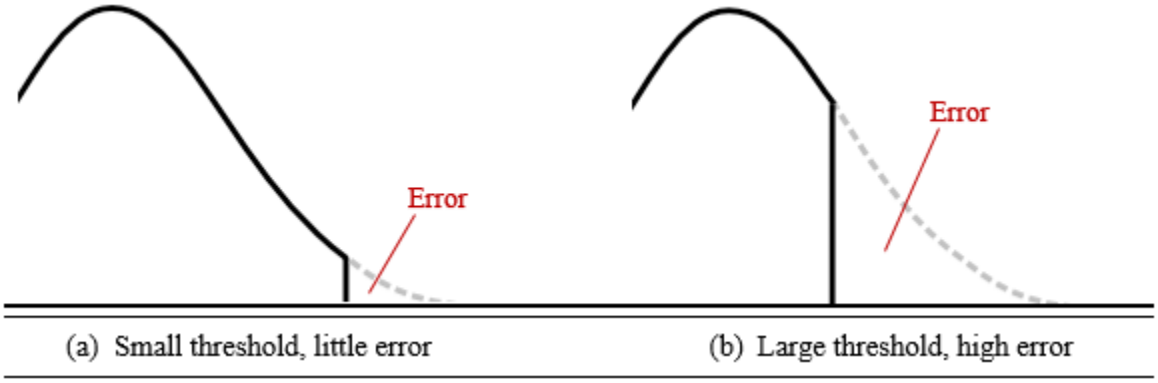


Figure 8.1: The threshold effect on a well-fitted curve

To test this hypothesis, we test our methods on a precipitation dataset at the day scale over a six year period.

8.1 Algorithms

The previous chapter presented two new algorithms, DECAF-EG-2 and DECAF-EG-3, which brings the total number of methods to six. As was the case with the simulations, each of the methods can be parameterized. So, we will review those parameterizations below.

DECAF. All DECAF methods threshold the data to determine *presence* and *absence*. For the precipitation data, it is 0.02 inches of precipitation in a 24-hour period (this number was chosen after experimentation as well as discussion with the High Plains Regional Climate Center).

The varieties of DECAF were further specified as follows:

DECAF-IK. The alpha hull algorithm was paramaterized with a starting alpha value of 0.05, a failure increment of 0.05, and a max alpha of 0.1 (if this threshold was reached, a convex hull is computed instead). The *presence/absence* raster was computed at a resolution of 150×70.

DECAF-EG. Point neighbors were identified using a static sector search (considered north, south, east, and west). Clusters with fewer than six members were dropped.

DECAF-EG-1. The interpolation algorithm used was IDW, capped at twelve neighbors.

DECAF-EG-2. The interpolation algorithm used was IDW, capped at twelve neighbors.

DECAF-EG-3. The *interior* interpolation algorithm was IDW capped at twelve neighbors. The *border* interpolation algorithm was kriging, same as specified below.

IDW. The maximum number of neighbors considered was twelve. This threshold, though arbitrary, is borrowed from the industry standard geographic information system, Esri's ArcGIS [18]. The IDW function in ArcGIS uses twelve as the default cutoff value.

Kriging. Ordinary kriging was used. The semivariogram was fitted with a spherical

model. The spherical model performed similar to other standard models—exponential, circular, Gaussian—in exploratory analysis. We did not consider more esoteric models. Furthermore, the literature contends that differences arising because of choice of model are typically dwarfed by other factors, such as data selection [6]. The semivariogram is fitted automatically for each test. We do not parameterize it *a priori*.

8.2 Dependent Variables

We are interested in the error associated with the six methods. For both tests, simulation and real-world, the predicted values outputted by the methods will be compared with known *true* values. So, error is known.

We measure error in two ways: *presence/absence* prediction and magnitude of prediction error. The first considers only the accuracy of event presence predictions (i.e., event e_i envelopes location $\langle x, y \rangle$). The second considers the accuracy of the continuous prediction (e.g., predicted 2" of rain at a location with 1" of true rain).

8.2.1 Presence/Absence

This is computed the same as before. Of note, however, the aggregated matrix is across time instead space. Previously, the number of *presence* and *absence* points were counted for one run of the simulation (so, the total PA numbers for the 100 verification points); this was performed individually for each simulation (with a result of 100 confusion matrices). The individual measures—PP, PA, AP, and AA—were averaged across runs.

Now, *presence* and *absence* predications are summed across time at a single point. The result is a confusion matrix at each point. The individual measures—PP, PA,

AP, and AA—were averaged across points.

From the confusion matrices, we compute the same measures as before: recall, precision, F-score, specificity, and accuracy. See Chapter 5 for a description of these measures.

8.2.2 Magnitude

For this test, we only compute three day-scale magnitude measures. These are given familiar names—Absolute Average Distance (AAD), Positive Average Difference ($AD+$), and Negative Average Distance ($AD-$)—but take note, there are differences. AAD remains exactly the same, but $AD+$ and $AD-$ have been tweaked to be more informative.

Absolute Average Difference (AAD). AAD is the absolute difference between the estimated and true value, averaged at the same point across days. In other words, *what is the average magnitude of error at this point?*

Positive Average Difference (AD+) $AD+$ the average difference between the estimated and true value across only those differences that are positive. In other words, *what is the average magnitude of over-estimation error?*

Previously, this was computed using the formula

$$AD+ = \frac{\sum overestimationerror}{n} \tag{8.1}$$

where n is the number of estimation errors that are measured. Now, we use the formula

$$AD+ = \frac{\sum overestimationerror}{n+m} \tag{8.2}$$

where n is the number of estimation errors that are measured and m is the number of perfect estimations (effectively, where *absence* was predicted correctly). Recall that one of the problems faced in the simulations chapter was the care that had to be taken to interpret $AD+$ and $AD-$ results, especially when they did not appear to match the AAD measure. The DECAF methods, which can and do get "perfect" errors (because of corrent *absent* predictions), ended up appearing to do worse than they do in reality—they were penalized for perfection.

Negative Average Difference (AD-) The average difference between the estimated and the true value across only those differences that a negative. In other words, *what is the magnitude of under-estimation error?*

$AD-$ is computed using the same new formula as $AD+$ (except that underestimation error is used instead of overestimation error, of course).

Finally, we also compute a fourth measure, accumulated error. This is the simple sum of error at points across days. From this measure we derive accumulated percent error, which is computed using the traditional percent error formula. As will be seen in the following chapter, in some circumstances this measure can be more illuminating than AAD, $AD+$, and $AD-$.

8.3 Real-world application

The real-world application comes from the domain of meteorology and climatology. Precipitation behaves in a delineated-continuous manner at fine temporal resolution-sstorms have a definite geographical extent, and within that extent precipitation falls continuously. We apply our two DECAF algorithms to the problem of precipitation

estimation. How well do DECAF-IK and DECAF-EG estimate the value of precipitation at unknown locations compared to IDW and kriging?

For this test, we use daily precipitation data for the state of Nebraska (United States). The temporal extend of the data set is the six year period 2007–2012 (inclusive). The time period is restricted to the months May–September of each year in order to avoid the non-trivial problem of normalizing snowfall and rainfall [12]. A volunteered dataset (CoCoRaHS) is used as the source dataset (P), and an institutional dataset (NWS Coop) is used for validation. The two data sets are shown in Figure 8.2.

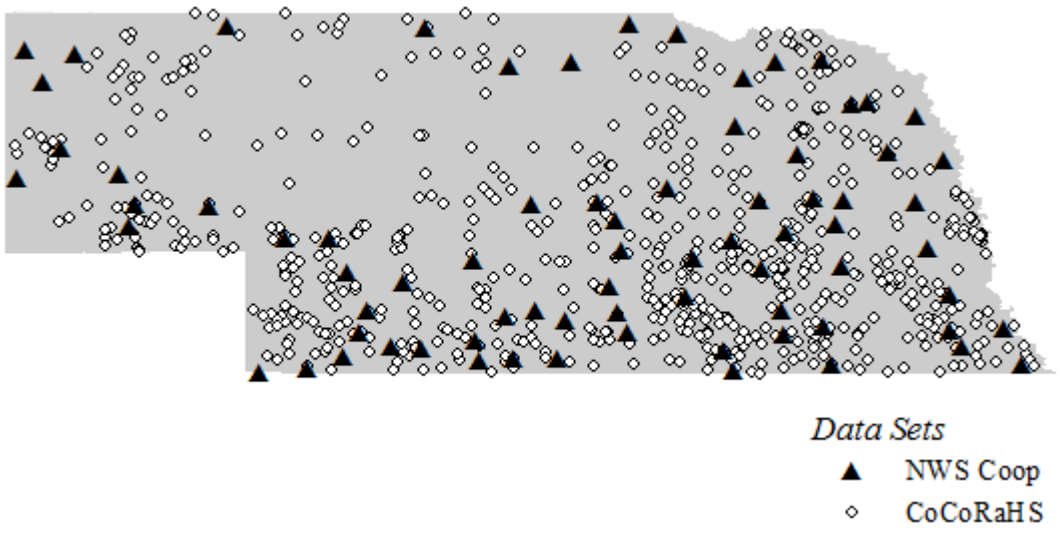


Figure 8.2: Source (CoCoRaHS) and Verification (NWS Coop) data sets

8.3.1 Source Set

The set P is drawn from the Community Collaborative Rain, Hail & Snow network (CoCoRaHS). CoCoRaHS is a volunteered geographic information (VGI) initiative where thousands of minimally-trained volunteers submit daily weather reports [13]. The daily participation rate varies widely among volunteers, with some contributing

regularly over long periods of time while others report for a couple weeks before stopping. The result is a large P (roughly 450 for any given t) with a composition that fluctuates through time.

8.3.2 Verification Set

We verify our results by comparing the estimated values with a second dataset, the National Weather Services Cooperative Observer Program (NWS Coop). These values are also collected by volunteers, but, in contrast to CoCoRaHS, the volunteers undergo extensive training and their reports are treated to a quality control regime [35]. The number of observations for any particular time t is fewer than CoCoRaHS. We further restrict the set to "centennial" stationsthat is, stations which have a history of reports dating back over a century. The final verification set is composed of 74 points.

8.3.3 Methodology

For each day (t) in the time span, the value of precipitation (V) is computed at each of the target 74 NWS Coop locations using the CoCoRaHS precipitation data as (P). We repeat this process for all six methods: DECAF-IK, DECAF-EG-1, DECAF-EG-2, DECAF-EG-3, IDW, and kriging. This totals to approximately 67,000 day-station observations for each technique (74 verification points $\times$ 6 years $\times$ 150 days).

Chapter 9

Experimental Results: Precipitation Data

This chapter reports the results of the six methods—DECAF-EG-1, DECAF-EG-2, DECAF-EG-3, DECAF-IK, IDW, and kriging—on the real world data set described in Chapter 8.

9.1 Presence/Absence Results

The presence/absence results are presented in Table 9.1.

Table 9.1: Presence/Absence measures

Method	Recall	Precision	F-Score	Specificity	Accuracy
DECAF-EG-1	0.764	0.778	0.771	0.893	0.851
DECAF-EG-2	0.763	0.784	0.774	0.897	0.853
DECAF-EG-3	0.749	0.797	0.772	0.906	0.854
DECAF-IK	0.646	0.864	0.739	0.950	0.850
IDW	0.922	0.585	0.716	0.678	0.759
Kriging	0.881	0.465	0.609	0.503	0.627

The table lends itself to several conclusions.

First, the three DECAF-EG algorithms are essentially identical. This should come as no surprise, as the three define the event boundary the same way. The (very minor) differences are the result of the occasional correct absence prediction inside of the event border.

Second, DECAF-IK no longer clearly eclipses all other methods at PA prediction. While it performed excellently in the comparatively simple simulations, it fails to repeat that performance on the real-world data. DECAF-IK continues to suffer the worst Recall—and now "worst" means 0.65 instead of 0.96—but it also has the best precision and specificity. So, its ability to detect *absence* remains unrivaled, but its capacity to correctly predict *presence* has suffered considerably. According to the two composite scores, F-score and Accuracy, it is indistinguishable from the DECAF-EGs.

Third, Kriging and IDW continue to have high recall (though not the 1.00 observed in the simulations). However, precision and specificity continue to suffer. Of note, kriging now underperforms IDW, which was not necessarily the case in the simulations.

What are the implications? Primarily, DECAF-EG is now an acceptable approximation of DECAF-IK (unless Precision is essential, in which case DECAF-IK may still be preferred). Furthermore, DECAF does what it is supposed to do: detect event extents better than either IDW or kriging.

Let us now turn to error magnitude—a low precision score may not matter if the magnitude of the resulting estimation errors is small.

9.2 Statistical differences among methods according to average error aggregated by location

On each of the 900 days, each point location in the verification set (NWS Coop) was estimated using the sample set (CoCoRaHS). The estimations were compared with the known value to quantify error at each location. For each station, error is averaged across time to produce a station-associated average absolute difference (AAD) value. Furthermore, we also compute the average positive difference ($AD+$) to quantify overestimation and the average negative difference ($AD-$) to measure underestimation.[1]

9.2.1 AAD

Table 9.2 on page 141 reports the results of a Tukey Honest Significant Difference test (Tukey's HSD test). The response variable is AAD. Differences between methods are reported. The test also blocked on point ID and the ANOVA test reported it effect as significant, but we have chosen not to report the associated differences because we expect spatial variation and we do not consider differences between individual points to be interesting.

From this table we observe two clusters of statistically indistinguishable methods. Group a is composed by all DECAFs; group b consists of DECAF-EG-1, kriging, and IDW. DECAF-EG-1 has membership in both groups, and DECAF-EG-3 is indistinguishable from kriging. We can approximately order the methods from best to worst as DECAF-EG-2/DECAF-IK, DECAF-EG-1/DECAF-EG-3, kriging/IDW.

[1]The $AD-$ and $AD+$ reported here are modified from the measures used in Chapter 6 to incorporate lessons learned from the simulation analysis. See Chapter 8 for details.

Table 9.2: Results of Tukey Honest Significant Difference test on AAD

	Difference	Lower	Upper	P value	Group
DECAF-EG-2 – DECAF-EG-1	-0.0010	-0.0023	0.0003	0.234	a
DECAF-EG-3 – DECAF-EG-1	-0.0007	-0.0021	0.0006	0.584	a
DECAF-IK – DECAF-EG-1	-0.0010	-0.0023	0.0003	0.214	a
IDW – DECAF-EG-1	0.0006	-0.0007	0.0019	0.770	b
Kriging – DECAF-EG-1	0.0003	-0.0010	0.0016	0.982	b
DECAF-EG-3 – DECAF-EG-2	0.0003	-0.0010	0.0016	0.992	a
DECAF-IK – DECAF-EG-2	0.0000	-0.0013	0.0013	>0 .999	a
IDW – DECAF-EG-2	0.0016	0.0003	0.0029	0.005 **	
Kriging – DECAF-EG-2	0.0013	0.0001	0.0026	0.044 *	
DECAF-IK – DECAF-EG-3	-0.0003	-0.0016	0.0010	0.988	a
IDW – DECAF-EG-3	0.0014	0.0000	0.0027	0.039 **	
Kriging – DECAF-EG-3	0.0011	-0.0002	0.0024	0.187	
IDW – DECAF-IK	0.0016	0.0003	0.0030	0.005 **	
Kriging – DECAF-IK	0.0014	0.0001	0.0027	0.038 **	
Kriging – IDW	-0.0003	-0.0016	0.0010	0.989	b

Significance: ***$\alpha < 0.001$ **$\alpha < 0.01$ *$\alpha < 0.05$

9.2.2 AD+

According to Table 9.3 on page 142, the DECAFs are statistically indistinguishable. Kriging and IDW are also indistinguishable. All DECAF methods outperform kriging and IDW. Also, note that DECAF-EG-2 consistently outperforms all other methods—the difference isnt statistically significant, but it is consistent.

9.2.3 AD-

According to Table 9.4 on page 143, The DECAF methods continue to be statistically indistinguishable. Again, all DECAF methods outperform kriging and IDW.

Table 9.3: Results of Tukey Honest Significant Difference test on AD+

	Difference	Lower	Upper	P value	Group
DECAF-EG-2 – DECAF-EG-1	-0.0013	-0.0030	0.0004	0.232	a
DECAF-EG-3 – DECAF-EG-1	-0.0007	-0.0024	0.0009	0.800	a
DECAF-IK – DECAF-EG-1	-0.0006	-0.0023	0.0011	0.915	a
IDW – DECAF-EG-1	0.0027	0.0010	0.0044	$8.1e^{-5}$ ***	
Kriging – DECAF-EG-1	0.0035	0.0019	0.0052	$4.4e^{-8}$ ***	
DECAF-EG-3 – DECAF-EG-2	0.0005	-0.0011	0.0022	0.937	a
DECAF-IK – DECAF-EG-2	0.0007	-0.0010	0.0024	0.832	a
IDW – DECAF-EG-2	0.0040	0.0023	0.0056	$5.0e^{-10}$ ***	
Kriging – DECAF-EG-2	0.0048	0.0032	0.0065	$< 2.2e^{-16}$ ***	
DECAF-IK – DECAF-EG-3	0.0002	-0.0015	0.0018	> 0.999	a
IDW – DECAF-EG-3	0.0034	0.0018	0.0051	$1.3e^{-7}$ ***	
Kriging – DECAF-EG-3	0.0043	0.0026	0.0060	$5.4e^{-12}$ ***	
IDW – DECAF-IK	0.0033	0.0016	0.0049	$5.6e^{-7}$ ***	
Kriging – DECAF-IK	0.0041	0.0025	0.0058	$8.7e^{-11}$ ***	
Kriging – IDW	0.0009	-0.0008	0.0025	0.683	

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

9.2.4 Conclusions

First, kriging is no longer the clear champion that it was in the simulation tests. The noise tests hinted that kriging might not perform as well on the comparatively messy precipitation data. The results of the Tukey HSD tests begin to bear this out.

Second, the DECAF methods generally outperform kriging and IDW but are indistinguishable among themselves. This suggests that PA logic is, in and of itself, sufficient to improve precipitation predictions, and therefore the precise variety of PA logic is less important.

Table 9.4: Results of Tukey Honest Significant Difference test on AD-

	Difference	Lower	Upper	P value	Group
DECAF-EG-2 − DECAF-EG-1	-0.0001	-0.0023	0.0021	> 0.999	a
DECAF-EG-3 − DECAF-EG-1	-0.0006	-0.0029	0.0016	0.969	a
DECAF-IK − DECAF-EG-1	-0.0023	-0.0045	-0.0001	0.040	a
IDW − DECAF-EG-1	0.0070	0.0047	0.0092	$< 2.2e^{-16}$ ***	
Kriging − DECAF-EG-1	0.0148	0.0126	0.0170	$< 2.2e^{-16}$ ***	
DECAF-EG-3 − DECAF-EG-2	-0.0005	-0.0028	0.0017	0.984	a
DECAF-IK − DECAF-EG-2	-0.0022	-0.0045	0.0000	0.054	a
IDW − DECAF-EG-2	0.0070	0.0048	0.0093	$< 2.2e^{-16}$ ***	
Kriging − DECAF-EG-2	0.0149	0.0126	0.0171	$< 2.2e^{-16}$ ***	
DECAF-IK − DECAF-EG-3	-0.0017	-0.0039	0.0006	0.261	a
IDW − DECAF-EG-3	0.0076	0.0053	0.0098	$< 2.2e^{-16}$ ***	
Kriging − DECAF-EG-3	0.0154	0.0132	0.0177	$< 2.2e^{-16}$ ***	
IDW − DECAF-IK	0.0093	0.0070	0.0115	$< 2.2e^{-16}$ ***	
Kriging − DECAF-IK	0.0171	0.0149	0.0193	$< 2.2e^{-16}$ ***	
Kriging − IDW	0.0078	0.0056	0.0101	$< 2.2e^{-16}$ ***	

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

9.3 Error maps, averages

Until now, we have analyzed estimation error more-or-less removed from spatial context (technically, we considered spatial context in the Tukey HSD statistical test because we blocked by point, but the subsequent analysis ignored space).

Figures 9.1 on page 145, 9.2 on page 146, and 9.3 on 147 show error maps (AAD, $AD+$, and $AD-$, respectively). These maps are interpolations (using IDW) of the values associated with the 77 verification points. The verification points are indicated on the maps by black circles. All maps within a figure use the same scale.

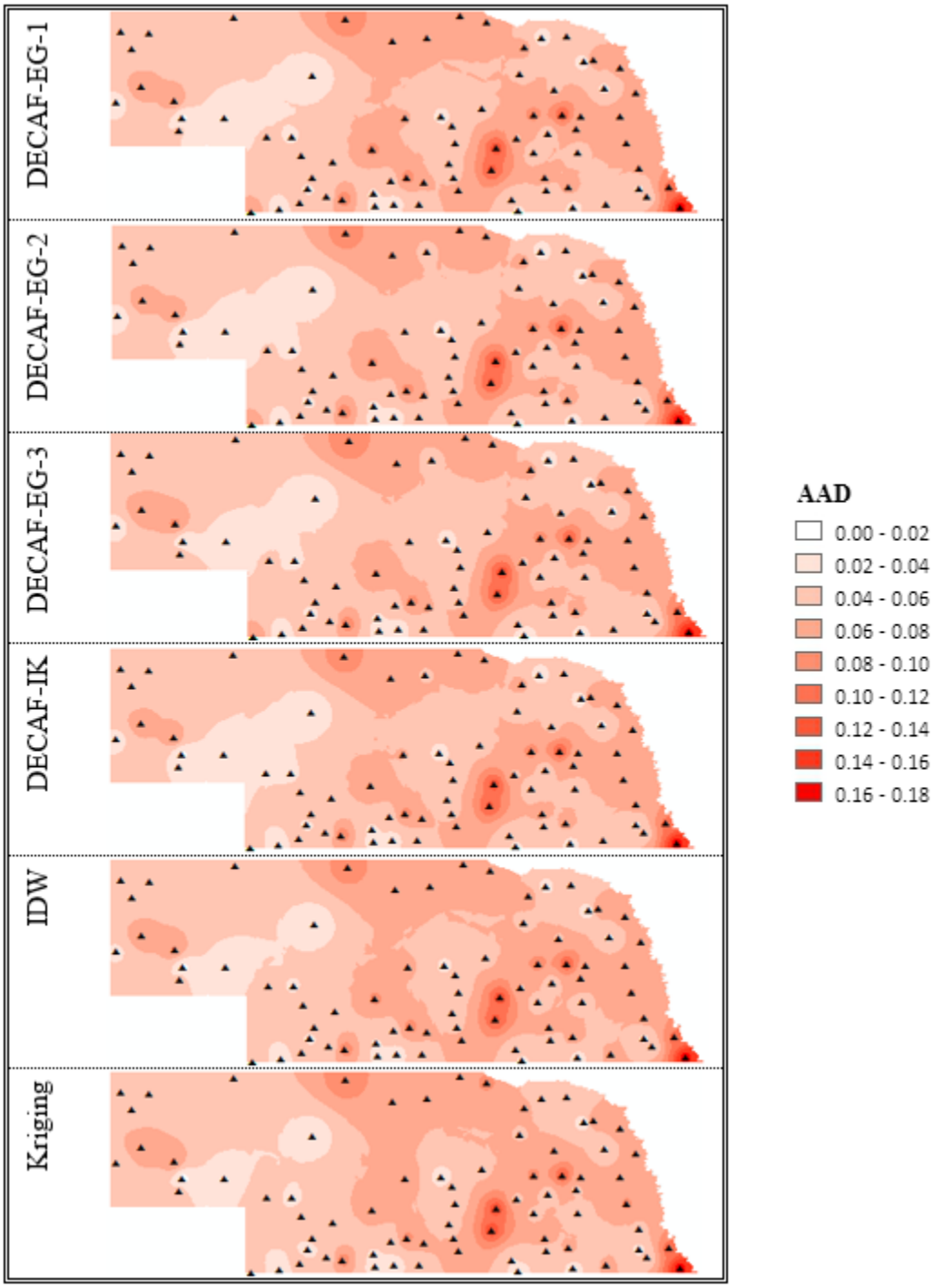


Figure 9.1: *AAD* error maps, interpolated

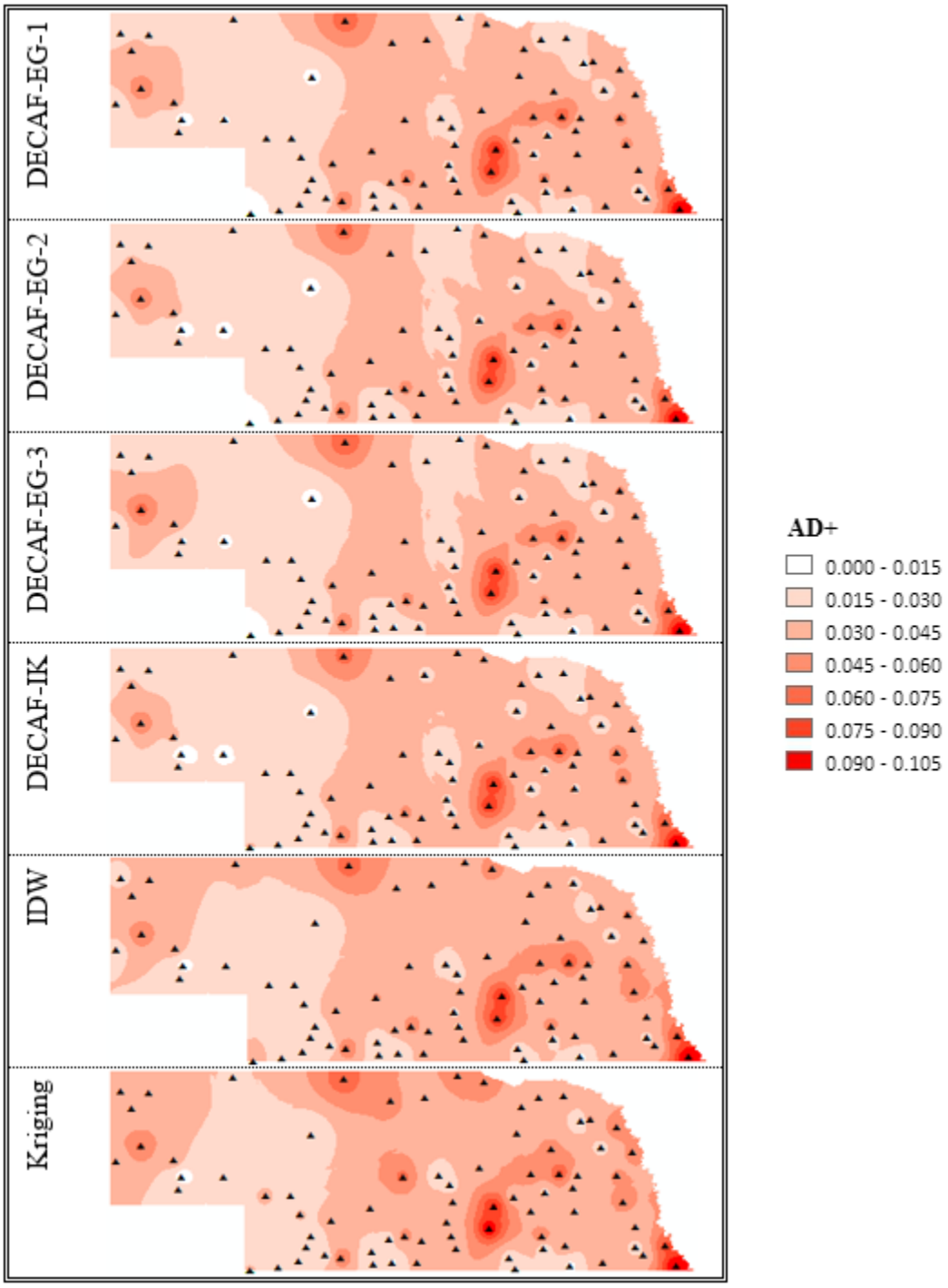


Figure 9.2: $AD+$ error maps, interpolated

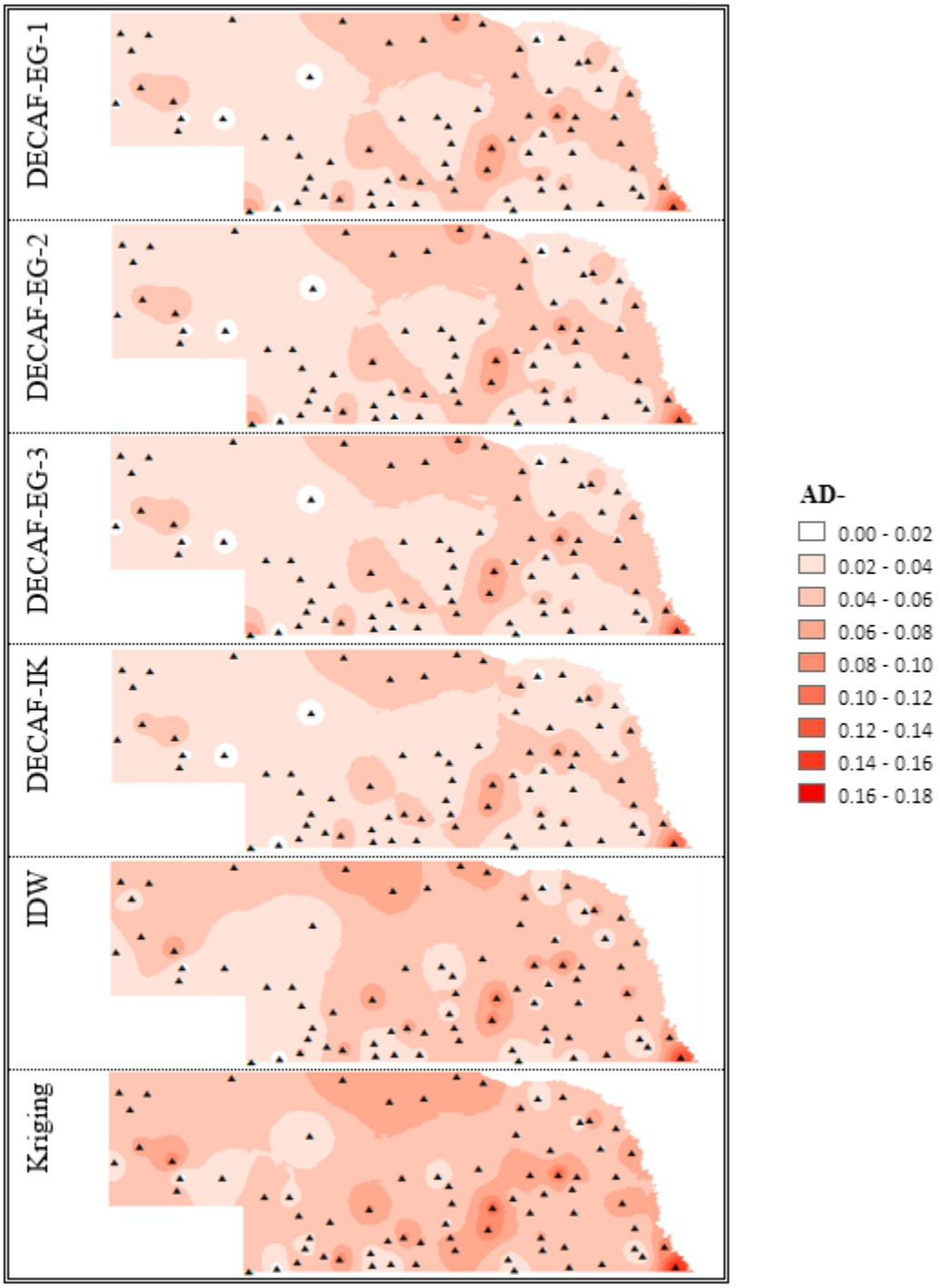


Figure 9.3: $AD-$ error maps, interpolated

9.3.1 General observations

The first impression made by these maps is their similarity. At first glance, no striking differences are immediately observable. This lends itself to a couple easy conclusions. First, the methods all manage reasonably robust approximations; no method is a disaster. Second, the fact that certain regions (certain interior points, in particular) manage to be consistently poorly estimated suggests that the majority of error is inherent to the underlying data (i.e., disagreements between CoCoRaHS and NWS Coop). For this reason, we will avoid drawing conclusions about the error magnitude of isolated methods; rather, we will only compare algorithms to one-another.

9.3.2 Outlier points

Let us now briefly set aside the general to focus on two specific points of interest identified in Figure 9.4.

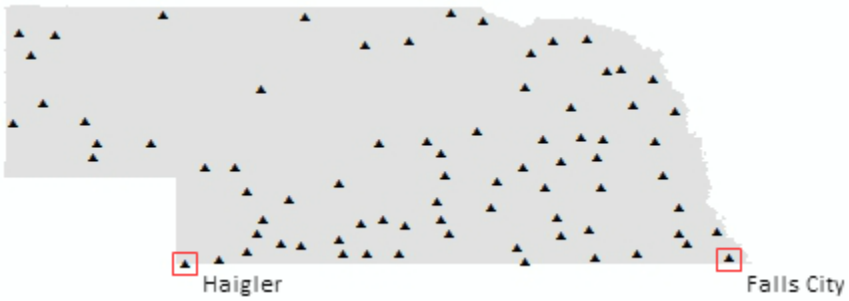


Figure 9.4: Haigler and Falls City

- **Haigler.** Haigler is interesting because it breaks the DECAF-EG family. The DECAF-EG algorithm has a known boundary problem: event borders are defined using sample points in P; if the verification point is near a boundary (e.g., the Nebraska-Kansas border) it may *always* fall outside of the event estimation. Our DECAF-EG implementation has border-snapping logic that fixes this prob-

lem most of the time—except, it appears, if the verification point is wedged into the very end of a 90° corner. All three DECAF-EG algorithms predict Haigler as absent at every day in the six-year span.

- **Falls City.** Falls City is also strange. Not only do *all* algorithms perform poorly at Falls City, but the error moves *both* directions—all methods overestimate badly when they overestimate and underestimate badly when they underestimate. In part, this is another border problem—Falls City is tucked inside an acute angle—and the DECAF-EG family does the *worst* here. Second, Falls City is in the rainiest part of Nebraska [42]. Because we are measuring raw magnitude in this section, Falls City will proportionally result in higher errors.

9.3.3 Method differences

Note that the DECAF algorithms tend to behave similarly, and that IDW and kriging also tend to behave similarly. This matches the results of the Tukey HSD tests. IDW and Kriging both suffer a bit more overestimation error than the DECAF-family methods (excepting DECAF-EG-3, of course). This is expected; the DECAF approach is very good at correctly identifying *absent* points, which avoids one source of overestimation.

Second, note that DECAF-EG-2 and DECAF-EG-3 $AD+$ maps are *very* similar, both differentiating themselves from DECAF-EG-1 and DECAF-IK with lower levels of $AD+$ error. This suggests that simply treating the *border* region of events differently than *core* regions results in a better prediction. The particular algorithm used in the border region isnt important; kriging and IDW are comparable. Rather, the simple act of considering exterior *absent* points is sufficient to reduce overestimation error. Note that the $AD-$ maps are extremely similar across DECAF-EG methods,

suggesting that underestimation error is dominated by PA prediction.

Third, and more perplexing, the entire DECAF family suffers *less* underestimation error than IDW and kriging. High specificity does not help here: underestimation results from doing poorly at *present* points. We may have been able to predict this behavior for DECAF-IK and DECAF-EG-1, both of which ignore all points external to event boundaries that they estimate, tending to ignore *absent* points and increasing the magnitude of estimations inside of the event as a result. Yet DECAF-EG-2 and DECAF-EG-3—both of which consider points outside of the estimated event boundary when predicting *border* points—also outperform IDW and Kriging. This suggests that the difference is not a result of improved *border* predictions but instead improved *core* predictions. This may happen when IDW or kriging uses nearby *absent* points to predict at a location inside of an event core, causing the location to be underestimated.

9.4 Error maps, accumulated percent error

The above maps showed the average day-scale error magnitudes. These maps allowed us to explore day-scale tendencies—methods tending to overestimate or underestimate as well as to perform interpretable Tukey HSD tests. However, because the maps report error magnitude (instead of a percent or proportion) there is no normalization across the event space. For example, east Nebraska is wetter than the west, and so error magnitudes may be higher while error percentages remain the same.

Yet, day-scale differences in magnitude are so small and true zeroes so frequent that calculating day-scale percent error is a fools errand. Instead, we consider the accumulated percent error: the day-scale predictions are summed across time at each station for each method; the truth is also summed across time for each station. These

two sums are used to find the percent error for each method at each predicted station location. The results are mapped in Figure 9.5 on page 152.

With these maps, the proclivities of each method are rendered more starkly. First, members of the DECAF family rarely overestimate, and where they do they overestimate *less* than IDW and kriging. Second, kriging and IDW overestimate more often than their counterparts (DECAF-EG-3 excepted), but also tend to better balance the two varieties of error.

Finally, we begin to detect some differences between DECAF-EG-1 and DECAF-EG-2/DECAF-EG3. DECAF-EG-1 suffers less underestimation error, as expected. DECAF-EG-2 and DECAF-EG-3 appear to suffer less overestimation error, also as expected.

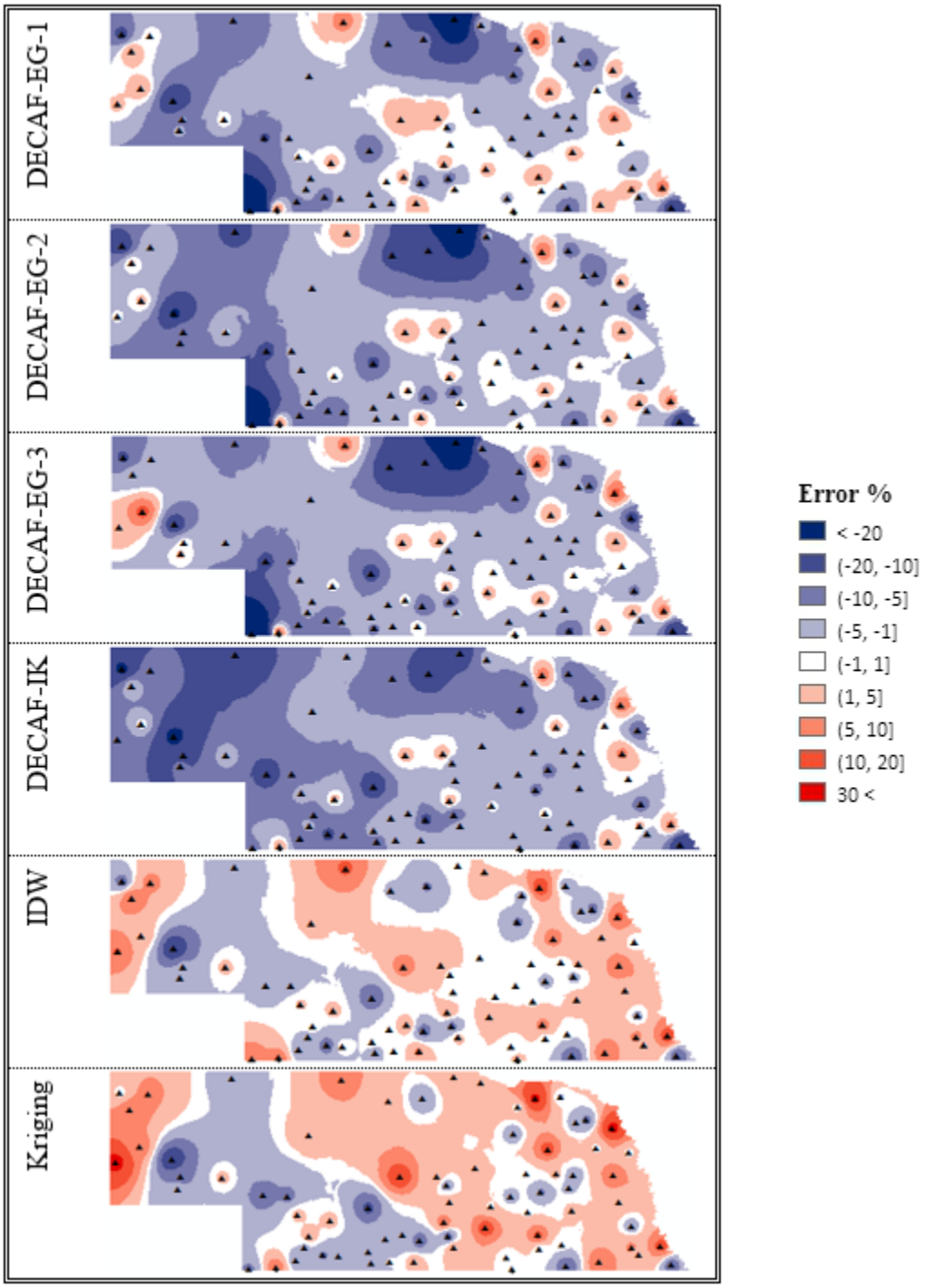


Figure 9.5: Maps of percent error for accumulated (2007–2012) estimations

9.5 Points of interest

Let us now put aside the maps of interpolated error to consider interesting individual estimated points. To conduct this analysis, we will be using the same accumulated percent error mapped in Figure 9.5 on page 152.

We will not consider DECAF-EG-1 and DECAF-IK but instead treat DECAF-EG-2 as a reasonable proxy for both. All three behave very similarly according to Figure 9.5: the differences across maps result from the degree to which each method tends to underestimate. DECAF-EG-1 suffers more overestimation error than DECAF-EG-2 which in turn is worse than DECAF-IK; on the reverse, DECAF-IK is more prone to underestimate than DECAF-EG-2, and DECAF-EG-2 is worse than DECAF-EG-1. DECAF-EG-2 acts as a reasonable average of DECAF behavior.

Furthermore, we will not consider DECAF-EG-3. DECAF-EG-3 is also functionally similar to DECAF-EG-2. There is a little less underestimation error than DECAF-EG-2, but it suffers overestimation errors at the same locations but to a greater degree. Interestingly, it also performs poorly at those points where kriging noteably underperforms (these are discussed below).

Figure 9.6 on page 154 reports all verification points (NWS Coop stations) where at least one of the three methods considered (DECAF-EG-2, Kriging, IDW) suffered greater than 10% absolute percent error. The location icon indicates which method performed the *worst* (e.g., IDW performs the worst at Auburn and Valentine). Furthermore, each point is coded with the colored string "D I K" to indicate how well each method performed at the location—red indicates "high" error (greater than 20%); yellow, moderate error (between 10-20%) and gray, low (less than 10%).

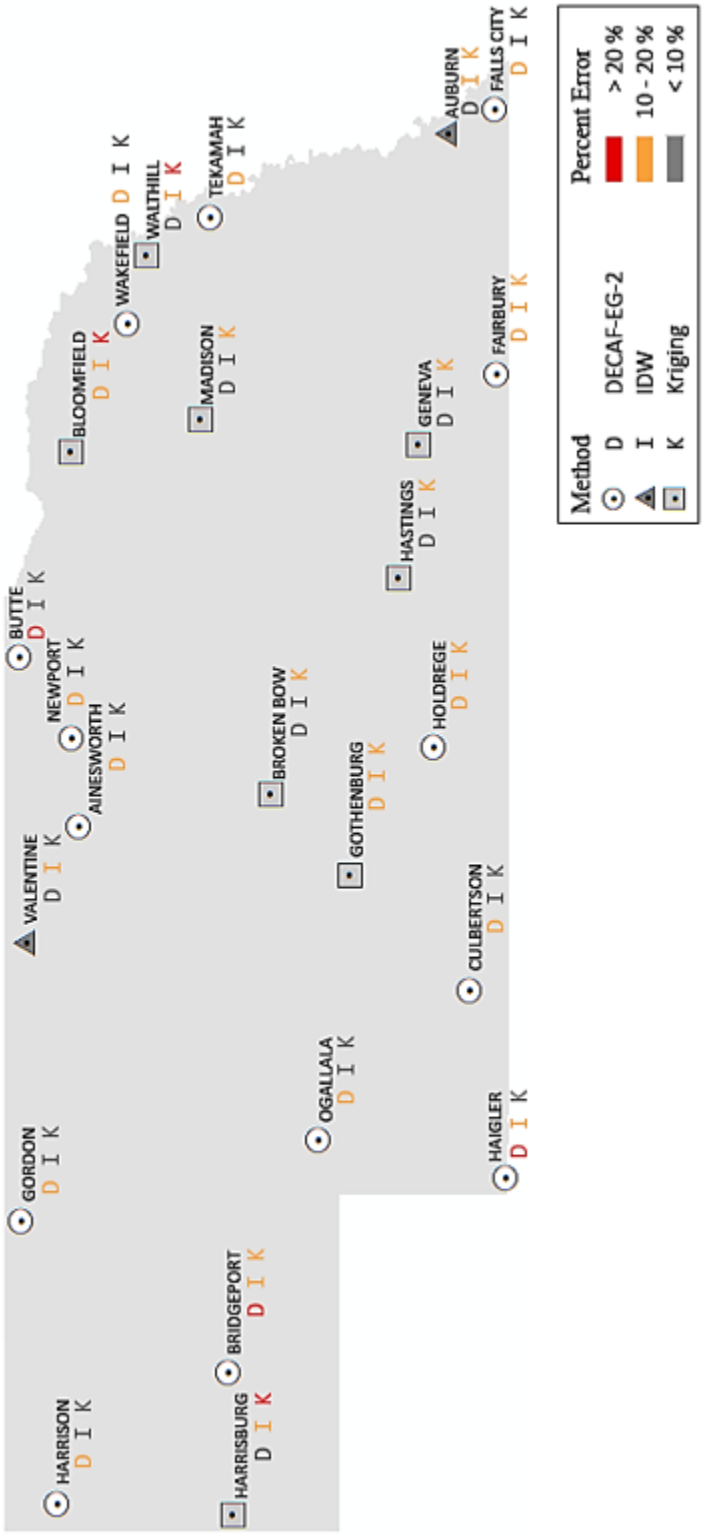


Figure 9.6: Locations of notable percent error

Table 9.5: IDW points of note

	Percent Error		
	DECAF-EG-2	IDW	Kriging
Auburn	6.91	13.35	12.52
Valentine	4.43	10.67	7.61

Observe that IDW is the inferior method at two points, Auburn and Valentine, but never breaches the 20% error threshold. Table 9.5 shows the error estimations at these two points. Note that in the case of Auburn, IDW is effectively the same as Kriging. The difference at Valentine is not particularly stark, either.

Next, note that DECAF-EG-2 breaks the 20% error threshold at three points (Bridgeport, Butte, and Haigler); kriging as well (Bloomfield, Harrisburg, Walthill).

We previously discussed how Haigler's location inside the corner caused the DECAF-EG algorithms to underestimate every associated *present* value. Butte, in the north-eastern quadrant, appears to be a similar case. IDW and kriging suffered little error, but DECAF-EG-2 underestimated the accumulated precipitation by 56%. While Butte isnt nestled into a corner, it is in a part of the state that is relatively sparsely covered by P (CoCoRaHS). Because DECAF-EG is so dependent upon the location of source points to estimate border extent, source point sparseness near border regions may render the border-snapping logic less efficacious.

The final point at which DECAF-EG-2 strongly underperforms—Bridgeport—reveals the limitations of this sort of map. Two methods in the same error categorization may, in fact, be less alike that two methods across categorizations (e.g., errors of 9% and 11% will be grouped separately but 11% and 20% will be grouped together). At Bridgeport, DECAF-EG-2 suffers -22% underestimation error, but both IDW and kriging suffer -18%. DECAF-EG-2 has the greatest error, but all three methods perform terribly.

Table 9.6: Kriging points of note

	Percent Error			
	DECAF-EG-2	IDW	Kriging	DECAF-EG-3
Bloomfield	10.23	15.28	22.85	10.92
Harrisburg	-0.59	10.38	23.83	4.31
Walthill	2.97	11.21	23.59	10.92

The points at which kriging breaks the 20% absolute error threshold are interesting (reported in Table 9.6). The DECAF-EG-3 error has also been included. While in most cases it behaves very similar to DECAF-EG-2, the effects of its kriging-based *border* approximations can be seen at Harrisburg and Walthill. Here, it suffers from kriging's tendency to overestimate these locations.

All three of these points are found along the Nebraska border; one at the westernmost extent and two in the northeast corner. It appears that kriging suffers from a border problem of its own. Instead of tendency to underestimate, kriging overestimates. Notably, at two of these points, DECAF-EG-2 performs excellently.

Because these points fall inside the "border" region of DECAF-EG estimated events, it not unexpected to see a closer relationship between kriging and DECAF-EG-3 than between kriging and DECAF-EG-2. Bloomfield is interesting; the similarity between the DECAF-EG measures suggests that kriging's low specificity played an important role it its overestimation.

Finally, lets consider points where DECAF-EG-2 differs by greater than 10% from the other methods. All differences between DECAF-EG-2 and IDW are less than 10%, so we can ignore IDW. We exclude Butte and Haigler from this analysis because we already know that DECAF-EG-2 performs terribly at these points. There are six such points; they are listed in Table 9.7.

We have already discussed three of these: Bloomfield, Harrisburg, and Walthill.

Table 9.7: Notable differences in error

	Percent Error	
	DECAF-EG-2	Kriging
Bloomfield	10.23	22.85
Broken Bow	3.05	15.78
Harrisburg	-0.59	23.83
Harrison	-15.31	0.22
Hastings	0.75	14.10
Walthill	2.97	11.21

All three appear on the periphery. Harrison is located on the periphery as well, but the associated border effect favors kriging.

Broken Bow and Bloomfield, however, are more intriguing. Both are located in the interior, so border effects cannot play a role in their errors. While two points are far too few to make any reasonable inferences, this hints that DECAF-EG-2 may outperform kriging on the event interior.

Consider Figure 9.6 again. Note that all points where DECAF-EG-2 suffers 10-20% error while the other methods suffer less than 10% are located along the periphery of Nebraska (there are nine such points). We already know that the DECAF-EG family of algorithms is not as robust near the border as alternatives (save for the fact that when kriging botches a point near the border, it *really* botches it), but this also suggests that DECAF-EG-2 is doing better away from the border.

Also, consider where kriging is in the 10-20% error bracket but the other methods are in the <10% category. There are only four of these, so we must be careful not to draw conclusions, but all four are located in the interior of Nebraska. Let us then consider the importance of location—*interior* vs. *periphery*—on error by returning to Table 9.7. A 10% threshold was used to construct this table. What if, instead, we consider a 5% threshold: all points where kriging and DECAF-EG-2 differ in

accumulated percent error by greater than 5% (positive or negative)? We plot these points in Figure 9.7.

Figure 9.7: Points where either Kriging or DECAF-EG-2 outperforms the by $> 5\%$ error.

Both methods have border problems. Many of these happen along the northern Nebraskan border, which is a region —the western stretch in particular—-that is sparsely covered by the CoCoRaHS dataset. In the northeast corner, we find two points that are located relatively far away from the border; however, the methods split the two points. Four points remain, all associated with kriging. These four are located in the Nebraskan interior, and all are located in regions very well covered by the CoCoRaHS dataset (refer to **??** on page **??**).

It should be noted that in all cases in Figure 9.7 where DECAF-EG-2 outperforms kriging, it is the result of kriging overestimation error. Similarly, in all cases where kriging outperforms DECAF-EG-2, it is the result of DECAF-EG-2 underestimation error.

9.6 Error by error-type

Evidence presented until now suggests that the DECAF family of algorithms compare favorably with Kriging and IDW. However, we cannot conclude that one method consistently outperforms the others. Consider Table 9.8, which reports absolute mean error, averaged across points. The mean errors are statistically indistinguishable. [2]

Table 9.8: Mean absolute error across points

	Mean Error
DECAF-IK	56.9
DECAF-EG-2	56.9
IDW	58.5
Kriging	58.2

We have considered error by geographic location; now let us consider error by error category. We can define three error subtypes:

1. PP error: The point was, in fact, *present* and the algorithm predicted *present*. Error results from the difference in predicted value.

2. AP error: The point was, in fact, *absent* but the algorithm predicted *present*. The error results from overestimating the zero value as a non-zero value.

3. PA error: The point was, in fact, *present* but the algorithm predicted *absent*. Error results from underestimating the value as zero.

Error is reported by type in Table 9.9.

Note that there is no AA error; no error can result from correctly estimating a zero-value as zero. Also, note that PP error + PA error + AP error = mean error as reported in Table 9.8.

[2]DECAF-EG-1 and DECAF-EG-3 behave almost identically to DECAF-EG-2

Table 9.9: Mean absolute error across points by error type

	PP Error	*Group*	*AP Error*	*Group*	*PA Error*	*Group*
DECAF-IK	37.9		7.1		11.7	a
DECAF-EG-2	39.9		8.1		8.9	a
IDW	44.9		10.8	a	2.8	b
Kriging	43.0		11.0	a	4.1	b

PP error. All methods are statistically distinguishable from one-another (according to a TukeyHSD test at the $\alpha = 0.05$ level). The DECAF methods outperform the traditional approaches.

AP error. The traditional approaches are indistinguishable from one-another. Again, the DECAF methods outperform these approaches.

PA error. The traditional approaches form one statistically indistinguishable group; the DECAF approaches another. This time, however, the traditional approaches significantly outperform the DECAF algorithms. Now, the low Recall of DECAF hurts its predictive power to such a degree that its PP and PA success is washed out.

This breakdown reveals that the two approaches produce complementary information. As a final excercise, we weigh and combine DECAF-EG-2 and kriging to demonstrate that the two produce complementary information. The combination of the two is an improvement over either individual method. Similar approaches are represented in the geospatial information fusion literature [56].

To effect this combination, we calculate the average error of each set.[3] The average error is added to the preciptiation estimation at points in the respective set, resulting

[3]Haigler and Butte were considered outliers and dropped.

in an average error functually equivalent to zero. Then, the DECAF-EG-2 and kriging estimations are averaged at each point. The interpolated map of aggregated error is shown in Figure 9.8. The original kriging and DECAF-EG-2 maps are included for the purposes of comparison.

It is apparent that the combination is an improvement upon each individual method, suggesting that future geospatial fusion engines might exploit the strengths of different methods to improve final predictions.

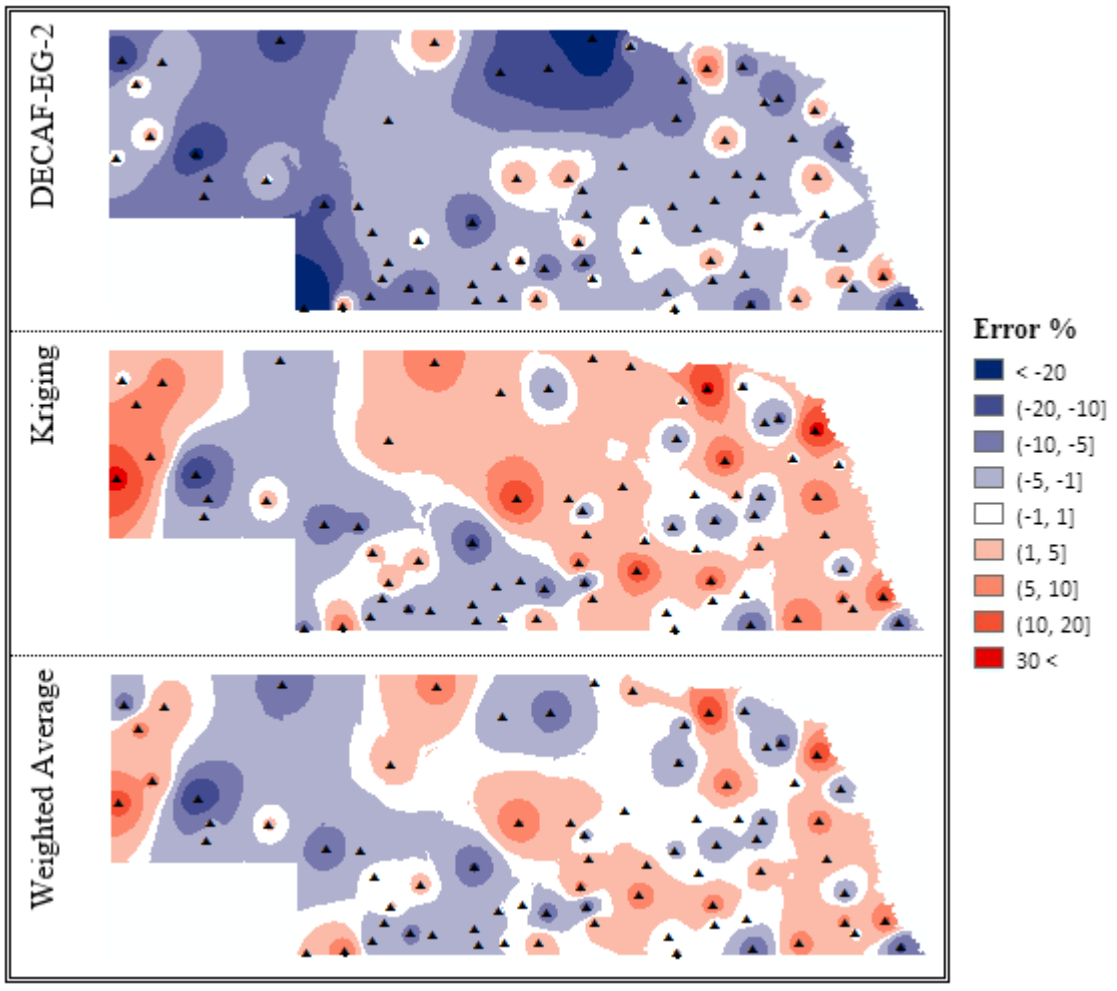


Figure 9.8: Maps of percent error for accumulated (20072012) estimations, weighted combination of DECAF-EG-2 and kriging

9.7 CPU runtime analysis

The theoretical time complexities are reported in Table 9.10 for reference.[4] In this table, the time complexity for each algorithm is split into two columns. The first column reports the time complexity to estimate the value at a single point q. The second column reports the time needed to estimate each subsequent q from the same sample set P (i.e., estimate at each subsequent verification point on the same day). This column is not essential for determining overall worst-case time complexities: for m points in the verification set Q, the time complexity would be a series m long with the first term drawn from the first column of the table and subsequent terms from the second column of the table. The first column is always of equal or greater time complexity, and so dominates the series.

Table 9.10: Algorithm time complexities

	Time Complexity (first q)	*Time Complexity* (subsequent q)
DECAF-EG-1	$O(n \log n)$	$O(n \log n)$
DECAF-EG-2	$O(n \log n)$	$O(n \log n)$
DECAF-EG-3	$O(n^3)$	$O(n^3)$
DECAF-IK	$O(u \cdot v \cdot n^3)$	$O(n \log n)$
IDW	$O(n \log n)$	$O(\log n)$
Kriging	$O(n^3)$	$O(n^3)$

Where $n = |P|$ (number of points in sample set) and u and v are the dimensions of the grid computed by DECAF-IK.

However, this second column is interesting because it helps to illuminate algorithm behavior for large verification sets. The non-kriging approaches all suffer significant overhead for the first q that can be amortized across subsequent q. For large verifi-

[4]These time complexities are developed earlier in the thesis. Chapter 2: IDW and kriging, Chapter 4: DECAF-EG-1 and DECAF-IK, Chapter 7: DECAF-EG-2 and DECAF-EG-3

cation sets, kriging may be less appropriate.

Let us pause to briefly consider each method.

Kriging. The time complexity of kriging is identical for all points in the verification set.

DECAF-EG. The worst-case time complexity for the first and subsequent points are the same. However, this warrants further discussion. Most of the computational overhead in DECAF-EG is the result of the neighborhood building and cluster analysis, which only needs to be performed once per set P (i.e., once per day). The subsequent $O(n \log n)$ time is the result of the `find_associated_event()` function. This function checks to see if the point q is inside of an event; in the worst case, all points in P are boundary points of the event polygon, resulting in $O(n)$ time. In the worst case, q falls inside the event and neighbors must be found—$O(\log n)$ time. Aggregated, this is $O(n \log n)$. Yet, it is a lighter $O(n \log n)$ for subsequent q than for the initial q.

IDW. IDW strictly improves. The initial $O(n \log n)$ time is required to compute the k-d tree. Subsequent approximations reuse this tree, and so require only $O(\log n)$ time.

DECAF-IK. DECAF-IK also strictly improves. The entire $u \cdot v$ grid must be computed for the first q and the event hulls subsequently extracted. Just as for DECAF-EG, these data structures can be reused for subsequent q and the time complexity becomes $O(n \log n)$.

To check these complexities, we collected total CPU time estimates for each of the methods as they worked through the 900 days. The results are presented in Table 9.11.

Table 9.11: Algorithm CPU times

	DECAF-EG-1	DECAF-EG-2	DECAF-EG-3	DECAF-IK	IDW	Kriging
Total Seconds	3253	3529	6396	24874	187	2639
Total Minutes	54	59	107	415	3	44
Seconds per day	3.61	3.92	7.11	27.64	0.21	2.93

The DECAF algorithms behave as we would expect. The IDW-based DECAF-EG-1 and DECAF-EG-2 algorithms perform more-or-less identically. DECAF-EG-3 is slower, reflecting its reliance on Kriging. DECAF-IK, unsurprisingly, is *very* slow. Note that the dimensions for the DECAF-IK grid were a low 150×70.

IDW and Kriging are a little trickier to interpret. Methods were implemented in Python. Because Python is an interpreted language, is suffers considerable overhead for even simple function calls.

IDW was implemented in Python and used the SciPy package's k-d tree implementation for neighbor-finding. Function calls are minimal. The result is a fast $O(n \log n)$. DECAF-EG-1 and DECAF-EG-2, on the other hand, have many function calls and perform considerably more computation (especially for subsequent q in the same day). The result is a slow $O(n \log n)$.

The kriging implementation that we used relies heavily on the Numpy package, which uses C routines for major computations. So, despite suffering a greater time complexity, kriging has a lower runtime than DECAF-EG-1 and DECAF-EG-2 because most of kriging's intensive routines are performed in a compiled language. It is reasonable to expect that DECAF-EG-1 and DECAF-EG-2 would see considerable improvement (and perhaps beat kriging) if they were implemented in C (or a comparable compiled language).

9.8 Summary

There is evidence that the DECAF-EG approach can outperform kriging. It appears that the DECAF-EG's high *presence/absence* specificity helps prevent overestimation error. On event interiors, this allows it to match (and occasionally outperform but very rarely underperform) kriging.

DECAF's weakness is clearly underestimation error. DECAF-IK's low recall is simply unacceptable. DECAF-EG performs better, but the DECAF-EG-1 algorithm fails to treat the *border* region of an event differently than the *core* of that event.[5] This results in overestimation error inside of events. The more sophisticated DECAF-EG-2 manages to successfully leverage the high precision of DECAF without a significant overestimation cost. DECAF-EG-3 appears to be functionally equivalent to DECAF-EG-2, but at a higher time complexity.

Preliminary results suggest that the DECAF approaches produce information complementary to traditional approaches. DECAF might be used in tandem with these methods to effect more accurate predictions.

[5]This is *not* the border of the event space (such as the boundary of Nebraska) but the border of a precipitation event.

Chapter 10

Conclusion

The Delineated-Event Continuous-Aspect Fusion algorithm arose from the realization that quality long-term, fine-grained temporal information fusion of dynamic delineated continuous phenomena may require that the event space be formally structured. We hypothesized that a method that could automatically distinguish between events (as well as the lack of an event) might be able to exploit that information to improve the quality of subsequent point-value estimations. This method had to be automated; otherwise, it could not be used to power a true information fusion engine.

Using precipitation as a template, we formalized the notion of a delineated continuous phenomenon. The boundaries of a delineated continuous event are distinctly defined, but the interior is a spatial random field. We noted that techniques are well developed for such static phenomena (zonal kriging). Experts must manually divide a space into distinct sub-regions (typically with help from automated methods, such as indicator kriging) and then apply methods appropriate to each sub-region. For a phenomenon that does not change through time, this may be a perfectly effective approach. But, if the phenomenon is *dynamic*—as is precipitation—sub-regions must be identified at each time step.

This thesis reports our attempts to design an effective and efficient DECAF algorithm and prove its usefulness. The first such algorithm introduced was DECAF-IK, which was built on top of traditional kriging techniques. It was no less efficient than these techniques, but it struggled to effect quality predictions and was generally overshadowed by the second variety of algorithm, the DECAF-EG family.

The original DECAF-EG had low time complexity, and the DECAF-EG-2 update maintained this efficiency. DECAF-EG-3 sacrificed efficiency for the sake of an increased effectiveness that was not attained—or at least not detected by our verification methods. So, we currently consider DECAF-EG-2 to be the most advanced realization of our attempts to formalize and exploit the structure of dynamic delineated continuous phenomenon.

We conducted tests based on simulated data to better understand how variations in event structure effected DECAF and traditional alternatives. These tests revealed DECAF to be robust to changes in event size, shape (strictly convex), orientation, and noise. They also revealed a border problem, which lead to the DECAF-EG-2 update.

Next, we conducted tests on real-world precipitation data. The precipitation results defied easy analysis: at the day scale, total precipitation is limited and so it is difficult to identify meaningful differences across methods; at larger scales, differences can be detected by the underlying cause of those differences is lost to aggregation. Nevertheless, our analysis revealed important strengths and weaknesses inherent to the DECAF approach.

10.1 Key observations

The event detection logic works. DECAF-IK and DECAF-EG both outperformed IDW and kriging at correctly predicting *presence*/*absence*. Because neither IDW nor kriging (of the ordinary variety) are intended for *presence*/*absence* prediction, this is not surprising.

The event detection logic is helpful. In the real-world experiments, the DECAF methods were less prone to overestimation error than kriging and IDW, a result of correctly predicting *absence* points.

Fuzzy logic is necessary. DECAF-IK did not perform well either the simulated or real-world datasets. DECAF-IK treats space in a strictly dichotomous manner—*present* or *absent*—and so suffered considerable area in regions where it is difficult to accurately make this distinction. DECAF-EG-1 suffered a similar problem (though not to the same degree). DECAF-EG-2 and DECAF-EG-3 use fuzzy logic in a selective way, treating *border* zones as mixture of *present* and *absence*. As a result, both eliminate the border-type overestimation error found in DECAF-EG-1.

The event detection logic is not *always* helpful. The reduction in recall causes DECAF methods to be more prone to underestimation error. However, the fuzzy logic introduced in DECAF-EG-2 and DECAF-EG-3 helped to alleviate this problem.

Existing border-snapping logic is imperfect. The DECAF-EG methods suffer inexcusable error rates in certain cases at the event space border. However, kriging is vulnerable to significant border problems as well—overestimation problems in particular, precisely the sort of error that DECAF corrects. Fur-

thermore, snapping logic is very correctable; better border-snapping algorithms exist in the literature and simply need to be properly implemented inside of DECAF.

Kriging is a very robust method. It is very difficult to beat kriging at its own game—interpolation—as the experimental results so starkly advertise. As expected, kriging was impaired by the spatial discontinuities of the day-scale precipitation data; yet, not so strongly as to allow DECAF-EG-2 to clearly outperform it.

10.2 Limitations of our analysis

We investigated the efficacy of our algorithms two ways. The first was a painstakingly thorough investigation into the effects of event size, orientation, shape, and two kinds of noise on simulated data. The second was involved over 1000 days of precipitation data, each day composed of hundreds of observations. Yet, the twin tyrannies of time and energy necessarily limit all, and no less the graduate student than any other. We must now discuss the deficiencies of our analyses and what improvements might be made.

10.2.1 Simulations

The simulation experimental design explored five factors (size, shape, orientation, and two kinds of noise) and arrived at robust conclusions concerning these. In this sense, the simulations accomplished what they were designed to do: answer specific questions about the behavior of our algorithms.

Yet, a secondary objective—to better understand where kriging does poorly—was not investigated well by our experimental design. The only interesting result regarding kriging was how it behaved when exposed to high levels of noise (it performed equivalently to IDW). Yet, additional tests could have been performed to better understand the strengths and weaknesses of DECAF vis-*à*-vis kriging.

Note: The following two paragraphs will not apply if we have the time to perform the associated experiments.

For example, one of the weaknesses of kriging is that it fits a semivariogram to the entire event space, assuming a certain spatial homogeneity. While we tested the effects of this based on the thresholding effect, we never tried to place two artificial events in the same event space. Ideally, a large event and a small event would coexist and the efficacy of DECAF and kriging in this event space could be compared.

Secondarily, the DECAF methods may perform better on oddly-shaped events. For example, an absent point wedged inside of concave space on an irregularly shaped event is likely to be overestimated by the traditional interpolation methods while the DECAF methods may, in fact, recognize it as absent. Simulations based on more complicated shapes would be revealing.

10.2.2 Precipitation data

The precipitation data itself is messy. While we try to establish a ground truth by selecting choosing National Weather Service stations that have been reporting for over a century, it remains at best an approximation. For example, we have documented certain stations that appear to be consistently bad (at least according to their neighbors). In fact, we witnessed more variation in error between verification points than we did across methods. This is an inescapable fact when working with precipitation

data.[1] Yet, it necessarily complicates (and reduces the effectiveness of) data analysis.

Also, it would have been ideal to expand beyond the seven year period that we tested. We were limited by the CoCoRaHS data, which begins in Nebraska in 2002 but does not reach complete spatial coverage until mid-summer 2006. An alternative approach would be to divide the NWS Coop stations into source and verification sets and run repeated tests (resampling each time). This could be performed on over a centurys worth of data; however, the number of points at each time step would be limited.

Finally, it would have also been ideal to test different states. Nebraska is flat, and except for a west-to-east dry-to-wet gradient doesnt exhibit interesting meteorological differences. Washington state, however, has a clearly discontinuous east-west resulting from the Cascade mountains. It would also be interesting to consider a region where storm-cells dominate, where act even more discretely than the sweeping fronts that typify Nebraskan precipitation.

10.2.3 Efficiency

One of the primary strengths of DECAF-EG-2 is its lower time complexity. It does not requre the inversion of matrices, and so has a much lower time (and memory) complexity than kriging. However, we were unable to empirically demonstrate that it performs faster than kriging (though we did do so compared to DECAF-IK).

Our algorithms were written in Python. Python is an interpreted language, and so is slow—particularly so during function calls (and our methods made extensive use

[1] We arrived at the NWS Coop set as our ground truth after wasting a considerable amount of time trying to use the High Plains Regional Climate Center data. We were originally interested in this set because it was collected by automated sensors which, being computer scientists, we presumed to be more accurate than human-sourced data. After much analysis and confusion, we eventually learned (from HPRCC itself) that the precipitation sensors are prone to significant underestimation error!

of function calls). Yet, we chose to use it because it has certain strengths. First, it is a language noted for getting out of the way of the programmer for the purposes of faster development, which allowed us to rapidly prototype many algorithm variations [32]. Second, Python's large standard library and freely-available third-party libraries allowed us to develop a sophisticated backend very quickly. Those libraries also allowed us to interface with external products, such as routines in the R language and Esri's ArcGIS.

The kriging implementation against which DECAF was compared was also written in Python. However, it made extensive use of the NumPy package, which is a scientific computing package that wraps highly-optimized C routines. Because of time constraints, DECAF was never so similarly optimized. As a result, we could not directly compare the two methods in a meaningful way.

10.3 Future Work

Future work can be categorized into improvement to DECAF itself and steps toward building upon it towards an information fusion engine.

10.3.1 Improving DECAF

Error estimation. Unlike kriging, DECAF does *not* produce an error map. Either additional error logic must be invented or DECAF must rely upon kriging instead of IDW to perform interpolation. The downside, of course, is increased time complexity.

Better exploit kriging. Choosing to set aside the time-complexity benefits of a non-kriging approach, we can instead focus on strict accuracy. Several possi-

bilities are immediately evident. First, we could expand upon DECAF-EG-3 to also use kriging in the event core. This would require logic that reverts to IDW when very small neighborhoods are found (or would require a custom implementation of kriging that handles this situation). Second, we could circumvent the problem of fitting the semivariogram to small amounts of data by developing generalized semivariograms based on cluster size. Finally, instead of trying to introduce kriging logic into DECAF, we could attempt the opposite: post-kriging, DECAF's *presence/absence* predictions could be used to zero-out absence values.

Adapt neighborhoods. DECAF-EG-2 is limited because of its reliance on the IDW algorithm. It could better use IDW by intelligently selecting the *number* of neighbors instead of relying on external parameterization (e.g., the 12 neighbor cap that we imposed in our experiments). In the simulations, for example, DECAF used too many neighbors, resulting in significant underestimation at the event maximum which kriging was able to avoid.

10.3.2 Building upon DECAF

Beat kriging at its own game. DECAF simply parameterizes an interpolation function by selectively composing neighborhoods. We could move one step down the ladder to consider how to weight neighbors within the neighborhood itself. Temporal correlations, measures of mutual information, and cluster membership data could be used to uncover relationships specific to individual point pairs, informing the value of in individualized weights in the interpolation process.

Better exploit temporal information. A product of the DECAF process is a well-structured history of event shapes, sizes, and locations. This history could

be mined to determine whether certain varieties of clusters exhibit certain patterns. Such patterns might be able to be exploited to improve predictions. Also, clusters could be composed at larger temporal scales (e.g., day pairs, week).

Consider additional phenomenon aspects. Instead of limiting the algorithm, for example, to precipitation data, it could also consider related aspects (e.g., stream flow, groundwater tables). Results could be compared to cokriging, but methods might be based on measures of mutual information.

10.4 Final thoughts

DECAF-EG-2 is a reasonable alternative to kriging for dynamic delineated continuous data where event shapes are difficult to predict *a priori* and are not simple (e.g., circles, ellipses), but is not an obvious improvement. Because it has lower time complexity, it may be especially useful in data-intensive environments.

At the moment, however, DECAF is not ready for immediate application. However, the method shows promise, and with additional development may consistently outperform kriging at estimating dynamic delineated continuous phenomena. In time, a properly evolved derivation may become the foundation for a true spatio-temporal information fusion engine.

[1]

[2]

[37]

[3]

standard: [50]

[4]

[5]

[6]

[7]

[8]

[10]

[11]

[13]

[12]

[14]

[15]

[9]

[16]

[17]

[19]

esri idw: [18]

[20]

[21]

[22]

[24]

[25]

[26]

[27]

[28]

[29]

[30]

[31]

[32]

[33]

[34]

[42]

[35]

[23]

[39]

[40]

[36]

[41]

[43]

[44]

[45]

[46]

[47]

[49]

[48]

[51]

[52]

[53]

[38]

[54]

[55]

[56]

[57]

[58]

[59]

Appendix A

Appendix to Chapter 6

A.1 ANOVA tables

Table A.1: ANOVA, event structure, *AD+*

	DF	Sums of Squares	Mean of Squares	F Value	P value
size	1	0.6466	0.6466	898	$< 2.2e^{-16}$ ***
shape	1	0.2395	0.2395	332	$< 2.2e^{-16}$ ***
theta	1	0.0000	0.0000	0.06	0.81
method	3	3.4764	1.1588	1610	$< 2.2e^{-16}$ ***
size * shape	1	0.0022	0.0022	3.03	0.08
size * theta	1	0.0016	0.0016	2.27	0.13
size * method	3	0.4888	0.1629	226	$< 2.2e^{-16}$ ***
shape * method	3	0.1432	0.0477	66.3	$< 2.2e^{-16}$ ***
theta * method	3	0.0010	0.0003	0.469	0.70
size * shape * method	3	0.0187	0.0062	8.67	$1.0e^{-5}$ ***
size * theta * method	3	0.0020	0.0007	0.95	0.42
Residuals	2376	1.7099	0.0007		

Significance: $^{***}\alpha < 0.001$ $^{**}\alpha < 0.01$ $^{*}\alpha < 0.05$

Table A.2: ANOVA, event structure, $AD-$

	DF	Sums of Squares	Mean of Squares	F Value	P value
size	1	2.053	2.053	1293	$< 2.2e^{-16}$ ***
shape	1	0.630	0.630	396	$< 2.2e^{-16}$ ***
theta	1	0.009	0.009	5.47	0.02 *
method	3	1.553	0.518	326	$< 2.2e^{-16}$ ***
size * shape	1	0.070	0.070	44.1	$3.96e^{-13}$ ***
size * theta	1	0.000	0.000	0.16	0.68
size * method	3	0.153	0.051	32.2	$< 2.2e^{-16}$ ***
shape * method	3	0.041	0.014	8.63	$< 2.2e^{-16}$ ***
theta * method	3	0.005	0.002	1.09	0.35
size * shape * method	3	0.015	0.005	3.05	0.03 *
size * theta * method	3	0.004	0.001	0.76	0.51
Residuals	2376	3.771	0.002		

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

Table A.3: ANOVA, event structure, MAD

	DF	Sums of Squares	Mean of Squares	F Value	P value
size	1	4.32	4.32	1053	$< 2.2e^{-16}$ ***
shape	1	2.08	2.08	508	$< 2.2e^{-16}$ ***
theta	1	0.00	0.00	0.02	0.87
method	3	8.05	2.68	654	$< 2.2e^{-16}$ ***
size * shape	1	0.15	0.15	36.8	$1.5e^{-9}$ ***
size * theta	1	0.00	0.00	0.09	0.77
size * method	3	0.31	0.10	24.9	$6.9e^{-16}$ ***
shape * method	3	0.13	0.04	10.9	$4.3e^{-7}$ ***
theta * method	3	0.00	0.00	0.21	0.88
size * shape * method	3	0.07	0.02	5.55	0.0009 ***
size * theta * method	3	0.00	0.00	0.33	0.81
Residuals	2376	9.75	0.00		

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

Table A.4: ANOVA, noise, $AD+$

	DF	Sums of Squares	Mean of Squares	F Value	P value
sill	2	1.194	0.5971	4481.2	$< 2.2e^{-16}$ ***
nugget	2	0.531	0.2653	1990.9	$< 2.2e^{-16}$ ***
method	3	1.103	0.3675	2758.5	$< 2.2e^{-16}$ ***
sill * nugget	4	0.049	0.0122	91.8	$< 2.2e^{-16}$ ***
sill * method	6	0.071	0.0119	89.1	$< 2.2e^{-16}$ ***
nugget * method	6	0.028	0.0047	35.2	$< 2.2e^{-16}$ ***
sill * nugget * method	12	0.004	0.0003	2.2	0.008
Residuals	3564	0.475	0.0001		

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

Table A.5: ANOVA, noise, $AD-$

	DF	Sums of Squares	Mean of Squares	F Value	P value
sill	2	2.179	1.0896	5189.1	$< 2.2e^{-16}$ ***
nugget	2	1.339	0.6694	3188.0	$< 2.2e^{-16}$ ***
method	3	0.016	0.0054	25.7	$< 2.2e^{-16}$ ***
sill * nugget	4	0.075	0.0186	88.8	$< 2.2e^{-16}$ ***
sill * method	6	0.034	0.0056	26.7	$< 2.2e^{-16}$ ***
nugget * method	6	0.015	0.0026	12.2	$< 2.2e^{-16}$ ***
sill * nugget * method	12	0.012	0.0010	4.8	$5.7e^{-8}$ ***
Residuals	3564	0.748	0.0002		

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

Table A.6: ANOVA, noise, MAD

	DF	Sums of Squares	Mean of Squares	F Value	P value
sill	2	27.265	13.633	4184.9	$< 2.2e^{-16}$ ***
nugget	2	8.185	4.093	1256.3	$< 2.2e^{-16}$ ***
method	3	0.507	0.169	51.9	$< 2.2e^{-16}$ ***
sill * nugget	4	1.310	0.327	100.5	$< 2.2e^{-16}$ ***
sill * method	6	0.390	0.065	20.0	$< 2.2e^{-16}$ ***
nugget * method	6	0.160	0.027	8.2	$8.6e^{-9}$ ***
sill * nugget * method	12	0.163	0.014	4.2	$1.5e^{-6}$ ***
Residuals	3564	11.610	0.003		

Significance: $***\alpha < 0.001$ $**\alpha < 0.01$ $*\alpha < 0.05$

Table A.7: ANOVA, event × noise, $AD+$

	DF	Sums of Squares	Mean of Squares	F Value	P value
sill	1	2.0421	2.0421	1843.1	$< 2.2e^{-16}$ ***
nugget	1	0.6568	0.6568	592.8	$< 2.2e^{-16}$ ***
method	3	20.9164	6.9721	6292.7	$< 2.2e^{-16}$ ***
theta	1	0.0399	0.0399	36.0	$< 2.2e^{-16}$ ***
shape	1	0.1863	0.1863	168.2	$< 2.2e^{-16}$ ***
size	1	0.7844	0.7844	708.0	$< 2.2e^{-16}$ ***
sill * nugget	1	0.1161	0.1161	104.8	$< 2.2e^{-16}$ ***
sill * method	3	0.3498	0.1166	105.2	$< 2.2e^{-16}$ ***
nugget * method	3	0.0841	0.0280	25.3	$< 2.2e^{-16}$ ***
sill * theta	1	0.0166	0.0166	15.0	0.0001 ***
nugget * theta	1	0.0100	0.0100	9.0	0.0027 ***
method * theta	3	0.1764	0.0588	53.1	$< 2.2e^{-16}$ ***
sill *shape	1	0.0329	0.0329	29.7	$< 2.2e^{-16}$ ***
nugget * shape	1	0.0280	0.0280	25.3	$< 2.2e^{-16}$ ***
method * shape	3	0.4406	0.1469	132.6	$< 2.2e^{-16}$ ***
sill * size	1	0.1116	0.1116	100.7	$< 2.2e^{-16}$ ***
nugget * size	1	0.0818	0.0818	73.8	$< 2.2e^{-16}$ ***
method * size	3	2.2663	0.7554	681.8	$< 2.2e^{-16}$ ***
theta * size	1	0.0037	0.0037	3.3	0.0683
shape * size	1	0.0176	0.0176	15.9	$< 2.2e^{-16}$ ***
sill * nugget * method	3	0.0129	0.0043	3.9	0.0086 ***
sill * nugget * theta	1	0.0138	0.0138	12.4	0.0004 ***
sill * method * theta	3	0.0038	0.0013	1.1	0.3340
nugget * method * theta	3	0.0046	0.0015	1.4	0.2427
sill * nugget * shape	1	0.0054	0.0054	4.9	0.0268 *
sill * method * shape	3	0.0035	0.0012	1.0	0.3736
nugget * method * shape	3	0.0029	0.0010	0.9	0.4467
sill * nugget * size	1	0.0052	0.0052	4.7	0.0310 *
sill *method * size	3	0.0125	0.0042	3.8	0.0105 *
nugget * method * size	3	0.0047	0.0016	1.4	0.2403
sill * theta * size	1	0.0014	0.0014	1.3	0.2503
nugget * theta * size	1	0.0040	0.0040	3.6	0.0584
method * theta * size	3	0.0417	0.0139	12.5	$3.6e^{-8}$ ***
sill * shape * size	1	0.0020	0.0020	1.8	0.1800
nugget * shape * size	1	0.0014	0.0014	1.2	0.2697
method * shape * size	3	0.0695	0.0232	20.9	$1.7e^{-13}$ ***
sill * nugget * method * theta	3	0.0071	0.0024	2.1	0.0950
sill * nugget * method * shape	3	0.0012	0.0004	0.3	0.7900
sill * nugget * method * size	3	0.0067	0.0022	2.0	0.1090
sill * nugget * theta * size	1	0.0085	0.0085	7.7	0.0056
sill * method * theta * size	3	0.0052	0.0017	1.6	0.1989
nugget * method * theta * size	3	0.0040	0.0013	1.2	0.3022
sill * nugget * shape * size	1	0.0001	0.0001	0.1	0.7460
sill * method * shape * size	3	0.0063	0.0021	1.9	0.1259
nugget * method * shape * size	3	0.0026	0.0009	0.8	0.5084
sill * nugget * method * theta * size	3	0.0082	0.0027	2.5	0.0606
sill * nugget * method * shape * size	3	0.0005	0.0002	0.1	0.9368
Residuals	9288	10.2909	0.0011		

Significance: ***$\alpha < 0.001$ **$\alpha < 0.01$ *$\alpha < 0.05$

Table A.8: ANOVA, event × noise, $AD-$

	DF	Sums of Squares	Mean of Squares	F Value	P value
sill	1	5.19379	5.1938	2186.7	$< 2.2e^{-16}$ ***
nugget	1	3.03908	3.0391	1279.5	$< 2.2e^{-16}$ ***
method	3	1.97242	0.6575	276.8	$< 2.2e^{-16}$ ***
theta	1	0.69837	0.6984	294.0	$< 2.2e^{-16}$ ***
shape	1	0.95869	0.9587	403.6	$< 2.2e^{-16}$ ***
size	1	4.60041	4.6004	1936.8	$< 2.2e^{-16}$ ***
sill * nugget	1	0.33061	0.3306	139.2	$< 2.2e^{-16}$ ***
sill * method	3	0.21122	0.0704	29.6	$< 2.2e^{-16}$ ***
nugget * method	3	0.16385	0.0546	23.0	$< 2.2e^{-16}$ ***
sill * theta	1	0.01623	0.0162	6.8	0.0090 **
nugget * theta	1	0.00690	0.0069	2.9	0.0884
method * theta	3	0.07249	0.0242	10.2	$1.1e^{-6}$ ***
sill *shape	1	0.02265	0.0227	9.5	0.002 **
nugget * shape	1	0.09578	0.0958	40.3	$2.3e^{-10}$ ***
method * shape	3	0.11770	0.0392	16.5	$1.1e^{-10}$***
sill * size	1	0.08935	0.0894	37.6	$8.9e^{-10}$ ***
nugget * size	1	0.14302	0.1430	60.2	$9.4e^{-13}$ ***
method * size	3	0.67104	0.2237	94.2	$< 2.2e^{-16}$ ***
theta * size	1	0.12719	0.1272	53.5	$2.7e^{-13}$ ***
shape * size	1	0.22997	0.2300	96.8	$< 2.2e^{-16}$ ***
sill * nugget * method	3	0.05287	0.0176	7.4	0.0001 ***
sill * nugget * theta	1	0.00004	0.0000	0.0	0.9031
sill * method * theta	3	0.00330	0.0011	0.5	0.7080
nugget * method * theta	3	0.00809	0.0027	1.1	0.3330
sill * nugget * shape	1	0.02496	0.0250	10.5	0.0012
sill * method * shape	3	0.00228	0.0008	0.3	0.8105
nugget * method * shape	3	0.00088	0.0003	0.1	0.9464
sill * nugget * size	1	0.00006	0.0001	0.0	0.8731
sill *method * size	3	0.00716	0.0024	1.0	0.3893
nugget * method * size	3	0.00070	0.0002	0.1	0.9613
sill * theta * size	1	0.00050	0.0005	0.2	0.6461
nugget * theta * size	1	0.00613	0.0061	2.6	0.1083
method * theta * size	3	0.02237	0.0075	3.1	0.0242 *
sill * shape * size	1	0.00041	0.0004	0.2	0.6786
nugget * shape * size	1	0.03114	0.0311	13.1	0.0003 ***
method * shape * size	3	0.01875	0.0063	2.6	0.0483 *
sill * nugget * method * theta	3	0.00495	0.0017	0.7	0.5548
sill * nugget * method * shape	3	0.00767	0.0026	1.1	0.3579
sill * nugget * method * size	3	0.00460	0.0015	0.6	0.5855
sill * nugget * theta * size	1	0.00031	0.0003	0.1	0.7178
sill * method * theta * size	3	0.00181	0.0006	0.3	0.8582
nugget * method * theta * size	3	0.00237	0.0008	0.3	0.8019
sill * nugget * shape * size	1	0.03740	0.0374	15.7	$7.3e^{-5}$ ***
sill * method * shape * size	3	0.00398	0.0013	0.6	0.6424
nugget * method * shape * size	3	0.00464	0.0015	0.7	0.5821
sill * nugget * method * theta * size	3	0.01253	0.0042	1.8	0.1527
sill * nugget * method * shape * size	3	0.00800	0.0027	1.1	0.3381
Residuals	9288	22.0610	0.0024		

Significance: ***$\alpha < 0.001$ **$\alpha < 0.01$ *$\alpha < 0.05$

Table A.9: ANOVA, event × noise, MAD

	DF	Sums of Squares	Mean of Squares	F Value	P value
sill	1	31.2570	31.2570	4875.80	$< 2.2e^{-16}$ ***
nugget	1	12.9877	12.9877	2025.96	$< 2.2e^{-16}$ ***
method	3	9.2308	3.0769	479.97	$< 2.2e^{-16}$ ***
theta	1	0.3807	0.3807	59.39	$1.4e^{-14}$ ***
shape	1	0.4137	0.4137	64.54	$1.1e^{-15}$ ***
size	1	0.1453	0.1453	22.66	$2.0e^{-6}$ ***
sill * nugget	1	1.8703	1.8703	291.76	$< 2.2e^{-16}$ ***
sill * method	3	2.0582	0.6861	107.02	$< 2.2e^{-16}$ ***
nugget * method	3	0.7681	0.2560	39.94	$< 2.2e^{-16}$ ***
sill * theta	1	0.5144	0.5144	80.24	$< 2.2e^{-16}$ ***
nugget * theta	1	0.0840	0.0840	13.10	0.0003
method * theta	3	0.2544	0.0848	13.23	$1.3e^{-8}$ ***
sill *shape	1	0.8028	0.8028	125.23	$< 2.2e^{-16}$ ***
nugget * shape	1	0.3898	0.3898	60.81	$7.0e^{-15}$ ***
method * shape	3	0.4905	0.1635	25.51	$< 2.2e^{-16}$ ***
sill * size	1	4.6378	4.6378	723.46	$< 2.2e^{-16}$ ***
nugget * size	1	1.1058	1.1058	172.50	$< 2.2e^{-16}$ ***
method * size	3	2.5392	0.8464	132.03	$< 2.2e^{-16}$ ***
theta * size	1	0.2082	0.2082	32.48	$1.2e^{-8}$ ***
shape * size	1	0.4784	0.4784	74.63	$< 2.2e^{-16}$ ***
sill * nugget * method	3	0.4269	0.1423	22.20	$2.6e^{-14}$ ***
sill * nugget * theta	1	0.0001	0.0001	0.01	0.9325
sill * method * theta	3	0.0046	0.0015	0.24	0.8690
nugget * method * theta	3	0.0007	0.0002	0.04	0.9911
sill * nugget * shape	1	0.1069	0.1069	16.67	$4.5e^{-5}$ ***
sill * method * shape	3	0.0011	0.0004	0.06	0.9823
nugget * method * shape	3	0.0069	0.0023	0.36	0.7840
sill * nugget * size	1	0.2890	0.2890	45.07	$2.0e^{-11}$ ***
sill *method * size	3	0.0375	0.0125	1.95	0.1189
nugget * method * size	3	0.0433	0.0144	2.25	0.0800
sill * theta * size	1	0.0081	0.0081	1.26	0.2615
nugget * theta * size	1	0.0133	0.0133	2.08	0.1496
method * theta * size	3	0.1258	0.0419	6.54	0.0002 ***
sill * shape * size	1	0.0558	0.0558	8.70	0.0032 **
nugget * shape * size	1	0.1128	0.1128	17.60	$2.8e^{-5}$ ***
method * shape * size	3	0.2192	0.0731	11.40	$1.9e^{-7}$ ***
sill * nugget * method * theta	3	0.0277	0.0092	1.44	0.2283
sill * nugget * method * shape	3	0.0160	0.0053	0.83	0.4747
sill * nugget * method * size	3	0.0528	0.0176	2.75	0.0414 *
sill * nugget * theta * size	1	0.0131	0.0131	2.05	0.1523
sill * method * theta * size	3	0.0272	0.0091	1.41	0.2369
nugget * method * theta * size	3	0.0259	0.0086	1.35	0.2570
sill * nugget * shape * size	1	0.0098	0.0098	1.53	0.2165
sill * method * shape * size	3	0.0644	0.0215	3.35	0.0182 *
nugget * method * shape * size	3	0.0283	0.0094	1.47	0.2200
sill * nugget * method * theta * size	3	0.0079	0.0026	0.41	0.7457
sill * nugget * method * shape * size	3	0.0153	0.0051	0.80	0.4961
Residuals	9288	59.5421	0.0064		

Significance: ***$\alpha < 0.001$ **$\alpha < 0.01$ *$\alpha < 0.05$

Bibliography

[1] Ackoff, R. L. From data to wisdom. *Journal of Applies Systems Analysis 16* (1989), 3–9.

[2] Ahlqvist, O., Keukelaar, J., and Oukbir, K. Rough and fuzzy geographical data integration. *International Journal of Geographical Information Science 17.3* (2003), 223–234.

[3] Artusa, A. Current u.s. drought monitor. `http://droughtmonitor.unl.edu/`, 9 2013.

[4] Beeri, C., Doytsher, Y., Kanza, Y., Safra, E., and Sagiv, Y. Finding corresponding objects when integrating several geo-spatial datasets. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems* (2005), ACM, pp. 87–96.

[5] Bleiholder, J., and Naumann, F. Data fusion. *ACM Computing Surveys (CSUR) 41*, 1 (2008), 1.

[6] Bohling, G. Kriging. *Kansas Geological Survey 940* (2005).

[7] Bostrom, H., Andler, S. F., Brohede, M., Johansson, R., Karlsson, A., Laere, J., Niklasson, L., Nilsson, M., Persson, A., and Ziemke, T. ISIF Welcome. `http://www.isif.org/sites/isif.org/files/FULLTEXT01.pdf`, 2007.

[8] Carrara, P., Bordogna, G., Boschetti, M., Brivio, P. A., Nelson, A., and Stroppiana, D. A flexible multi-source spatial-data fusion system for environmental status assessment at continental scale. *International Journal of Geographical Information Science 22*, 7 (2008), 781–799.

[9] Center, N. D. M. Drought monitor: State-of-the-art blend of science and subjectivity. http://droughtmonitor.unl.edu/classify.htm, 1 2008.

[10] Chen, H., and Meer, P. Robust fusion of uncertain information. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 35*, 3 (2005), 578–586.

[11] Clark, I., and Harper, W. V. *Practical Geostatistics*. Ecosse North America LLC, 2000.

[12] CoCoRaHS. In depth snow measuring. http://www.cocorahs.org/media/docs/Measuring%20Snow-National-Training%201.1.pdf, 2011.

[13] CoCoRaHS. About us. http://www.cocorahs.org/Content.aspx?page=aboutus, 2013.

[14] Couclelis, H. *People Manipulate Objects (but Cultivate Fields): Beyond the Raster-Vector Debate in GIS*. Springer, 1992, ch. Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Lecture Notes in Computer Science, pp. 65–77.

[15] Deardorff, A. Determinants of bilateral trade: does gravity work in a neoclassical world? In *The regionalization of the world economy*. University of Chicago Press, 1998, pp. 7–32.

[16] Egenhofer, M. J., Clementini, E., and Di Felice, P. Evaluating inconsistencies among multiple representations. In *Proceedings of the Sixth International Symposium on Spatial Data Handling* (1994), vol. 2, Citeseer, pp. 901–920.

[17] Epstein, E. F. Liability insurance and the use of geographical information. *International Journal of Geographical Information Science 12*, 3 (1998), 203–214.

[18] ESRI. Arcgis resource center: Idw (spatial analyst). `http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//009z0000006m000000.htm`, June 2011.

[19] ESRI. Understanding indicator kriging. `http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00310000004n000000`, June 2013.

[20] Farah, I. R., Boulila, W., Ettabaa, K. S., Solaiman, B., and Ben Ahmed, M. Interpretation of multisensor remote sensing images: Multiapproach fusion of uncertain information. *Geoscience and Remote Sensing, IEEE Transactions on 46*, 12 (2008), 4142–4152.

[21] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. From data mining to knowledge discovery in databases. *AI magazine 17.3* (1996), 37–54.

[22] Feekin, A., and Chen, Z. Duplicate detection using k-way sorting method. In *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1* (2000), ACM, pp. 323–327.

[23] For Toxic Substances & Disease Registry, A. *Toxicological profile for radon*. U.S. Department of Health and Human Services, 2012.

[24] French, K., and Li, X. Feature-based cartographic modelling. *International Journal of Geographical Information Science 24.1* (2010), 141–164.

[25] Geary, R. C. The contiguity ratio and statistical mapping. *The incorporated statistician 5*, 3 (1954), 115–146.

[26] Gervais, M. On the importance of external data quality in civil law. *Fundamentals of Spatial Data Quality* (2006), 283–300.

[27] Graler, B. cokriging and indicator kriging. `http://ifgi.uni-muenster.de/~b_grae02/copulaIntro/02_Co-Kriging_Indicator-Kriging.pdf`, February 2011.

[28] Halevy, A., Rajaraman, A., and Ordille, J. Data integration: the teenage years. In *Proceedings of the 32nd international conference on Very large data bases* (2006), VLDB Endowment, pp. 9–16.

[29] Hua, B. Fusion of uncertain information using vague sets and dempster-shafer theories. In *Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on* (2009), IEEE, pp. 17–22.

[30] Hunter, G. J., Bregt, A. K., Heuvelink, G. B., De Bruin, S., and Virrantaus, K. *Spatial data quality: problems and prospects*. Springer, 2009.

[31] Ledoux, H., and Gold, C. A voroni-based map algebra. In *Progress in Spatial Data Handling–12th International Symposium on Spatial Data Handling* (2006), pp. 117–131.

[32] Lutz, M. *Programming Python*. O'Reilly, 2011.

[33] Mitchell, H. B. *Data Fusion: Concepts and Ideas*. Springer, 2007.

[34] Moran, P. A. Notes on continuous stochastic phenomena. *Biometrika 37*, 1/2 (1950), 17–23.

[35] Oceanic, N., and Service, A. A. N. W. Cooperative observer program. http://www.nws.noaa.gov/om/coop/, June 2013.

[36] Olmedo, O. E. Kriging: Ordinary kriging. r package version 1.0.1. http://CRAN.R-project.org/package=kriging.

[37] Pateiro-Lopez, B., and Rodriguez-Casal, A. *alphahull: Generalization of the convex hull of a sample of points in the plane*, 2011. R package version 0.2-1.

[38] Programme, U. N. D. Human development reports: Human development index. http://hdr.undp.org/en/statistics/hdi/.

[39] Rahm, E., and Bernstein, P. A. A survey of approaches to automatic schema matching. *The VLDB Journal 10*, 4 (2001), 334–350.

[40] Raper, J.; Livingstone, D. Development of a geomorphological spatial model using object-oriented design. *International Journal of Geographical Information Systems 9.4* (1995), 359–383.

[41] Rowley, J. The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science 33.2* (2007), 163–180.

[42] Service, S. C. A. Average annual precipitation nebraska. http://www.worldatlas.com/webimage/countrys/namerica/usstates/weathermaps/neprecip.htm, 2000.

[43] Sester, M., Anders, K.-H., and Walter, V. Linking objects of different spatial data sets by integration and aggregation. *GeoInformatica 2*, 4 (1998), 335–358.

[44] Shekhar, S., Evans, M., and Kang, J. Identifying patterns in spatial information: A survey of methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.3* (2011), 193–214.

[45] Solaiman, B., Pierce, L. E., and Ulaby, F. T. Multisensor data fusion using fuzzy concepts: application to land-cover classification using ers-1/jers-1 sar composites. *Geoscience and Remote Sensing, IEEE Transactions on 37*, 3 (1999), 1316–1326.

[46] Solberg, A. S., Jain, A. K., and Taxt, T. Multisource classification of remotely sensed data: fusion of landsat tm and sar images. *Geoscience and Remote Sensing, IEEE Transactions on 32*, 4 (1994), 768–778.

[47] Sowa, J. F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing, 1999.

[48] Srinivasan, B.-V., Duraiswami, R., and Murtugudde, R. Efficient kriging for real-time spatio-temporal interpolation. In *Proceedings of the 20th Conference on Probability and Statistics in the Atmospheric Sciences* (2010), pp. 228–235.

[49] Sun, W., Heidt, V., Gong, P., and Xu, G. Information fusion for rural land-use classification with high-resolution satellite imagery. *Geoscience and Remote Sensing, IEEE Transactions on 41*, 4 (2003), 883–890.

[50] ASTM Standard D5922-96. *Standard Guide for Analysis of Spatial Variation in Geostatistical Site Investigations*. American Society for Testing and Materials, 2010.

[51] Tobler, W. A computer movie simulating urban growth in the detroit region. *Economic Geography* (1970), 234–240.

[52] Tobler, W. Three presentations on geographical analysis and modeling. *National Center for Geographic Information and Analysis Technical Report* (1993).

[53] Tung, A., Han, J., Ng, R., and Lakshmanan, L. *Lecture Notes in Computer Science-Database Theory*. Springer, 2001, ch. Constrained Clustering on Large Databases.

[54] Valet, L., Mauris, G., and Bolon, P. A statistical overview of recent literature in information fusion. In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on* (2000), vol. 1, IEEE, pp. MOC3–22.

[55] Wan, B., Yang, L., Hu, M., and Ye, Y. An intelligent multilayered middleware model and heterogeneous spatial data fusion application study. In *Environmental Science and Information Application Technology, 2009. ESIAT 2009. International Conference on* (2009), vol. 1, IEEE, pp. 423–427.

[56] Wentz, E. A., Peuquet, D. J., and Anderson, S. An ensemble approach to space–time interpolation. *International Journal of Geographical Information Science 24*, 9 (2010), 1309–1325.

[57] Wingle, W. *Evaluating Subsurface Uncertainty Using Modified Geostatistical Techniques*. Colorado School of Mines, 1997.

[58] Wingle, W. L., Poeter, E. P., and McKenna, S. A. UNCERT geostatistics, uncertainty analysis and visualization software applied to groundwater flow and contaminant transport modeling. *Computers & Geosciences 25.4* (1999), 365–376.

[59] Zins, C. Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology 58.4* (2007), 479–493.